## Contents

### Preface
- Audience  
- Downloading Oracle Communications Documentation  
- Documentation Accessibility  
- Document Revision History

### 1 List of Billing Features
- Basic Billing Tasks 1-1  
- Advanced Billing Features 1-1  
- Advanced Billing Configuring Tasks 1-2

### 2 Running Billing Scripts
- Running the Billing Scripts 2-1  
- Changing the Path in the Billing Scripts 2-3  
- Running the pin_bill_day Script for Bill Run Management 2-3  
- Handling Billing Failures 2-3

### 3 About the Billing Utilities
- Billing Accounts By Using the pin_bill_accts Utility 3-1  
  - When to Run the pin_bill_accts Utility 3-2  
  - About Bill States and the pin_bill_accts Utility 3-2  
- Prorating Cycle-Forward Fees and Canceling Charge Offers By Using the pin_cycle_fees Utility 3-2  
  - When to Run the pin_cycle_fees Utility 3-3  
  - Improving Performance of the pin_cycle_fees Utility 3-3

### 4 Setting Default Billing Properties for Account Creation
- Setting the Default Accounting Day of Month 4-1  
- Setting the Default Billing-Cycle Length 4-2
## Configuring 31-Day Billing

- **About Using 31-Day Billing**  
- **Enabling the 31-Day Billing Feature**
  - Switching to 31-Day Billing during BRM Installation
  - Switching to 31-Day Billing after You Install BRM
- **Setting the Forward and Back Billing Options**

## Configuring the Billing Cutoff Time

- **About Configuring the BRM Cutoff Time**
  - How Billing and Invoicing Are Affected by Changing the Cutoff Time
  - How Rating Is Affected by Changing the Cutoff Time
  - How General Ledger (G/L) Is Affected by Changing the Cutoff Time
  - How Timestamp Fields Are Affected by Changing the Cutoff Time
- **Configuring the Billing Cutoff Time**

## Configuring Billing for Groups and Hierarchies

- **Setting Up Billing for Charge and Discount Sharing Groups**
- **Skipping Validation of Billing for Nonpaying Child Bill Units**

## Load Balancing Billing Runs

- **About Managing Billing Cycles**
- **Implementing Bill Cycle Management**
  - Setting Up Billing Segments
    - Editing the Billing Segment Configuration File
    - Updating Billing Segments
  - Associating Bill Units with Billing Segments
    - Changing a Bill Unit's Billing Segment
  - Assigning Accounting Days of Month to Bill Units in Billing Segments
    - Manually Assigning a Billing DOM
    - Automatically Assigning a Billing DOM
    - Changing a Bill Unit's Billing DOM
About Proration

Calculating Prorated Cycle Fees

Calculating the Unit Interval
- Calculating Unit Interval When use_number_of_days_in_month Is Not Set or 0
- Calculating Unit Interval When use_number_of_days_in_month Is Set to 1
- Calculating Unit Interval When Billing Day of Month Is 29, 30, or 31

Examples of Proration
- Example 1: Use_number_of_days_in_month Is Not Set or Set to 0
- Example 2: Use_number_of_days_in_month Is Set to 1

Examples Using the 29th, 30th, and 31st for Billing Day of Month
- Example 3a: Use Forward Option with use_number_of_days_in_month Set to 0
- Example 3b: Use Forward Option with use_number_of_days_in_month Set to 1
- Example 3c: Use Back Option with use_number_of_days_in_month Set to 1
- Example 3d: Use Back Option with use_number_of_days_in_month Set to 0

Proration for Special Cases

Special Cases

Addressing Special Cases

About 30-Day-Based Proration

Examples of 30-Day-Based Proration
- Example 6: Prorated Purchase Fee with 31-day Billing Cycle
- Example 7: Prorated Cancel Fee with 31-day Billing Cycle
- Example 8: Prorated Purchase Fee with 28-Day Billing Cycle
- Example 9: Prorated Cancel Fee with 28-Day Billing Cycle

Special Cases
- Example 10: Full Purchase Fee Charged When Service Is Provided for 1 Day Less
- Example 11: Full Cancel Fee Refunded When Service Has Been Used for 1 Day

Enabling 30-Day-Based Proration

Using Two Events to Prorate Charges for Charge Offers Whose Validity Ends in First Cycle

Prorating Cycle Fees after a Discount Purchase or Cancellation

Examples of Cycle Fee Proration
- Example 12: Cycle Fee Is Refunded after a Discount Purchase
- Example 13: Minutes Are Prorated after a Discount Cancellation
- Example 14: Canceled Discount Proration Is Not Taken into Account When Charge Offer Is Canceled

Prorating Cycle Fees When a Discount’s Cycle Start or End Date Is Changed
Examples of Cycle Fee Proration

Example 15: Cycle Fee Is Refunded When a Discount's Cycle Start Date Is Changed 13-21
Example 16: Cycle Fee Is Charged When a Discount's Cycle Start Date Is Changed 13-21
Example 17: Cycle Fee Is Refunded When Discount's Cycle End Date Is Changed 13-22
Example 18: Cycle Fee Is Charged When Discount's Cycle End Date Is Changed 13-23

14 Managing Large Billing Runs

Billing Only Specified Accounts and Bill Units 14-1
Splitting a Billing Run into Multiple Runs 14-3
Configuring Auto-Triggered Billing for Bill Run Management 14-3
Configuring a Split Billing Run 14-4
  About Sponsored Charges in Split Billing Runs 14-7
  Including Only Specified Billing DOMs in Billing Runs 14-7
  Including Only Specified Billing Segments in Billing Runs 14-7
  Sample Billing Run Configuration File 14-8
  Validating Your Billing Run Configuration File Edits 14-8
Managing Bill Due Dates 14-8
Managing Payment Terms 14-9
  Editing the Payment Terms Configuration File 14-9
  Loading Payment Terms 14-10
  Updating the pin.conf File to Use Payment Terms 14-11
  Assigning Payment Terms to Bill Units 14-11
Managing Billing Calendars 14-12
  Setting Up Billing Calendars 14-12
  Editing the Billing Calendar Configuration File 14-13
  Updating Billing Calendars 14-17
  Associating Billing Calendars with Payment Terms 14-17
Specifying Due Date Adjustments in a Billing Run 14-17
  Editing the Billing Run Configuration File to Adjust Bill Due Dates 14-19
  Sample Billing Run Configuration File 14-20
  Validating Your Billing Run Configuration File Edits 14-21

15 About Bill Suppression

About Suppressing Bills 15-1
  About Automatic Bill Suppression 15-1
  About Manual Bill Suppression 15-2
16  Creating Custom Bill Items

About Custom Bill Items 16-1
About Defining Custom Bill Items 16-1
Tracking Charges in Bill Items 16-1
About Creating /item Objects 16-2
About Assigning Custom Bill Items to Events 16-2
   About Using Event and Service Combinations to Assign Bill Items 16-3
   About Using Event Attributes to Assign Bill Items 16-5
How BRM Assigns Custom Bill Items to Events 16-5
   Cumulative Custom Item for Taxes 16-6
Setting Up BRM to Assign Custom Bill Items to Events 16-6
   Assigning Item Tags Based on Event and Service Combinations 16-7
   Assigning Item Tags Based on Event Attributes 16-8
      Setting Up Online Charging to Assign Items Based on Event Attributes 16-8
   Mapping Item Tags to Item Types 16-8
Assigning Bill Items to Event Balance Impacts 16-11
Creating Custom Sponsored Bill Items 16-12
   Splitting Sponsored Charges into Multiple Items 16-13

17  Creating Corrective Bills

About Corrective Bills 17-1
Corrective Bills and Billing Cycles 17-3
Configuring Corrective Billing 17-3
   Enabling Corrective Billing 17-3
   Configuring Accounts and Bills for Corrective Billing 17-4
   Restricting Corrective Billing Permissions 17-4
Configuring Bill Numbers for Corrective Bills 17-4
Customizing Correction Reasons 17-5
Specifying the Minimum Threshold Amount for Corrective Bills 17-6
# Running Trial Billing

About Trial Billing 18-1
- About Trial Billing for Bill Unit Hierarchies and Sharing Groups 18-2
About Trial Invoices 18-3
About Collecting Revenue Assurance Data from Trial Billing 18-3
Configuring Trial Billing 18-4
- Specifying Accounts for Trial Billing 18-4
-Specifying Bill Units, Billing Segments, and DOMs for Trial Billing 18-5
- Specifying Bill Units for Trial Billing 18-6
- Specifying Accounting DOMs for Trial Billing 18-7
- Specifying Billing Segments for Trial Billing 18-7
Running Trial Billing 18-8
- Running Trial Billing With Date Ranges 18-8
- Running Trial Billing According to Payment Type 18-9
Creating Trial Bills without Generating Trial Invoices 18-10
Purging Trial Invoices 18-11
Exporting Trial Invoices 18-11

# Suspending Billing

About Suspending Billing of Accounts and Bills 19-1
Suspending Billing of Closed Accounts 19-1
Suspending Billing of an Account's Bill 19-2

# Setting Up Billing in a Multischema Environment

Setting Up Billing to Run in a Multischema Environment 20-1
Running Billing on One Schema at a Time 20-1
Running Billing on Multiple Schemas Simultaneously 20-2
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>load_config_item_tags</td>
<td>21-1</td>
</tr>
<tr>
<td>load_config_item_types</td>
<td>21-2</td>
</tr>
<tr>
<td>load_pin_bill_suppression</td>
<td>21-3</td>
</tr>
<tr>
<td>load_pin_billing_segment</td>
<td>21-5</td>
</tr>
<tr>
<td>load_pin_calendar</td>
<td>21-6</td>
</tr>
<tr>
<td>load_pin_payment_term</td>
<td>21-7</td>
</tr>
<tr>
<td>pin_bill_accts</td>
<td>21-9</td>
</tr>
<tr>
<td>pin_cycle_fees</td>
<td>21-13</td>
</tr>
<tr>
<td>pin_make_corrective_bills</td>
<td>21-15</td>
</tr>
<tr>
<td>pin_rollover</td>
<td>21-18</td>
</tr>
<tr>
<td>pin_trial_bill_accts</td>
<td>21-19</td>
</tr>
<tr>
<td>pin_trial_bill_purge</td>
<td>21-22</td>
</tr>
</tbody>
</table>
Preface

This guide describes how to configure and run billing in Oracle Communications Billing and Revenue (BRM).

Audience

This guide is intended for operations personnel and system administrators.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Technology Network:

http://docs.oracle.com

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

https://edelivery.oracle.com

Documentation Accessibility

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

Access to Oracle Support

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

Document Revision History

The following table lists the revision history for this guide:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E51002-01</td>
<td>December 2017</td>
<td>Initial release.</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>E51002-02</td>
<td>September 2019</td>
<td>Documentation Updates for BRM 12.0 Patch Set 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuring Auto-Triggered Billing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuring Auto-Triggered Billing for Delayed Billing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuring Delayed Billing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifying How to Handle Partial Accounting Cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuring 31-Day Billing</td>
</tr>
</tbody>
</table>
List of Billing Features

This document lists the billing features in Oracle Communications Billing and Revenue Management (BRM).

To learn about billing, see BRM Concepts.

Basic Billing Tasks

This table lists basic tasks for configuring and running billing.

<table>
<thead>
<tr>
<th>Task</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn about billing utilities</td>
<td>About the Billing Utilities</td>
</tr>
<tr>
<td>Run billing</td>
<td>Running the Billing Scripts, Changing the Path in the Billing Scripts, Running the pin_bill_day Script for Bill Run Management, Handling Billing Failures</td>
</tr>
<tr>
<td>Setting standard billing options</td>
<td>Setting the Default Accounting Day of Month, Setting the Default Billing-Cycle Length, Specifying the Maximum Number of Months Allowed in Billing Cycles, Setting the Default Accounting Type, Setting the First Billing Cycle to the Day after Account Creation, Specifying the Minimum Payment to Collect</td>
</tr>
<tr>
<td>Improve billing performance</td>
<td>Load Balancing Billing Runs, Managing Large Billing Runs, See also BRM System Administrator's Guide</td>
</tr>
</tbody>
</table>

Advanced Billing Features

This table lists advanced billing features.

<table>
<thead>
<tr>
<th>Task</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Bill Now to bill accounts at any time</td>
<td>Configuring Bill Now</td>
</tr>
<tr>
<td>Configure billing when usage events arrive late</td>
<td>Setting Up Delayed Billing</td>
</tr>
<tr>
<td>Configure billing to use all 31 calendar days</td>
<td>Configuring 31-Day Billing</td>
</tr>
<tr>
<td>Run corrective billing</td>
<td>Creating Corrective Bills</td>
</tr>
<tr>
<td>Validate billing results without billing customers</td>
<td>Running Trial Billing</td>
</tr>
<tr>
<td>Suspend billing for accounts or bill units that you do not want to bill.</td>
<td>Suspending Billing</td>
</tr>
</tbody>
</table>
### Advanced Billing Configuring Tasks

This table lists billing features.

| Task                                                                 | Topics                                                                 |
|                                                                     |                                                                       |
| Configure billing for groups and hierarchies                        | Setting Up Billing for Charge and Discount Sharing Groups               |
|                                                                      | Skipping Validation of Billing for Nonpaying Child Bill Units          |
| Learn about how BRM prorates recurring charges                       | About Proration                                                        |
| Postpone finalizing a bill until the end of a future billing cycle. | About Bill Suppression                                                 |

| Task                                                                 | Topics                                                                 |
|                                                                     |                                                                       |
| Setting advanced billing options                                     | Customizing the Format of Bill and Invoice Numbers                     |
|                                                                      | Specifying When to Apply Custom Bill Numbers                           |
|                                                                      | Configuring Auto-Triggered Billing                                    |
|                                                                      | Setting the Bill Unit Status When Billing Errors Occur                 |
| Manage billing and accounting cycles                                 | Specifying How to Handle Partial Accounting Cycles                     |
|                                                                      | Configuring Timestamp Rounding                                         |
|                                                                      | Aligning Account and Cycle Start and End Times                         |
|                                                                      | Defining When Billing-Time Discounts Are Applied                       |
|                                                                      | Including Previous Balances in the Current Amount Due in Open Item Accounting |
|                                                                      | Specifying Which Billing Cycle to Assign to Deferred Purchase Fees    |
|                                                                      | Calculating Cycle Fees for Backdating                                  |
| Configure the billing cutoff time, for example, if you set the      | Configuring the Billing Cutoff Time                                    |
| cutoff time to 10 a.m., activity for events that occurred before    |                                                                       |
| 10 a.m. are billed.                                                  |                                                                       |
| Create custom bill items to track different types of charges.        | Creating Custom Bill Items                                             |
| Set up billing in a multischema environment                          | Setting Up Billing in a Multischema Environment                         |
Running Billing Scripts

This document describes how to run the Oracle Communications Billing and Revenue Management (BRM) billing utilities.

Before running billing, you must set the billing configuration defaults, as described in "Configuring Billing".

Topics in this document:

• Running the Billing Scripts
• Changing the Path in the Billing Scripts
• Running the pin_bill_day Script for Bill Run Management
• Handling Billing Failures

See also:

• About the Billing Utilities
• List of Billing Features
• BRM Concepts

Running the Billing Scripts

Billing scripts run one or more billing utilities on a daily, weekly, or monthly basis. By default, billing scripts are located in BRM_home/bin.

You can customize the scripts.

Note the following:

• Use billing scripts for regular bills only, not for corrective bills.
• Run billing as pin_user, not as root. Running billing as pin_user provides the read/write permissions for billing.
• Since billing generates a lot of system activity, it's best to run billing scripts at night.
• Use a different time for all three scripts so you do not run billing utilities simultaneously.
• Check for billing errors daily. The log file for the billing utilities is pin_billd.pinlog.

To run the billing scripts, use a cron job with a crontab entry.

• The following crontab entry runs pin_bill_day at 1:00 a.m. daily:

  0 1 * * * BRM_home/bin/pin_bill_day &

• The following crontab entry runs pin_bill_week every Sunday at 12:05 a.m.

  5 0 * * 0 BRM_home/bin/pin_bill_week &
Note:

By default, the **pin_bill_week** script runs the **pin_collect** utility with the utility’s **end** parameter set to **7**. If you modify the script to run the utility with the **end** parameter set to **1** or **0**, do not run the script at the same time that you run the **pin_bill_day** script. If you do, accounts whose payment collection date is on the day or the day before the utility runs may by double charged.

- The following **cron** entry runs **pin_bill_month** at 12:05 a.m. on the first day of the month:

  ```
  5 0 1 * * BRM_home/bin/pin_bill_month &
  ```

Table 2-1 shows the billing utilities in each script:

### Table 2-1 Utilities in Billing Scripts

<table>
<thead>
<tr>
<th>Billing Script</th>
<th>Description</th>
</tr>
</thead>
</table>
| **pin_bill_day** | Run daily. Runs the following billing utilities:  
  - **pin_deferred_act**. Runs deferred actions.  
  - **pin_bill_accts**. Bills accounts.  
  - **pin_collect**. Collects BRM-initiated payments. See *BRM Configuring and Collecting Payments*.  
  - **pin_refund**. Grants refunds. See *BRM Managing Accounts Receivable*.  
  - **pin_inv_accts**. Generates invoices.  
  - **pin_deposit**. Deposits pre-authorized credit card payments, such as payments authorized when creating an account. See *BRM Configuring and Collecting Payments*.  
  - **pin_cycle_fees**. Prorates balance impacts for cycle forward fees. See "*Prorating Cycle-Forward Fees and Canceling Charge Offers By Using the pin_cycle_fees Utility*". |
| **pin_bill_week** | Run weekly. Runs the **pin_collect** utility with the **rebill** option on all active accounts with a payment collection date at least 8 days old. Collects outstanding balances from active credit card or direct debit accounts that could not be collected during regular daily billing.  
For example, the daily billing run might return a soft decline on a BRM-initiated payment. In that case, the payment is not collected, but the bill is left open so that the **pin_collect** utility can attempt to collect the payment again when you run the **pin_bill_week** script. |
| **pin_bill_month** | Run monthly. Runs the **pin_collect** utility with the **rebill** option on all closed and inactive accounts with a payment collection date at least 31 days old. Collects outstanding balances from closed or inactive accounts. |
Changing the Path in the Billing Scripts

You can change the path to the billing utilities in the billing scripts. For example, to change the path for the `pin_bill_day` script, edit these lines:

```
PINDIR=/opt/portal/${VERSION}
CNFDIR=${PINDIR}/apps/pin_billd
NVDIR=${PINDIR}/apps/pin_inv
OGDIR=/var/portal/${VERSION}/pin_billd
PATH=/usr/bin:/bin:${PINDIR}/bin
cd ${CNFDIR}
```

Running the `pin_bill_day` Script for Bill Run Management

You must run the `pin_bill_day` script manually instead of automatically to do the following:

- Reduce the load and duration of a large daily billing run. See "Configuring a Split Billing Run".
- Add days to the due dates of bills in a daily billing run at run-time. See "Specifying Due Date Adjustments in a Billing Run".

When running `pin_bill_day` manually with bill run management, use this syntax:

```
pin_bill_day -file file_name
```

Where `file_name` is the name and location of a billing run configuration file.

The `-file` parameter when used with `pin_bill_accts`, affects only the `pin_bill_accts` utility.

Note:

When you use a `cron` job to run `pin_bill_day`, do not include the configuration file name. If you do, some bill units might never be billed.

Handling Billing Failures

Billing can fail in the following cases:

- When an internal BRM component, such as a CM or DM, goes offline.
- When the online payment processor goes offline.
- When a connection between BRM components is broken.

If the `pin_collect` utility or the `pin_deposit` utility is interrupted in progress, you can run it again. However, you might need to resolve failed BRM-initiated payment transactions. See "Resolving Failed BRM-Initiated Payment Transactions" in BRM Configuring and Collecting Payments.

All other billing utilities, `pin_cycle_fees`, `pin_deferred_act`, `pin_inv_accts`, and `pin_bill_accts`, can be run again. You do not need to resolve failed transactions.
If the billing utilities were not run at all, for example, if the database was offline, you can run all of the billing utilities with no problems. The `pin_bill_accts` utility bills all accounts that are due for billing, not just those that are due on the day that you run the utility.

The `pin_collect` utility and `pin_inv_accts` utility act on accounts that were billed by the `pin_bill_accts` utility, so if you run the `pin_bill_accts` utility first, payments for all accounts that are due are collected or invoiced.
About the Billing Utilities

This document describes the main billing utilities in Oracle Communications Billing and Revenue Management (BRM). For information about all billing utilities, see “Billing Utilities”.

Topics in this document:

• Billing Accounts By Using the pin_bill_accts Utility
• Prorating Cycle-Forward Fees and Canceling Charge Offers By Using the pin_cycle_fees Utility

See also:

• Running Billing Scripts
• List of Billing Features
• BRM Concepts

Billing Accounts By Using the pin_bill_accts Utility

Use the **pin_bill_accts** utility to generate regular bills (not corrective bills).

The **pin_bill_accts** utility calculates the balance due for each account bill unit, and creates a bill for the balance due. It creates bills for accounts whose billing date is any day before midnight of the day that you run the **pin_bill_accts** utility as shown in Figure 3-1.

**Figure 3-1    Accounts Included when Running pin_bill_accts**

<table>
<thead>
<tr>
<th>Midnight</th>
<th>Midnight</th>
<th>Midnight</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 1</td>
<td>May 2</td>
<td></td>
</tr>
</tbody>
</table>

Running **pin_bill_accts** any time on May 2 bills accounts with a billing date of May 1 or earlier.

The balance due amount is requested as a payment by the **pin_collect** utility, and the amount shown on the invoice.

The **pin_bill_accts** utility also performs the accounting cycle activity, such as creating new bill items.

For information about the **pin_bill_accts** utility syntax, see "**pin_bill_accts**".
When to Run the pin_bill_accts Utility

Use the `pin_bill_day` script to run the `pin_bill_accts` utility daily.

If you use bill unit hierarchies, you must run the `pin_bill_accts` utility to bill nonpaying child bill units before their paying parent bill units. The correct order is set in the `pin_bill_day` script.

You must run the `pin_bill_accts` utility before you run `pin_collect` because `pin_collect` needs the balance due amount collected by the `pin_bill_accts` utility.

About Bill States and the pin_bill_accts Utility

The bill unit stores the state of a bill:

• 0 to indicate that the bill has been finalized.
• 1 to indicate that partial billing has been completed.
• 2 to indicate that all cycle charges and billing-time discounts have been applied at the end of the billing period and the bill needs to be finalized. A bill advances to this state only after billing is run for its account with the `pin_bill_accts - cycle_charge_only` parameter.

Running the `pin_bill_accts` utility with the `-cycle_charge_only` parameter runs billing on accounts with bill states 0 and 1. All cycle fees and billing-time discounts are applied, but totals are not calculated, and the bill is not finalized. At the end of the billing run, the bill state is set to 2.

To finalize a bill, run the `pin_bill_accts` utility with the `-finalize_bill` parameter. This runs billing on accounts with bill state 2. Totals are calculated, and items, bills, and so on are updated. At the end of the billing run, the bill state is set to 0.

Prorating Cycle-Forward Fees and Canceling Charge Offers By Using the pin_cycle_fees Utility

The `pin_cycle_fees` utility performs two tasks:

• Use this utility to apply charges for cycle-forward fees that have reached the end of free billing periods. For example, if a customer signs up for one month of free service, the `pin_cycle_fees` utility finds when the free period is over, and applies the cycle-forward fee balance impact to the customer's account balance group.
• Use this utility to cancel charge offers that have an expired pending cancellation. For example, if a charge offer is set to cancel at a future date, the `pin_cycle_fees` utility cancels the charge offer.

Note:

These two tasks are performed by running the `pin_cycle_fees` utility twice with different parameters.
If a free period ends before the customer's billing date, the `pin_cycle_fees` utility calculates the prorated fees for the time between the end of the free period and the start of the customer's next accounting cycle.

For example, a customer opens an account on February 15 and is given one free month, but the customer's billing date is on the 1st of the month. When you run `pin_cycle_fees` on March 15, it finds that the customer's free time period has ended. The utility then assesses the prorated fee due for March 15 through March 31 and impacts the customer's balance with the prorated amount. The result is that the system makes no charges to the customer on March 1, but charges the prorated fee and the cycle fee on April 1 as shown in Figure 3-2.

**Figure 3-2 Proration of Cycle Forward Fees by `pin_cycle_fees`**

<table>
<thead>
<tr>
<th>February 15</th>
<th>March 1</th>
<th>March 15</th>
<th>April 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free month</td>
<td>Prorated cycle fee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account created</td>
<td>Free month ends</td>
<td>Billing date</td>
<td></td>
</tr>
</tbody>
</table>

For information about the `pin_cycle_fees` utility syntax, see "pin_cycle_fees".

**When to Run the `pin_cycle_fees` Utility**

Use the `pin_bill_day` script to run the `pin_cycle_fees` utility daily. This applies the prorated balance impacts as soon as they are due. If you do not run the `pin_cycle_fees` utility daily, the `pin_bill_accts` utility applies the balance impacts for the expired cycle forward fees. The only difference is that the balance impacts are not calculated by the `pin_cycle_fees` utility on the day that the cycle-forward fee expires.

**Improving Performance of the `pin_cycle_fees` Utility**

If system performance slows unacceptably when running the `pin_cycle_fees` utility, edit the `pin_bill_day` script and change the default `start` and `end` parameters for the `pin_cycle_fees` utility to every other day or every third day.
Setting Default Billing Properties for Account Creation

This document describes how to configure Oracle Communications Billing and Revenue Management (BRM) billing and to set up systemwide billing defaults, such as the default billing-cycle length.

Topics in this document:

- Setting the Default Accounting Day of Month
- Setting the Default Billing-Cycle Length
- Specifying the Maximum Number of Months Allowed in Billing Cycles
- Setting the Default Accounting Type
- Setting the First Billing Cycle to the Day after Account Creation

See also:

- List of Billing Features
- BRM Concepts

Setting the Default Accounting Day of Month

Note:

It is a good idea to leave the accounting DOM set to the date the account was created. This distributes the load for the billing utilities throughout the month.

To set the default accounting day of month (DOM):

1. Open the Connection Manager (CM) configuration file (`BRM_home/sys/cml/pin.conf`) in a text editor. `BRM_home` is the directory where you installed BRM components.
2. Uncomment the following line and enter a value from 1 to 28.
   ```
   - fm_cust_pol actg_dom 28
   ```

Note:

To use the day that the account was created as the default, comment out the line by using the pound (#) symbol.
3. Save and close the file.
The new value becomes effective immediately and applies to the next account created. You do not need to restart the CM to enable this entry.

Setting the Default Billing-Cycle Length

To set the default billing-cycle length:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Add the following line and enter the number of months in one billing cycle.

   - fm_cust_pol bill_when 2

3. Save and close the file.
The new value becomes effective immediately and applies to the next account created. You do not need to restart the CM to enable this entry.

Specifying the Maximum Number of Months Allowed in Billing Cycles

A billing cycle defines the time period between bills, which corresponds to the billing frequency (how often a bill is generated). For example, if the billing cycle is three months, a bill is generated every three months.

The default maximum number of months allowed in billing cycles is 24.

To change the maximum number of months allowed in billing cycles, modify the value of the bill_when field.

Setting the Default Accounting Type

You can set the default accounting type for all bill units (billinfo objects) by using the CM pin.conf file. BRM uses this setting during account creation only when an accounting type is not passed in the input flist of the Customer FM standard opcode.

To set the default accounting type:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Add the following line and enter the appropriate value.
   - To set the default accounting type to open item accounting, enter 1.
   - To set the default accounting type to balance forward accounting, enter 2. This is the default.

   - fm_cust_pol actg_type 1

---

Note:
The default is 1, which is monthly billing.
3. Save and close the file.

The new value becomes effective immediately and applies to the next account created. You do not need to restart the CM to enable this entry.

**Setting the First Billing Cycle to the Day after Account Creation**

Normally, an account is billed after one month on the day on which the account is created. For example, if an account is created on January 10, the account is billed on February 10, then on March 10, April 10, and so on. However, you can set the first billing date to be the day after account creation. For example, if an account is created on December 16, the account is billed on December 17. After the first billing run, all remaining bills for the account are generated normally. In this example, the account is billed on January 17, February 17, and so on. This option is called *advance billing cycle*.

To set the first billing cycle to the day after account creation:

1. Open the CM configuration file (*BRM_home/sys/cm/pin.conf*) in a text editor.
2. Add the following line and set the value to 1.
   
   - `fm_bill advance_bill_cycle 1`

   **Note:**

   To set the first billing cycle to one month after the account is created, comment out the line by using the pound (`#`) symbol.

3. Save and close the file.
4. Stop and restart the CM.
5

Configuring Billing

This document describes how to configure Oracle Communications Billing and Revenue Management (BRM) billing and to set up systemwide billing defaults, such as the default billing-cycle length.

Topics in this document:

• Specifying the Minimum Payment to Collect
• Customizing the Format of Bill and Invoice Numbers
• Specifying When to Apply Custom Bill Numbers
• Configuring Auto-Triggered Billing
• Setting the Bill Unit Status When Billing Errors Occur

See also:

• List of Billing Features
• BRM Concepts

Specifying the Minimum Payment to Collect

You can specify the minimum payment for billing. The pin_collect billing utility retrieves only those account bill units with an amount due greater than the minimum you specify. Charges accrue in the account balances associated with the bill unit until they reach the minimum amount, and then the amount due is collected.

The minimum value is expressed in terms of the account currency.

By default, the minimum amount is 2.

1. Open the billing utility configuration file (BRM_home/apps/pin_billd/pin.conf) in a text editor.
2. Change the value of the minimum entry.
3. Save the file.

Setting the Minimum Amount to Charge

To set the default minimum amount to charge to a credit-card:

• Customize the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode. See BRM Opcode Guide.
• Specify the amount in the CM pin.conf file minimum_payment entry.

There is no BRM configuration entry to set the minimum charge for accounts that pay their bills by using the invoice payment method.
Customizing the Format of Bill and Invoice Numbers

Bill numbering for regular bills is B1-1, B1-2, B1-3, and so on. For corrective bills, the default numbering is CB1-1, CB1-2, CB1-3, and so on. The default numbering takes the format PrefixDB_num-Seq_Number, where:

- **Prefix** represents the prefix to be used for the bill. B is used for regular bills, C is used for corrective bills. The number
- **DB_num** indicates the number of the database in which the original bill is stored. For example, when DB_num is 1, 3, or 7, the prefixes for regular bills are B1, B3, or B7.
- **Seq_Number** is the sequence number.

To customize how BRM formats bill numbers and invoice numbers, create a custom application that calls the PCM_OP_BILL_POL_SPEC_BILLNO opcode. See BRM Opcode Guide.

Specifying When to Apply Custom Bill Numbers

For corrective bills, BRM assigns a new bill number when it generates the corrective bill only. It does not support applying bill numbers at any other time in the corrective billing process.

For regular bills, you can control when custom bill numbers must be assigned to account bill units, which can be useful for revenue and expense accounting purposes. Custom bill numbers can be applied at the beginning of the first accounting cycle or at the end of the previous accounting cycle for multimonth billing cycles.

To specify when BRM assigns custom bill numbers:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Change the value of the custom_bill_no entry.
   - 0 assigns custom bill numbers at the end of the previous accounting cycle. This is the default.
   - 1 assigns custom bill numbers at the beginning of the first accounting cycle.
3. Stop and restart the CM.

Configuring Auto-Triggered Billing

If you perform no offline usage charging, you might not need auto-triggered billing. You can enable or disable auto-triggered billing by setting the following business parameters:
• **ConfigBillingDelay**. By default, the **ConfigBillingDelay** parameter is set to 0. If you are using delayed billing, you must set it to a value greater than 0.

When billing delay is specified, the system maintains an internal list of bill items for both the previous billing cycle and the current billing cycle so that new events impact bill items of the current billing cycle and old events impact bill items of the previous billing cycle.

• **AutoTriggeringLimit**. By default, **AutoTriggeringLimit** is set to 2. For example, if billing for the previous cycle has not occurred and the billing for the current cycle is due in the next two days, then billing for the previous cycle is auto-triggered within these two days.

You set these parameters by running the **pin_bus_params** utility. For information on this utility, see *BRM Developer's Guide*.

To set the **ConfigBillingDelay** parameter, see "Configuring Delayed Billing".

![Note:](image)

> If you do not use delayed billing but want to disable auto-triggered billing, set **ConfigBillingDelay** to 0.

To set the **AutoTriggeringLimit** parameter:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, enter a value greater than 0.
   ```
   <AutoTriggeringLimit>5</AutoTriggeringLimit>
   ```
4. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing.xml
   ```
5. Stop and restart the CM.
6. (Multischema systems only) Run the **pin_multidb** script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

## Setting the Bill Unit Status When Billing Errors Occur

When the billing utility (**pin_bill_accts**) encounters an error while generating a bill for a bill unit, the utility sets the billing status of the bill unit to PIN_BILL_ERROR. Bill units with an error status are not selected when billing is run.
When billing fails for a nonpaying child bill unit, the status of the nonpaying bill unit and of the paying parent bill unit are both set to PIN_BILL_ERROR. This ensures that when billing fails for a nonpaying bill unit, the paying bill unit is also not billed. Otherwise, the parent bill might not include charges from the nonpaying bill unit, resulting in incorrect billing.

After you have resolved the billing errors, you can rerun billing for the failed bill units by running the billing utility with the `-retry` option. See “pin_bill_accts”.

To set the bill unit status when billing errors occur:

1. Open the billing utility configuration file (`BRM_home/apps/pin_billd/pin.conf`) in a text editor.
2. Add the following line and enter the appropriate value:
   - `0` sets the billing status in the `/billinfo` object. This is the default.
   - `1` does not set the billing status in the `/billinfo` object.

   `- pin_bill_accts unset_error_status 1`
3. Save the file.
4. Run the billing utility.
Managing Billing and Accounting Cycles

This document describes how to manage billing and accounting cycles in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

• Specifying How to Handle Partial Accounting Cycles
• Configuring Timestamp Rounding
• Aligning Account and Cycle Start and End Times
• Defining When Billing-Time Discounts Are Applied
• Specifying Which Billing Cycle to Assign to Deferred Purchase Fees
• Calculating Cycle Fees for Backdating
• Load Balancing Billing Runs

See also:

• List of Billing Features
• BRM Concepts

Specifying How to Handle Partial Accounting Cycles

When you change the accounting cycle date in the middle of an accounting cycle, the new date does not take effect until after the current accounting cycle is over. This results in a gap of time between the end of the old accounting cycle and the start of the new accounting cycle.

For example, for a 30-day month, if the current accounting cycle ends on the 15th and the new cycle starts on the 1st, there is a gap of 15 days between the end of the old cycle and the start of the new cycle. By default, the BRM system treats those 15 days as a short, but complete accounting cycle. At the end of that short cycle, the accounting cycle resumes its normal monthly cycle. A timeline for this scenario is displayed in Figure 6-1.

Figure 6-1  Short Accounting Cycle

If the short cycle is less than 15 days, a long cycle is created instead. In that case, the extra days are added to the next one-month accounting cycle. This results in a long
cycle with the start date of the old cycle and the end date of the new cycle as seen in Figure 6-2.

Figure 6-2  Long Accounting Cycle

![Diagram of accounting cycle with start and end dates]

Monthly charges are prorated for accounting cycles less than or greater than one month.

A short or long cycle can also occur when a customer creates an account and the billing DOM is different from the day of month when they the account is created. For example, your company might require that all customers be billed on the first day of the month. If a customer creates an account on January 26, by default the first bill is created on March 1. To bill the customer on February 1, you must change the default partial billing cycle to short.

By default, BRM creates a short cycle and a one-month cycle. You can configure BRM to always create short cycles by setting the ShortCycle business parameter using the `pin_bus_params` utility. By default, the ShortCycle parameter is disabled. When this parameter is disabled, if the number of days in the current accounting cycle is less than 15 days, BRM creates a long cycle and adds the remaining days to the next month's accounting cycle. If the number of days in the current accounting cycle is greater than 15 days, BRM creates a short cycle.

To create short cycles:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `config/business_params` object:
   
   ```shell
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, set the value for the following entry to `enabled`:
   
   ```xml
   <ShortCycle>enabled</ShortCycle>
   ```
4. Load the XML file into the BRM database:
   
   ```shell
   pin_bus_params bus_params_billing.xml
   ```
5. Stop and restart the CM.
6. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

You can also configure BRM to round up a long cycle so that the scale for the long cycle equals 2. This enables you to charge your customers for two full cycles.

To round up long billing cycles:
1. Open the CM configuration file (*BRM_home*/sys/cm/pin.conf) in a text editor.

2. Change the value of the following entry to 1:
   - `fm_rate rating_longcycle_roundup_flag` 1

3. Set the value of rounding precision to 0:
   - `fm_rate rating_quantity_rounding_scale` 0

4. Save the file.

5. Stop and restart the CM.

To change how BRM handles short and long cycles, customize the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode source code. See *BRM Opcode Guide*.

### Configuring Timestamp Rounding

By default, BRM rounds time stamps to midnight. You can configure BRM to use exact time stamps by changing the `timestamp_rounding` entry in the CM configuration file (*BRM_home*/sys/cm/pin.conf) from 1 (enabled) to 0 (disabled). You might need to do that to support a custom application.

The following features are affected by time-stamp rounding. Before disabling time-stamp rounding, consider how that change might impact these features:

- **Billing cutoff time.** To use a billing cutoff time other than the default, time-stamp rounding must be enabled. Changing the cutoff time also changes the time to which time stamps are rounded throughout BRM. See "Configuring the Billing Cutoff Time" and "How Timestamp Fields Are Affected by Changing the Cutoff Time."

- **Unit interval used to calculate prorated cycle fees.** If time-stamp rounding is enabled, the unit interval is calculated in days because time stamps are rounded to midnight. If time-stamp rounding is disabled, the unit interval is calculated in seconds, and proration begins from the time indicated by the time stamp. See "Calculating the Unit Interval."

- **Validity period start time of resources granted by cycle events.** Use the CM time-stamp rounding entry to specify whether this time stamp is rounded.

- **Validity period start time of resources granted by purchase events.** Use the CM time-stamp rounding entry and a business parameter to specify whether this time stamp is rounded.

To set the timestamp rounding:

1. Open the CM configuration file (*BRM_home*/sys/cm/pin.conf) in a text editor.

2. Set the `timestamp_rounding` entry to 1.

3. Save and close the file.

### Aligning Account and Cycle Start and End Times

You can align purchase, cycle, and usage start and end times to the accounting cycle, but only if the following are true:
You configure delayed purchase, cycle, or usage start and end times when you set up your product offerings or when you create an account.

The delayed start and end time is a whole number, not a fraction.

The delay is measured in cycles.

The purchase, cycle, or usage start and end times are not modified when a bundle is purchased.

To align the purchase, cycle, and usage start and end times with the accounting cycle:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, set the following entry to `enabled`:
   ```xml
   <CycleDelayAlign>enabled</CycleDelayAlign>
   ```
   **Note:** If the entry is set to `disabled` or not set, the start and end times are not aligned.
4. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
5. Stop and restart the CM.
6. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

The delayed purchase, cycle, or usage start time is set to the accounting cycle start date.

For example, if you create a customer account on May 5 and the accounting cycle is monthly, the billing DOM is set to the 5th of each month by default. If you configured the cycle start delay for 1 cycle, the customer purchases a bundle on May 20, and the accounting cycle is short, the charges begin on June 5. If the accounting cycle is long, the charges begin on July 5.

**Figure 6-3** shows the cycle start time for the above example:

**Figure 6-3  Aligning Account and Cycle Start and End Times**
Defining When Billing-Time Discounts Are Applied

BRM performs accounting operations, such as applying cycle fees, rollover, and billing-time discounts, at the end of every accounting cycle. You can configure BRM to apply billing-time discounts at the end of the billing cycle instead of the accounting cycle. You might do this, for example, when the billing cycle spans multiple accounting cycles and a billing-time discount is based on the aggregated usage for the billing cycle.

**Note:**
This configuration applies only to regular billing, not to bill-now and on-purchase billing.

To enable this feature, run the `pin_bus_params` utility to change the `BillTimeDiscountWhen` business parameter. For information about this utility, see *BRM Developer's Guide*.

To apply billing-time discounts at the end of the billing cycle:

1. Go to *BRM_home*/sys/data/config.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```
3. In the XML file, change `disabled` to `enabled`:
   ```xml
   <BillTimeDiscountWhen>enabled</BillTimeDiscountWhen>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

Including Previous Balances in the Current Amount Due in Open Item Accounting

When you set the accounting type to open item accounting, the total amount due on the bill is reflected in the PIN_FLD_PENDING_RECV field in the `/billinfo` object. It is calculated by using the sum of the current balance and the current nonpaying child bill unit balances: the previous balance of open items is not included. As a result, the customer's bill will not include amounts from previous bills.
You can configure BRM to include the previous total amount due (PIN_FLD_PREVIOUS_TOTAL field) in the total amount due of the current bill unit during open item accounting. This will cause the current bill to reflect the total open charges on an account.

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Change the value of the open_item_actg_include_prev_total entry.
   The values are:
   • 0: The previous total is not added to the pending amount due during open item accounting.
   • 1: The previous balance is added to the pending amount due during open item accounting.
3. Save the file.
4. Stop and restart the CM.

Specifying Which Billing Cycle to Assign to Deferred Purchase Fees

You can assign deferred purchase fees to the previous billing cycle or to the next billing cycle. By default, the purchase fee is assigned to the next billing cycle.

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Change the value of the purchase_fees_backcharge entry.
   The values are:
   • 0: The purchase fees apply to the next cycle.
   • 1: The purchase fees apply to the previous cycle.
3. Save the file.
4. Stop and restart the CM.

Calculating Cycle Fees for Backdating

By default, cycle fees are calculated by using the date that the current accounting cycle ends.

To handle cases where a charge offer's purchase date has been backdated, you can use the CM configuration file calc_cycle_from_cycle_start_t entry to calculate fees based on the charge offer's purchase date. This feature is useful when activating an inactive charge offer.
To set the charge offer cycle start time:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`) in a text editor.
2. Edit the `calc_cycle_from_cycle_start_t` entry:
   - `fm_bill calc_cycle_from_cycle_start_t 1`
     - 0 retains the default BRM behavior to calculate cycle fees (based on the date specified in the PIN_FLD_ACTG_NEXT_T field).
     - 1 sets the charge offer cycle start time to consider the date specified in the PIN_FLD_CYCLE_START_T field for calculating the cycle fees.
3. Save the file.

You do not need to restart the CM to enable this entry.

---

**Note:**

If the cycle start time is not aligned with the billing DOM, the cycle start time is first aligned with the billing DOM before it is used to calculate the cycle charges for the charge offer. However, the cycle start time is aligned only after short and long billing cycle differences are considered.
Use Bill Now to create a bill at any time. This document describes how to configure Bill Now in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- About Bill Now
- Providing Discounts to Closed Accounts When Using Bill Now
- Prorating Cycle Arrears and Cycle Forward Arrears for Bill Now
- Running Bill Now to Create Two Bills during the Delayed Billing Period
- Customizing Bill Now

See also:

- List of Billing Features
- BRM Concepts

About Bill Now

To create a bill at any time, use the Bill Now feature in Billing Care or Customer Center. Bill Now generates a bill that includes all pending items for a bill unit. Bill Now adds the previous total amount to the current total. For example, for an account that has a previous total of $20, and a current total of $10, when Bill Now is run, the account is billed for $30. Bill Now works with both open item and balance forward accounting types.

- When an account has multiple bill units, only one parent bill unit can be processed at a time. If you run Bill Now on a parent bill unit, a bill is created that contains a total of the items from both the parent and any nonpaying bill unit objects.
- If you run Bill Now on a nonpaying bill unit, a bill is created for the parent bill unit that includes only the nonpaying bill unit items.
- You can choose to bill all pending items or select specific items. When a bill is generated for specific items, it does not include the cycle arrears and cycle forward arrears fees.
- Bill Now ignores the cycle_tax_interval entry in the CM configuration file, which specifies whether deferred taxes are calculated separately for the parent and each nonpaying child bill unit or are consolidated into a single item for the parent. When you run Bill Now on a parent bill unit that includes nonpaying child bill units, rolls activities for each nonpaying child bill unit into the parent bill unit and calculates taxes for the parent only. The single tax item for the parent includes taxes from both the parent and the nonpaying child bill units.

See these topics for information about configuring Bill Now:
Providing Discounts to Closed Accounts When Using Bill Now

To apply discounts with Bill Now to closed accounts, you must ensure that BRM does not delete canceled discounts. For information about deleting canceled discounts, see "Specifying to Delete Canceled Discounts" in BRM Managing Customers.

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Set the keep_cancelled_products_or_discounts entry to 1:
   
   ```
   - fm_subscription_policy keep_cancelled_products_or_discounts 1
   ```
   
   If this entry is not present, add it.
3. Save the file.
4. Stop and restart the CM.

Prorating Cycle Arrears and Cycle Forward Arrears for Bill Now

By default, when you use Bill Now, cycle arrears charges and cycle forward arrears charges are not prorated.

You can specify to prorate cycle arrears charges and cycle forward arrears charges for Bill Now by running the pin_bus_params utility to change the ApplyCycleFeeForBillNow business parameter. For information about this utility, see BRM Developer's Guide.

### Note:

When you use Bill Now, you can enable proration for cycle arrears and cycle forward arrears charges; however, cycle forward fees are not prorated.

To prorate cycle forward arrears charges when you use Bill Now:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   
   ```
   pin_bus_params -x BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change disabled to enabled:
   
   ```
   <ApplyCycleFeeForBillNow>enabled</ApplyCycleFeeForBillNow>
   ```
4. Save the file as bus_params_billing.xml.
5. Load the XML file into the BRM database:
   
   ```
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

Running Bill Now to Create Two Bills during the Delayed Billing Period

By default, generating two bills with Bill Now during the delayed billing period is disabled in BRM.

To enable this feature, run the `pin_bus_params` utility to change the `CreateTwoBillNowBillsInDelay` business parameter. For information about this utility, see *BRM Developer's Guide*.

Delayed billing must already be set up before enabling this feature. If you have not already set up delayed billing, see "Setting Up Delayed Billing".

To enable Bill Now during the delayed period:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change `disabled` to `enabled`:
   ```xml
   <CreateTwoBillNowBillsInDelay>enabled</CreateTwoBillNowBillsInDelay>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

Customizing Bill Now

You can customize the following:

- The due date for a bill created with Bill Now is calculated as the billing cycle length minus one day after Bill Now is run. To change the Bill Now due date, customize the `PCM_OP_BILL_POL_CALC_PYMT_DUE_T` policy opcode.
- Bill Now generates a bill that includes all pending items. To change the default behavior, edit the search criteria in the `PCM_OP_BILL_POL_GET_PENDING_ITEMS` policy opcode.
- To apply discounts or folds when using Bill Now, use the `PCM_OP_BILL_MAKE_BILL_NOW` opcode.
- To generate a Bill Now type of bill for a specific service, use `PCM_OP_BILL_CREATE_SPONSORED_ITEMS`, `PCM_OP_BILL_POL_GET_PENDING_ITEMS`, and `PCM_OP_BILL_POL_GET_PENDING_ITEMS`.

See *BRM Opcode Guide*. 
Setting Up Delayed Billing

This document describes how to set up delayed billing in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

• About Delayed Billing
• Configuring Delayed Billing
• Configuring Auto-Triggered Billing for Delayed Billing
• Configuring an Accounting Cycle Delay Period
• Specifying When to Apply Cycle Forward Fees and Cycle Rollovers
• Enforcing Partial Billing in the Billing Delay Interval
• Setting Delayed Cycle Start Dates to the 29th, 30th, or 31st
• Billing Cycle Override for Delayed Billing

See also:

• List of Billing Features
• BRM Concepts

About Delayed Billing

You can set up BRM to delay billing for accounts after the end of a billing cycle. This is called delayed billing. Delayed billing essentially extends a billing cycle by the delay interval. You use delayed billing to bill for events that occur within a billing cycle but are not recorded during that cycle.

For example, if a batch of events does not arrive until after the end of the billing cycle, you delay billing until all events in the batch are recorded in the BRM database. When you use delayed billing, billing for all the accounts in your BRM system is delayed for the same amount of time; you cannot specify multiple billing delay periods.

Figure 8-1 shows that with delayed billing, the billing date occurs after the original billing date, during the next accounting cycle.
When your system is set up to use delayed billing, an account is created with two pending bills, one for the current cycle and one for the next cycle, which are combined in the balance seen in Billing Care or Customer Center. The combined pending bill includes separate items for the previous accounting cycle and for the next accounting cycle. When the bill for the current cycle is finalized at the end of the delay interval, the system makes the bill for the next cycle to be the current bill and creates a new bill for the next cycle.

The BRM system automatically triggers billing inside the delay interval when it detects that a new event has occurred for the next billing cycle. When billing is triggered during the delayed billing period, the bill for the previous cycle is partially processed (partial billing), but the bill is not finalized.

BRM performs partial billing to enable the new event to be rated and applied to the correct billing period. Partial billing ensures that new events impact bill items of the next billing cycle and old events impact bill items of the previous billing cycle. BRM maintains an internal list of bill items for the previous and next bill cycles. The bill for the previous cycle is finalized (final billing) after the delay interval.

During partial billing, BRM does the following:

- Applies deferred cycle forward fees to the next billing cycle.
- Applies cycle arrears fees to the previous billing cycle.
- Applies cycle forward fees.
- Applies cycle rollovers.

During final billing, BRM does the following:

- Applies cycle discounts (billing time discounts).
- Applies cycle folds.
- Applies cycle taxes.
- Calculates a `/bill` object for the previous billing cycle.
- Initializes the next billing cycle.
- Creates a new empty `/bill` object for the next billing cycle with default and pre-created items.

The following example is illustrated in Figure 8-2:

- An account’s billing date is January 1 and the billing delay interval is five days.
- On January 2, a new usage event for the account occurs for the next billing cycle.
- To ensure that the new usage event impacts items in the next billing period (B2), BRM performs partial billing.
- On January 6, final billing is run for the previous billing cycle (B1) by running the billing utility; the status of all the bill items for the previous bill is changed to open so that they stop accumulating charges.

**Figure 8-2  Partial and Final Billing Timeline**

When you use delayed billing, auto-triggered billing is disabled for all but the delay interval and only the last two days of each bill unit’s accounting cycle.

When a bill-triggering event occurs during the delayed billing period, BRM auto-triggers partial billing and, if final billing has not occurred before the last two days of the next billing cycle, BRM auto-triggers final billing. This ensures previous billing is run before the next billing run occurs.

For example:

- An event for the next billing cycle is recorded after the billing delay interval on January 7.
- The BRM system detects that the delay period is over but final billing for the previous billing cycle has not occurred yet.
If auto-triggered billing is **disabled** (the default when delayed billing is configured), BRM does not run final billing on January 7. In this case, the delay interval is virtually extended until final billing is performed by running the billing utility or auto-triggered during the last two days of the next accounting cycle.

If auto-triggered billing is **enabled**, BRM auto-triggers final billing for the previous billing cycle on January 7.

**Note:**

- You can change auto-triggered billing to be **always enabled** when delayed billing is used by setting the `auto_triggering_limit` parameter to 0.
- You can also change the number of days auto-triggered billing is **enabled** at the **end** of each accounting cycle.

### Delayed Billing and Rollovers

If you enable delayed billing, delayed events can borrow rollover from the current cycle even if events from the current cycle have consumed the rollover. Unless you set up rerating and rollover correction, current cycle events can remain rated as free even if their rollover has been consumed by delayed events. For more information, see the discussion of rerating in BRM Implementing Charging.

### Changing the Billing DOM When Delayed Billing is Enabled

If you change a customer’s billing DOM during the delayed billing period, BRM defers the DOM change until after the delay interval ends. This ensures that the change to the billing DOM occurs in the future billing cycle. For example, assume that a customer’s billing DOM is 1 and the delay interval is 5, making the billing date for the first three months of the year to be January 1, February 1, and March 1. If on January 3 the customer changes the billing DOM to 15, BRM defers making the change until January 6 (January 1 plus 5 days). This changes the billing date for the first three months to January 1, February 1, and March 15.

**Note:**

To have the billing DOM change deferred until after the delay interval, auto-triggered billing must be disabled; otherwise, the billing DOM change occurs immediately. For example, changing the billing DOM to 15 on January 3 would change the billing date to February 15.

### How BRM Assigns Delayed Events to Items

In BRM, every event object is associated with a bill item. When delayed events are rated by ECE, some of the events need to be assigned to the next accounting cycle. BRM pre-creates items in the following cases:
• When an account is created.
• When you run billing.
• When a CSR uses Bill Now.
• At the end of the accounting cycle when you use delayed billing.

To choose the correct bill item, ECE can do one of the following:
• Assign the event to the current open bill item.
• Assign the event to the next open bill item.

To decide which item to apply the bill to, ECE takes into account the following dates:
• The date when the call occurred.
• The current system date.
• The date when the current accounting cycle ends. This is called the next accounting cycle date.
• The number of days after the current accounting cycle ends when delayed billing runs. This number is called the delayed billing offset.

To assign the event to an item:
• If the event date falls before the next accounting date, the event is assigned to the current item.
• If the event date falls after the next accounting date, the event is assigned to the next item. This can happen because the event might occur after the close of the accounting cycle but before the delayed billing offset date.

Figure 8-3 shows how events are assigned:

Figure 8-3  Event Assignment

Note:
The customer billing date is not relevant when choosing which item to use for the event. There might be multiple accounting cycles in one billing cycle. New items are created for each accounting cycle.

If ECE needs to assign an event to the current item, but billing for that item has already occurred, ECE includes the event and assigns it to the current item. For
example, if the event is rated on May 20 and loaded after the account cycle ends, it is still included in the current item as shown in Figure 8-4.

Figure 8-4  Current Event Assignment after Billing

Configuring Delayed Billing

By default, delayed billing is enabled. You can configure delayed billing by setting the ConfigBillingDelay parameter using the pin_bus_params utility. For information on this utility, see BRM Developer's Guide.

To configure delayed billing:

1. Go to $BRM_home/sys/data/config$.
2. Create an XML file from the /config/business_params object:
   
   pin_bus_params -r BusParamsBilling bus_params_billing.xml

3. In the XML file, enter a value greater than 0:

   <ConfigBillingDelay>D[:H]</ConfigBillingDelay>

   where $D$ is the number of days and $H$ is the number of hours. Leading zeros are allowed when specifying the delay interval.

   For example:
   - $0:12$ sets billing delay interval to 12 hours.
   - 2 sets billing delay interval to 2 days
   - $1:3$ sets billing delay interval to 1 day and 3 hours.
   - $01:03$ also sets billing delay interval to 1 day and 3 hours.
   - 0 sets billing delay interval to zero.
Note:

- To use delayed billing, set `ConfigBillingDelay` to a value ranging from 1 to 27. The length of the billing delay interval must be shorter than one accounting cycle.
- If you do not want to use delayed billing, set `ConfigBillingDelay` to 0.
- To disable delayed billing, set this parameter to -1. This is supported for backward compatibility.

Caution:

You can change the value of `ConfigBillingDelay` at any time. However, after you begin rating events in a production database, do not reset this parameter. Doing so might cause database errors.

4. Load the XML file into the BRM database:

   ```shell
   pin_bus_params bus_params_billing.xml
   ``

5. Stop and restart the CM.

6. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

Configuring Auto-Triggered Billing for Delayed Billing

When a systemwide billing delay is set in BRM, by default auto-triggered billing is disabled for all but the delay period and only the last two days of each bill unit's accounting cycle.

You can specify auto-triggered billing to be enabled for more than two days at the end of each accounting cycle, or you can specify that it is always enabled when delayed billing is used.

To change this feature, run the `pin_bus_params` utility to change the `AutoTriggeringLimit` business parameter. For information about this utility, see `BRM Developer's Guide`.

Note:

This is a systemwide setting; it applies to the accounting cycle of every bill unit in your BRM system.

Configure the auto-triggered billing period for delayed billing as follows:

1. Go to `BRM_home/sys/data/config`.

2. Create an XML file from the `/config/business_params` object:

   ```shell
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change the number of days for which auto-triggered billing is enabled at the end of each accounting cycle. For example, if you change the value to 10, auto-triggered billing is enabled for the last 10 days of every accounting cycle and in the billing delay interval.

   To always enable auto-triggered billing when delayed billing is used, change 2 to 0. For example, if billing delay interval is five days and AutoTriggeringLimit is 0, auto-triggered billing is enabled all the time.

   \(<\text{AutoTriggeringLimit}>10</\text{AutoTriggeringLimit}>\)

4. Save the file as bus_params_billing.xml.

5. Load the XML file into the BRM database:

   \(\text{pin_bus_params_bus_params_billing.xml}\)

6. Stop and restart the CM.

7. (Multischema systems only) Run the pin_multidb script with the -R CONFIG parameter. For more information, see BRM System Administrator's Guide.

---

**Configuring an Accounting Cycle Delay Period**

In offline charging, the following scenario can occur:

1. The event is rated by Offline Mediation Controller.

2. Before the event is loaded by Rated Event Loader (RE Loader), billing is run.

3. RE Loader must find the item that the event applies to and apply the balance impact.

Delayed billing assigns delayed events to items of the billing cycle in which they occurred. However, in the scenario described above, you might need to assign the delayed events to the accounting cycle in which they occurred. To assign delayed events to items of the accounting cycle in which they occurred, you configure an accounting cycle delay period. You can do this to enable accurate general ledger reporting.

When a billing cycle spans multiple accounting cycles, the items for those accounting cycles are not closed until billing is run. If you run a general ledger (G/L) report at the end of an accounting cycle for which billing has not yet been run, the status of the revenue in the G/L can change if additional events are rated and loaded for that cycle before the accounts are billed.

If you require that G/L data not change after the G/L report is run, you can configure an accounting cycle delay period after which events are no longer assigned to items of that accounting cycle, even if those items are not closed. You then run G/L reports after the accounting cycle delay period has ended. This ensures that the revenue reported in the G/L is accurately represented and that the state of the revenue (earned, unearned, billed, and unbilled) does not change after the G/L report is run.

When you configure an accounting cycle delay period, BRM assigns delayed events to items based on when the accounting cycle delay period ends. BRM assigns events to items when RE Loader loads the events into the database:

- When delayed events (events that occurred in the previous cycle) are loaded after the accounting day of month (DOM), but before the end of the accounting cycle delay period, those events are posted to the item for which the DOM has just passed as shown in Figure 8-5:
Figure 8-5  Delayed Events Arriving During Cycle Delay Period

- When the billing cycle has ended and delayed events are loaded after the end of the accounting cycle delay period, but before delayed billing is run, those events are posted to the item for the next (current) accounting cycle, even though the previous cycle has not been billed and its items are still pending. This is shown in Figure 8-6:

Figure 8-6  Delayed Events Arriving After Cycle Delay Period

- After the account is billed, items for the billed cycle are closed so delayed events are posted to the item for the following (the current) cycle.
Note:

If the accounting cycle delay period is longer than the delayed billing period, the accounting cycle delay period is ignored after billing is run. After billing is run, if any remaining events that occurred in the previous cycle are rated and loaded in the current cycle, they are assigned to the current cycle's item.

You specify the accounting cycle delay period by running the `pin_bus_params` utility to change the `AcctCycleDelayPeriod` business parameter. For information on this utility, see `BRM Developer's Guide`.

To configure an accounting cycle delay period:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `config/business_params` object:
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change `-1` to the number of days in the accounting cycle delay period. The number of days must be a positive value. (A value of `-1` indicates that there is no accounting cycle delay period.):
   ```
   <AcctCycleDelayPeriod>3</AcctCycleDelayPeriod>
   ```

Note:

Do not set the accounting cycle delay period to be longer than the delayed billing period.

4. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing.xml
   ```
5. Stop and restart the CM.
6. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

Specifying When to Apply Cycle Forward Fees and Cycle Rollovers

Cycle forward fees and cycle rollovers are normally applied at the beginning of the accounting cycle to charge for services provided during that cycle and to roll over unused balances for use in subsequent cycles. However, when your system is set up for delayed billing, cycle forward fees and cycle rollovers are applied during partial billing by default.

When you used delayed billing, the BRM system provides the flexibility to specify when to charge cycle forward fees and when to roll over balances. You can specify to charge cycle forward fees and roll over balances during either partial billing or final billing by setting the `delay_cycle_fees` entry in the CM configuration file (`pin.conf`).
To specify when to apply cycle forward fees and cycle rollovers:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`) in a text editor.
2. Add the `- fm_bill delay_cycle_fees` entry and set it to 0 or 1.
   - 0 (the default) applies cycle forward fees and cycle rollover during partial billing.
   - 1 applies cycle forward fees and cycle rollover during final billing.

   **Note:**
   You can change the setting for `delay_cycle_fees` either before partial billing or after final billing. Do not change this setting between partial billing and final billing.

3. Save the file.
4. Stop and restart the CM.

---

### Enforcing Partial Billing in the Billing Delay Interval

Partial billing is run only when your BRM system is set up for delayed billing. The BRM system automatically triggers partial billing by default when it detects that a new event has occurred for the next billing cycle inside the billing delay interval.

When there are no new events in the delay interval and partial billing has not occurred, you can force the BRM system to run partial billing when the billing utility is run in the delay interval. Later, if a new event occurs in the delay interval, the new event is processed immediately, without waiting for the partial billing run to complete.

To force partial billing:

1. Open the billing utility configuration file (`BRM_home/apps/pin_billd/pin.conf`) in a text editor.
2. Set the `- pin_bill_accts enforce_billing` entry to 0 or 1.
   - 0 does not enforce partial billing.
   - 1 enforces partial billing. This is the default.
Note:
The `enforce_billing` entry is used by the BRM system to enforce partial billing only if the `config_billing_delay` entry is specified and set to a number greater than zero.

3. Save the file.
4. Run the billing utility.

Setting Delayed Cycle Start Dates to the 29th, 30th, or 31st

By default, when you delay a customer's cycle fees by one month: for example, to provide a promotional month of free service: BRM sets the delayed cycle start date to any date from the 1st through the 28th of the month. Therefore, any delayed cycle fees due on the 29th, 30th, or 31st of the month are advanced to the first day of the following month. For example, if you delay cycle fees by one month for a bundle purchased on October 29, BRM sets the delayed cycle start date to December 1.

To configure BRM to enable delayed cycle start times on the 29th, 30th, or 31st of a month:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`) in a text editor.
2. Change the `fm_bill cycle_delay_use_special_days` entry:
   - To set the delayed cycle start date to the 1st of the following month for all bundles purchased on the 29th, 30th, or 31st, enter 0. This is the default setting.
   - To enable BRM to assign delayed cycle start dates to the 29th, 30th, or 31st of the month, enter 1.
3. Save the file.
4. Stop and restart the CM.

Billing Cycle Override for Delayed Billing

You can override the billing cycle for events that occur during the delayed billing interval. By default, events recorded during the delayed billing interval are billed in the previous billing cycle when the event time precedes the previous billing cycle end date. Otherwise, the event is billed in the current billing cycle. You can configure BRM to specify whether such events are billed in the previous or current billing cycle. BRM enables you to specify a configurable billing cycle interval. You can then choose which events recorded during this interval are to be billed in the previous or current billing cycle. Events that are not recorded during this interval are billed as usual, using the default delayed billing implementation.

To configure the billing cycle for events that occur during the delayed billing interval:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`) in a text editor.
2. Set the `config_billing_cycle` entry to the length of the configurable billing cycle interval.
3. For example, the following sets the interval to 5 days:
config_billing_cycle 5

Note:
The `config_billing_cycle` value must be greater than zero and less than or equal to the `config_billing_delay` (delayed billing interval value). Otherwise, BRM reports an error and terminates the CM.

Note:
Setting the configurable billing cycle to be the same as the delayed billing interval will affect system performance because each event occurring within the delayed billing interval is passed to the PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE policy opcode for additional processing.

4. Modify the PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE policy opcode:
   - To bill the event in the previous cycle, set the output flist field FLAGS to `BILL_IN_PREVIOUS_CYCLE`.
   - To bill the event in the current cycle, set the FLAGS field to `BILL_IN_CURRENT_CYCLE`.

See the documentation for PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE in `BRM Opcode Guide`. 
Configuring 31-Day Billing

This document describes how to enable Oracle Communications Billing and Revenue Management (BRM) to use all 31 days in a month as the billing day of month.

See also:
- List of Billing Features
- BRM Concepts

About Using 31-Day Billing

By default, you can set the billing day of month (DOM) to any day between 1 and 28. If your customer signs up on the 29th, 30th, or 31st, the billing DOM is set to the 1st. This can result in a large number of customers being billed on the 1st.

You can change this setting to support billing on all days of the month. For example, if you create a customer account on the 29th, the billing DOM is set to the 29th instead of the 1st.

If your customers' billing DOM is the 29th, 30th, or 31st, for the months that do not have these days, you can configure whether billing should be run on the last day for the same month (set to back option) or the first day of the next month (set to forward option). By default, the billing DOM is set forward to the 1st of the next month.

For example, when the DOM is set to 31, because April has 30 days:
- The set to forward option sets the billing date to May 1.
- The set to back option sets the billing date to April 30.

Enabling the 31-Day Billing Feature

To use 31-day billing, you must either modify the init_objects.source file before loading it into the database or set the 31DayBilling business parameter in the billing instance by running the pin_bus_params utility. For information on this utility, see BRM Developer's Guide. After BRM is configured to allow 31-day billing, you can specify to use the set to back option or the set to forward option.

Switching to 31-Day Billing during BRM Installation

Before loading init_objects.source, change the value of the PIN_FLD_MAXIMUM field from 28 to 31 in the /config/fld_validate object that has the Actg_cycle value in the PIN_FLD_NAME field as follows:

```
# /config/fld_validate - Actg_cycle validation
<PCM_OP $PIN_OPNAME=$PIN_CONF_INIT_OPNAME; $PIN_OPFLAGS=$PIN_CONF_INIT_OPFLAGS>
0 PIN_FLD_POID POID [0] $PIN_CONF_DB_NO /config/fld_validate 606 0
0 PIN_FLD_DESCR STR [0] "Field Validation"
0 PIN_FLD_HOSTNAME STR [0] "-"
```
Switching to 31-Day Billing after You Install BRM

To switch to 31-day billing after you have installed BRM, set the 31DayBilling business parameter in the billing instance by running the pin_bus_params utility.

To enable the 31-day billing feature:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, set the following entry to enabled:
   
   ```xml
   <31DayBilling>enabled</31DayBilling>
   ```
4. Load the XML file into the BRM database:
   
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
5. Stop and restart the CM.
6. (Multischema systems only) Run the pin_multidb script with the -R CONFIG parameter. For more information, see BRM System Administrator's Guide.

Setting the Forward and Back Billing Options

To enable 31-day billing, run the pin_bus_params utility to change the MoveDayForward business parameter. For information on this utility, see BRM Developer's Guide.

To configure the billing DOM to be last day of the month or 1st of the next month:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change disabled to enabled:
   
   ```xml
   <MoveDayForward>firstDay</MoveDayForward>
   ```
4. Load the XML file into the BRM database:
5. Stop and restart the CM.

6. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*. 

`pin_bus_paramsbilling.xml`
10

Configuring the Billing Cutoff Time

This document describes how to configure the billing cutoff time in Oracle Communications Billing and Revenue Management (BRM).

See also:

- List of Billing Features
- BRM Concepts

About Configuring the BRM Cutoff Time

By default, BRM defines the business day as starting at 12:00:00 a.m. and ending at 11:59:59 p.m. For example, if you run billing at any time on December 5, billing is performed for all activity that occurred until 11:59:59 p.m. on December 4 for the accounts to be billed.

You can change the cutoff time to start your billing activity at any time of the day. For example, if you set the cutoff time to 10 a.m., activity for events that occurred before 10 a.m. are billed.

Figure 10-1 shows how billing works for different cutoff times:
Changing the cutoff time does not just change how billing works; it changes how all activities in BRM work, including accounting and billing cycles, usage rating, cycle fees, proration, general ledger posting, and searches. The cutoff time is used for all accounts.

How Billing and Invoicing Are Affected by Changing the Cutoff Time

- The start and end dates for accounting and billing cycles are based on the cutoff time. For example, if the cutoff time is 10:00 a.m., a customer who creates an account at 9:00 a.m. on December 5 has a billing date of December 4.
- The following utilities run by the `pin_bill_day` script use the cutoff time to calculate the billing periods:
  - `pin_deferred_act`
  - `pin_bill_accts`
  - `pin_collect`
  - `pin_refund`
  - `pin_inv_accts`
  - `pin_deposit`
  - `pin_cycle_fees`
• When searching for accounts, the `pin_inv_accts`, `pin_inv_send`, and `pin_inv_export` utilities use the cutoff time to calculate the start and end times for flagging accounts to be invoiced.

### How Rating Is Affected by Changing the Cutoff Time

• When you define start and end times for any pricing component (for example, the start and end times for a discount), BRM uses the cutoff time. For example, if you specify that a discount is valid until December 5 and the cutoff time is 10:00 a.m., the discount is valid until 10:00 a.m. on December 5.

• You can set up special charges for events that occur on certain days. BRM uses the cutoff time to determine which day an event is assigned to.

### How General Ledger (G/L) Is Affected by Changing the Cutoff Time

When searching for events for collecting G/L information and generating G/L reports, the `pin_ledger_report` utility uses the cutoff time to calculate the start and end times for the G/L report.

### How Timestamp Fields Are Affected by Changing the Cutoff Time

Many BRM features use timestamps to determine how to perform activities. Timestamps are usually rounded to midnight. If you change the cutoff time, the timestamp is rounded to the cutoff time instead.

> **Note:**

The cutoff time is also considered while setting the timestamp values for product offering start and end dates.

These fields affect the accounting cycle dates:

• `PIN_FLD_ACTG_LAST_T`
• `PIN_FLD_ACTG_NEXT_T`
• `PIN_FLD_ACTG_FUTURE_T`

These fields affect rating and proration:

• `PIN_FLD_PURCHASE_START_T`
• `PIN_FLD_PURCHASE_END_T`
• `PIN_FLD_USAGE_START_T`
• `PIN_FLD_USAGE_END_T`
• `PIN_FLD_CYCLE_START_T`
• `PIN_FLD_CYCLE_END_T`

These fields affect billing cycle dates:

• `PIN_FLD_LAST_BILL_T`
• `PIN_FLD_NEXT_BILL_T`
Configuring the Billing Cutoff Time

Note:
After you set the cutoff time, you cannot change it in a production system.

To configure the billing cutoff time:

1. Set the timestamp_rounding entry in the CM pin.conf file.
2. Set the cutoff time by running the pin_bus_params utility to change the BillingCycleOffset business parameter. For information on this utility, see BRM Developer's Guide.

To set the timestamp rounding:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf) in a text editor.
2. Set the timestamp_rounding entry to 1.
3. Save and close the file.

To set the cutoff time:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change 0 to the desired cutoff time. For example, to set the cutoff time to 10:00 a.m., change 0 to 10. The default for this field is 0, which is equivalent to 12:00 a.m.
   ```
   <BillingCycleOffset>10</BillingCycleOffset>
   ```
4. Save the file as bus_params_billing.xml.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the pin_multidb script with the -R CONFIG parameter. For more information, see BRM System Administrator's Guide.
11 Configuring Billing for Groups and Hierarchies

This document describes configuration options for groups and hierarchies in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

• Setting Up Billing for Charge and Discount Sharing Groups
• Skipping Validation of Billing for Nonpaying Child Bill Units

See also:

• List of Billing Features
• BRM Concepts

Setting Up Billing for Charge and Discount Sharing Groups

By default, when billing is run, bill units are billed in this order:

1. All nonpaying child bill units in all accounts.
2. All remaining bill units in all accounts.

If you have discount or charge sharing groups in your BRM system, you must reconfigure your system to bill accounts in this order:

1. All nonpaying child bill units in all accounts.
2. All remaining discount group member bill units in all member accounts.
3. All remaining charge sharing group member bill units in all member accounts.
4. All remaining bill units in all accounts.

This ensures that billing is run for all member accounts before it is run for any discount or charge sharing group owner account.

Note:

If you do not reconfigure your system, discount and charge sharing group owner accounts might be billed before some of their member accounts. When this occurs, the members’ sponsored charges are not included in the owner’s bill for the current billing cycle. Instead, they are added to the owner’s bill for the next billing cycle.

To enable this feature, run the pin_bus_params utility to change the BillingFlowSponsorship business parameter. For information about this utility, see BRM Developer’s Guide.
The following setups ensure that two-level discount and charge sharing relationships are correctly billed. For sharing relationships that exceed two levels, the billing sequence might be incorrect, resulting in incorrect billing.

For example, suppose account A owns a sharing group to which account B belongs, and account B owns a sharing group to which account C belongs. Owner account A and member account B are billed in the correct order: member before owner. Owner account B and member account C, however, might not be billed in the correct order. This happens because B is an owner of account C's group and a member of account A's group. As a member, it is billed at the same time all other member accounts are billed and thus might be billed before account C.

To set up billing for charge sharing groups:

1. Go to `BRM_home/.sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```
   pin_bus_params --r BusParamsBillingFlow bus_params_billing_flow.xml
   ```
3. In the XML file, change `undefined` to one of the following:
   - `sponsorsFirst` to bill group owner accounts before member accounts.
   - `sponsoreesFirst` to bill member accounts before group owner accounts.

   If the billing order of the owner and member accounts does not matter, keep the original setting of `undefined`.

   ```xml
   <BillingFlowSponsorship>undefined</BillingFlowSponsorship>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing_flow.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

To set up billing for discount sharing groups:

1. Go to `BRM_home/.sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```
   pin_bus_params --r BusParamsBillingFlow bus_params_billing_flow.xml
   ```
3. In the XML file, change `undefined` to one of the following:
   - `discountParentsFirst` to bill group owner accounts before member accounts.
   - `memberDiscountFirst` to bill member accounts before owner accounts.

   If the billing order of owner and member accounts does not matter, keep the original setting of `undefined`.

   ```xml
   <BillingFlowDiscount>undefined</BillingFlowDiscount>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing_flow.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see BRM System Administrator's Guide.

**Skipping Validation of Billing for Nonpaying Child Bill Units**

If you use bill unit hierarchies, BRM validates that all nonpaying bill units have been billed successfully before billing the paying parent bill unit. When billing fails for a nonpaying child bill unit, the parent bill unit is not billed. In rare instances, however, when billing for the nonpaying child bill unit continuously fails and you want to proceed with billing the parent bill unit, you can skip validation of billing for the nonpaying child bill unit.

To enable this feature, run the `pin_bus_params` utility to change the `SkipCheckForSubordinatesBilled` business parameter. For information about this utility, see BRM Developer's Guide.

To skip validation of billing for nonpaying child bill units:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change `disabled` to `enabled`:
   ```xml
   <SkipCheckForSubordinatesBilled>enabled</SkipCheckForSubordinatesBilled>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see BRM System Administrator's Guide.
Load Balancing Billing Runs

This chapter describes how to load balance billing operations by running billing based on the customer's accounting day of month in Oracle Communications Billing and Revenue Management (BRM).

Implementing this feature requires using opcodes to create a custom application.

See also:

- List of Billing Features
- BRM Concepts

About Managing Billing Cycles

Accounting cycles and billing cycles begin on a bill unit's billing day of month (DOM). By default, the DOM on which an account is created automatically becomes the billing DOM for all the account's bill units. This can result in an uneven distribution of a system's billing operations across each month.

To load-balance billing operations, you can use billing segments to assign accounts to billing DOMs. A billing segment is a user-defined category, such as wholesale, retail, and senior citizen. To implement bill cycle management, you create billing segments in your BRM system and then assign bill units to them. The billing segment determines which DOMs are available to the bill units.

A billing segment can be associated with any number of DOMs. For example, billing segment A might be associated with DOMs 1 through 31 and billing segment B might be associated with DOMs 1, 15, and 31. For each DOM with which it is associated, a billing segment contains the following information:

- A status (open, closed, or restricted) that determines whether the DOM can be assigned to the bill units that belong to the segment.
- The maximum number of accounts that can be associated with the DOM–billing segment pair.
- The maximum number of services that can be associated with the DOM–billing segment pair.
- The following data, which you must use third-party data warehousing software to gather:
  - The number of accounts currently associated with the DOM–billing segment pair.
  - The number of services currently associated with the DOM–billing segment pair.
  - The total amount of time that it took to process the bills associated with the DOM–billing segment pair during the previous billing run.
Note:

To accommodate the frequent updates that such data may require, BRM automatically refreshes cached billing segment data once a day (see "Updating Billing Segments").

BRM uses billing segment information with new bill cycle management functions in the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode to determine the accounting DOMs to which a bill unit can be assigned. For more information, see "Implementing Bill Cycle Management" and BRM Opcode Guide.

The status of the accounting DOMs in each billing segment determines whether and how a particular DOM can be assigned to the bill units associated with the billing segment as shown in Table 12-1:

Table 12-1   DOM Status Assignments

<table>
<thead>
<tr>
<th>DOM Status</th>
<th>Manually Assignable?*</th>
<th>Automatically Assignable?**</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>restricted</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>closed</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

* See "Manually Assigning a Billing DOM".
** See "Automatically Assigning a Billing DOM".

You specify the status of accounting DOMs in the pin_billing_segment.xml file. For more information, see "Setting Up Billing Segments".

Note:

- Accounting DOMs not explicitly associated with a billing segment are considered closed for the segment. For example, if billing segment C is associated only with DOMs 1 and 2, DOMs 3 through 31 are closed for that segment and cannot be assigned to bill units that belong to the segment.
- If a billing segment is not associated with any accounting DOMs, all DOMs are open for the segment. In such cases, the default process, not the bill cycle management process, is used to assign an accounting DOM to the bill units that belong to the segment.

Implementing Bill Cycle Management

To implement bill cycle management:

1. Set up billing segments in your system. See "Setting Up Billing Segments".
2. Perform the following tasks programmatically or through a custom user interface:
• Associate bill units with billing segments at account creation and account maintenance time. See "Associating Bill Units with Billing Segments ".

• (Optional) Select a billing DOM for bill units. See "Assigning Accounting Days of Month to Bill Units in Billing Segments".

3. (Optional) Customize the PCM_OP_CUST_POL_PREP_BILLINFO opcode to select the DOM most likely to have the lightest billing load of all available DOMs. See the chapter about billing in BRM Opcode Guide.

Setting Up Billing Segments

To set up billing segments in your system, edit the billing segment configuration file BRM_home/sys/data/config/pin_billing_segment.xml, and load its contents into the /config/billing_segment object in the BRM database.

Note:
The utility that loads billing segments into the database overwrites existing billing segments. When updating billing segments, you cannot load new segments only. You must load the complete set of billing segments each time you run the utility.

1. Open the pin_billing_segment.xml file in an XML editor or a text editor.
2. Enter the appropriate information into the file. See "Editing the Billing Segment Configuration File".
3. Save the file.
4. Use the following command to run the "load_pin_billing_segment" utility from the directory in which the pin_billing_segment.xml file is located:
   
   ```
   load_pin_billing_segment pin_billing_segment.xml
   ```
   
   If you do not run the utility from the directory in which pin_billing_segment.xml is located, include the complete path to the file. For example:
   
   ```
   load_pin_billing_segment BRM_home/sys/data/config/pin_billing_segment.xml
   ```
   
   For more information, see "load_pin_billing_segment".

5. Activate the feature that automatically refreshes billing segment data in the Connection Manager (CM) cache. See "Automatically Refreshing Billing Segment Data".

6. Stop and restart the CM.
Note:

- If the data does not need to be added to the CM cache until the next time the cache is automatically refreshed, you do not have to do this.
- If an error occurs at CM startup, the space allocated to billing segment data in the CM cache may not be sufficient to accommodate the size of your billing segment data. See "Increasing the Size of the CM Cache for Billing Segment Data".

7. To verify that the billing segment information was loaded, display the \config\billing_segment object by using one of the following features:

- Object Browser
- `robj` command with the `testnap` utility

For general instructions on using `testnap`, see "Using testnap" in \BRM\Developer's Guide. For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in \BRM\Developer's Guide.

Editing the Billing Segment Configuration File

You configure all the billing segments in your BRM system in the \BRM_home\sys\data\config\pin_billing_segment.xml file.

To edit this configuration file, open it in an XML editor or a text editor and then perform these tasks:

- To add billing segments to the file, see "Defining Billing Segments".
- To associate the billing segments with billing DOMs. See "Associating Billing Segments With Accounting Days of Month".

Defining Billing Segments

In the billing segment configuration file, billing segments are defined as `BillingSegment` child elements of the `BillingSegments` parent element.

A `BillingSegment` child element consists of a billing segment ID and a description (string):

```xml
<BillingSegmentConfiguration>
  <BillingSegments>
    <BillingSegment ID="int ">string</BillingSegment>
  </BillingSegments>
</BillingSegmentConfiguration>
```

To create a billing segment, add a `BillingSegment` child element to the `BillingSegments` parent element. In the child element, specify values for the items listed in Table 12-2:
### Table 12-2 XML Elements in Billing Segment

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A number that identifies the billing segment in the BRM database. When a bill unit is linked to a billing segment, this number is put in the PIN_FLD_BILLING_SEGMENT field of the /billinfo object. An array of all the billing segment IDs is stored in the /config/billing_segment object.</td>
<td>To use bill cycle management to assign billing DOMs to bill units in the segment, specify any integer greater than or equal to 101. <strong>Note:</strong> ID 0 triggers BRM to use the non-bill-cycle-management assignment process.</td>
</tr>
<tr>
<td>string</td>
<td>A character string that describes the type of accounts in the billing segment (for example, wholesale or retail).</td>
<td>Minimum length is 1 character. Maximum length is 1023 characters. <strong>Note:</strong> This string is mapped to the PIN_FLD_DESCR field in the /config/billing_segment object, which can be used to populate a list of billing segments in a user interface (UI). When creating the string, take any UI length restrictions into account.</td>
</tr>
</tbody>
</table>

### Associating Billing Segments With Accounting Days of Month

In the billing segment configuration file, the **DomAssignments** parent element contains **DomAssignment** child elements, each of which associates a billing segment with a DOM:

```xml
<BillingSegmentConfiguration>
  <DomAssignments>
    <DomAssignment billingSegmentRef="int" status=" status " dom=" ---gDay " maxAccounts="int" maxServices="int"> 
      <NumAccounts>int</NumAccounts>
      <NumServices>int</NumServices>
      <TotalProcessTime>duration</TotalProcessTime>
    </DomAssignment>
    <DomAssignment>
      </DomAssignments>
  </BillingSegmentConfiguration>
```

To create a billing segment–DOM association, add a **DomAssignment** child element to the **DomAssignments** parent element. In the child element, specify values for the items listed in **Table 12-3**.

---

**Table 12-3** XML Elements in Billing Segment

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A number that identifies the billing segment in the BRM database. When a bill unit is linked to a billing segment, this number is put in the PIN_FLD_BILLING_SEGMENT field of the /billinfo object. An array of all the billing segment IDs is stored in the /config/billing_segment object.</td>
<td>To use bill cycle management to assign billing DOMs to bill units in the segment, specify any integer greater than or equal to 101. <strong>Note:</strong> ID 0 triggers BRM to use the non-bill-cycle-management assignment process.</td>
</tr>
<tr>
<td>string</td>
<td>A character string that describes the type of accounts in the billing segment (for example, wholesale or retail).</td>
<td>Minimum length is 1 character. Maximum length is 1023 characters. <strong>Note:</strong> This string is mapped to the PIN_FLD_DESCR field in the /config/billing_segment object, which can be used to populate a list of billing segments in a user interface (UI). When creating the string, take any UI length restrictions into account.</td>
</tr>
</tbody>
</table>
**Note:**

- You must add one `DomAssignment` child element for every billing segment–DOM association. For example, to associate billing segment 101 with DOMs 1 through 31, you must add 31 `DomAssignment` child elements. The `billingSegmentRef` value of each child element will be the same (101), but each child element will have a different `dom` value.
- If a billing segment is not explicitly associated with any accounting DOMs, all DOMs are open for the segment.
- An array of all the billing segment and DOM pairs in a BRM system (PIN_FLD_MAP) is stored in the `/config/billing_segment` object.

### Table 12-3  DOM Assignments

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>billingSegmentRef</code></td>
<td>The billing segment ID value in the ID attribute of a <code>BillingSegment</code> child element in the <code>BillingSegments</code> parent element.</td>
<td>See the <code>billingSegmentId</code> entry in the table in &quot;Defining Billing Segments&quot;.</td>
</tr>
<tr>
<td><code>status</code></td>
<td>The status of a DOM for the billing segment.</td>
<td>One of the following: • open • restricted • closed <strong>Important:</strong> These values are case sensitive.</td>
</tr>
<tr>
<td><code>dom</code></td>
<td>A billing DOM.</td>
<td>Any two-digit value from 01 through 31.</td>
</tr>
<tr>
<td>(Optional) <code>maxAccounts</code></td>
<td>Maximum number of accounts that can be associated with the DOM–billing segment pair.</td>
<td>Any non-negative integer.</td>
</tr>
<tr>
<td>(Optional) <code>maxServices</code></td>
<td>Maximum number of services that can be associated with the DOM–billing segment pair.</td>
<td>Any non-negative integer.</td>
</tr>
<tr>
<td>(Optional) <code>NumAccounts</code></td>
<td>Number of accounts currently associated with the DOM–billing segment pair.</td>
<td>Any non-negative integer. <strong>Note:</strong> This information is generated by third-party data warehousing software.</td>
</tr>
<tr>
<td>(Optional) <code>NumServices</code></td>
<td>Number of services currently associated with the DOM–billing segment pair.</td>
<td>Any non-negative integer. <strong>Note:</strong> This information is generated by third-party data warehousing software.</td>
</tr>
</tbody>
</table>
Table 12-3  (Cont.) DOM Assignments

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Optional) TotalProcessTime</td>
<td>Total amount of time (in seconds) that it took to process the bills associated with the DOM–billing segment pair during the previous billing run.</td>
<td>Any duration type value. For example, P1Y3M2DT1H20M30S (1 year, 3 months, 2 days, 1 hour, 20 minutes, and 30 seconds). The &quot;load_pin_billing_segment&quot; utility converts this value into seconds. <strong>Note:</strong> This information is generated by third-party data warehousing software.</td>
</tr>
</tbody>
</table>

Sample Billing Segment Configuration File

The following is a sample pin_billing_segment.xml file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig business_configuration.xsd">
  <BillingSegmentConfiguration>
    <BillingSegments>
      <BillingSegment ID="101">First Billing Segment</BillingSegment>
      <BillingSegment ID="102">Second Billing Segment</BillingSegment>
      <BillingSegment ID="103">Third Billing Segment</BillingSegment>
    </BillingSegments>
    <DomAssignments>
      <DomAssignment billingSegmentRef="101" status="restricted" dom="---31" maxAccounts="7400" maxServices="70033">
        <NumAccounts>4</NumAccounts>
        <NumServices>5</NumServices>
        <TotalProcessTime>PT20S</TotalProcessTime>
      </DomAssignment>
      <DomAssignment billingSegmentRef="102" status="open" dom="---07" maxAccounts="7400" maxServices="733">
        <NumAccounts>76</NumAccounts>
        <NumServices>5</NumServices>
        <TotalProcessTime>P1D</TotalProcessTime>
      </DomAssignment>
    </DomAssignments>
  </BillingSegmentConfiguration>
</BusinessConfiguration>
```

Validating Your Billing Segment Configuration File Edits

After editing the contents of the XML file, you use the "load_pin_billing_segment" utility to load the contents of the file into the /config/billing_segment object in the database. See "Setting Up Billing Segments".

Before loading the contents of the file, the utility validates the contents against the file's schema definition. If the contents do not conform to the schema definition, the load operation fails. By default, the schema definition is BRM_home/xsd/pin_billing_segment.xsd.
The XML file is not directly linked to its schema definition file. Instead, it is linked to the \BRM_home\sys\data\config\business_configuration.xsd file.

For more information, see "About Validating XML Configuration Files" in BRM System Administrator's Guide.

Updating Billing Segments

To update billing segment data, re-edit the billing segment configuration file, and then run the "load_pin_billing_segment" utility to load the updated contents of the file into the /config/billing_segment object in the BRM database. See "Setting Up Billing Segments".

**Note:**

The utility that loads billing segments into the database overwrites existing billing segments. When updating billing segments, you cannot load new segments only. You must load the complete set of billing segments each time you run the utility.

Adding Updated Billing Segment Data to the CM Cache

To add newly loaded billing segment data to the CM cache, do one of the following:

- Manually refresh the cache by stopping and restarting the CM after running the load utility.
- Automatically refresh the cache. See "Automatically Refreshing Billing Segment Data".

**Note:**

If a re-edited billing segment configuration file is significantly larger than the previous version of the file, you might have to increase the space allocated to the data in your CM cache to prevent an error from occurring at CM startup. See "Increasing the Size of the CM Cache for Billing Segment Data".

Automatically Refreshing Billing Segment Data

To refresh billing segment data automatically:

1. Open the CM configuration file (BRM_home\sys\cm\pin.conf).
2. Uncomment the fm_cust billing_segment_config_refresh_delay entry by deleting the number sign (#) at the beginning of the entry.
3. (Optional) Change the refresh frequency.

   By default, this entry is set to 86400 (24 hours, in seconds). This refreshes the cache once a day. To change the frequency, replace this value with the appropriate number of seconds.
For example, to refresh the data only once a week, change the value to 604800 (60 seconds x 60 minutes x 24 hours x 7 days).

4. Stop and restart the CM.

Note:

To turn off the refresh feature, see "Preventing Automatic Updates of Billing Segment Data".

Increasing the Size of the CM Cache for Billing Segment Data

If your billing segment configuration file contains a lot of data, you might need to increase the space allocated to that data in the CM cache to prevent an error from occurring at CM startup:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf).
2. Increase cache_size in the following entry:
   
   - cm_cache fm_cust_billing_segment number_of_entries, cache_size, hash_size
   
   The default is 51200 bytes.
   
   For example, change this:
   
   - cm_cache fm_cust_billing_segment 1, 51200, 1
   
   To this:
   
   - cm_cache fm_cust_billing_segment 1, 102400, 1
   
3. Increase cache_size in the following entry:

   - cm_cache fm_cust_dom_map number_of_entries, cache_size, hash_size

   The default is 102400 bytes.
   
   For example, change this:
   
   - cm_cache fm_cust_dom_map 1, 102400, 1
   
   To this:
   
   - cm_cache fm_cust_dom_map 1, 204800, 1

4. Stop and restart the CM.

Preventing Automatic Updates of Billing Segment Data

To prevent your system from automatically refreshing billing segment data in the CM cache:

1. Open the CM configuration file (BRM_home/sys/cm/pin.conf).
2. Comment out the fm_cust_billing_segment_config_refresh_delay entry by inserting a number sign (#) at the beginning of the entry.
3. Stop and restart the CM.
By default, this feature is turned off. To turn it on, see "Automatically Refreshing Billing Segment Data".

Associating Bill Units with Billing Segments

All the billing segments in a BRM system are defined in the /config/billing_segment object. Within that object, the PIN_FLDBILLING_SEGMENTS array contains the ID and description (PIN_FLD_DESCR) of each billing segment.

To link a bill unit to a billing segment, put the ID of the billing segment into the PIN_FLDBILLING_SEGMENT field of the /billinfo object.

Every bill unit in an account and every nonpaying bill unit associated with the account must belong to the same billing segment. If you try to associate the bill units with different billing segments, BRM returns an error.

To change a bill unit's billing segment, you need to run the PCM_OP_CUST_UPDATE_CUSTOMER opcode. See BRM Opcode Guide.

Changing a Bill Unit's Billing Segment

You can change a bill unit's billing segment in one of two ways:

- **Change the billing segment but not the billing DOM**
  
  To change a bill unit's billing segment **but not** its billing DOM, call the PCM_OP_CUST_UPDATE_CUSTOMER opcode with the new billing segment ID to put in the PIN_FLDBILLING_SEGMENT field of the /billinfo object.

  The status of the DOM in the new billing segment must be **open**. If it is not, an error is returned.

- **Change the billing segment and the billing DOM**
  
  To change both a bill unit's billing segment and its billing DOM, call PCM_OP_CUST_UPDATE_CUSTOMER with the following /billinfo field values:

  - A new billing segment ID to put in the PIN_FLDBILLING_SEGMENT field.
  - A new billing DOM to put in the PIN_FLD_ACTG_FUTURE_DOM field. The status of the DOM must be **open** in the specified billing segment.

  To change only a bill unit's billing DOM, see "Changing a Bill Unit's Billing DOM"
Assigning Accounting Days of Month to Bill Units in Billing Segments

When a bill unit is associated with a billing segment, one of the available billing DOMs in the segment must be assigned to the bill unit. A DOM's availability depends in part on the status of the DOM.

DOMs can be assigned in either of the following ways:

- **Manually Assigning a Billing DOM**
- **Automatically Assigning a Billing DOM**

To change a bill unit's billing DOM, see "Changing a Bill Unit's Billing DOM".

### Manually Assigning a Billing DOM

To assign billing DOMs to new or existing bill units manually, create an application that enables customer service representatives to select one of the open or restricted DOMs in the billing segment with which a bill unit is associated.

**Note:**

Essentially, the CSR is selecting the bill unit's *billing DOM*.

The PIN_FLD_MAP array in the `/config/billing_segment` object contains all the billing segment–DOM pairs configured in your system.

For each pair, the status of the DOM is stored in the PIN_FLD_STATUS field.

To validate the CSR's selection, the application should call PCM_OP_CUST_UPDATE_CUSTOMER. That opcode calls the PCM_OP_CUST_SET_BILLINFO opcode, which calls the PCM_OP_CUST_POL_PREP_BILLINFO and the PCM_OP_CUST_POL_VALID_BILLINFO policy opcodes.

To link a bill unit to an billing DOM, put the DOM value (1-31) into the PIN_FLD_ACTG_CYCLE_DOM field of the `/billinfo` object.

### Automatically Assigning a Billing DOM

If a billing DOM is not *manually* selected for a bill unit after the unit is assigned to a billing segment, the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode *automatically* assigns a DOM to the bill unit.

The policy opcode assigns DOMs whose status is *open*; it cannot assign DOMs whose status is *restricted*.

### Changing a Bill Unit'sBilling DOM

To change a bill unit's billing DOM, call PCM_OP_CUST_UPDATE_CUSTOMER with the new DOM to put the PIN_FLD_ACTG_CYCLE_DOM field of the `/billinfo` object.
Note:

The status of the new DOM must be open in the bill unit's billing segment. If it is not, an error is returned.

To change a bill unit's billing segment, see "Changing a Bill Unit's Billing Segment".
About Proration

This chapter information about how proration works in Oracle Communications Billing and Revenue Management (BRM).

See also:

• List of Billing Features
• BRM Concepts

Calculating Prorated Cycle Fees

BRM calculates the prorated fee for a specified period by multiplying the cycle fee defined in the applicable charge offer by a scale based on the amount of time the charge offer is valid during the cycle. The scale for a whole cycle (unit interval, or usually an accounting cycle) is equal to 1.

BRM calculates the scale for a partial cycle by dividing the number of days in the partial cycle by the number of days in the unit interval within which it falls. The number of days in a unit interval will vary depending on its start and end dates.

If the period for which the prorated fee is being calculated is greater than one unit interval, the scale will be greater than 1. Likewise, if the period is less than one unit interval, the scale will be less than 1. For example, if the cycle fee is $100, and the period being prorated is half a unit interval, the scale for that interval will be 0.5. So the prorated cycle fee is $50.

BRM does the following to calculate the prorated fee for each charge offer that is valid during the cycle:

1. Calculates the unit intervals. See "Calculating the Unit Interval".
2. Calculates the scales for the part of the period to be prorated that falls into each unit interval.
3. Sums up the scales for all the unit intervals to get the scale for the entire period to be prorated.
4. Calculates the prorated amount by multiplying the scale for the entire period by the cycle fee amount.

Calculating the Unit Interval

BRM takes into account what the use_number_of_days_in_month entry in the Connection Manager (CM) configuration (pin.conf) file is set to while calculating the unit interval. Also, the use of special days (29th, 30th, and 31st of a month) for billing is taken into account while calculating the unit interval.
If the `timestamp_rounding` entry in the CM `pin.conf` file is set to 0, the unit interval is calculated in seconds because the timestamp will not be rounded to midnight and the proration will begin from the time that is indicated by the timestamp. If `timestamp_rounding` is set to 1, the unit interval will be calculated in days because the timestamp will be rounded to midnight.

Calculating Unit Interval When `use_number_of_days_in_month` Is Not Set or 0

If `use_number_of_days_in_month` is not set or is set to 0, the unit interval is calculated based on the billing time. Assuming the billing time to be March 22, starting from the billing time, BRM calculates the last unit interval by moving to the left (on the time axis) one month at a time until the beginning of the period to be prorated is covered. For example, while calculating the prorated fee for Mar 1–Mar 15, it takes the unit interval as 28 (the number of days between February 22 and March 22) because both March 1 and March 15 fall between February 22 and March 22. See "Example 1: Use_number_of_days_in_month Is Not Set or Set to 0".

Calculating Unit Interval When `use_number_of_days_in_month` Is Set to 1

If `use_number_of_days_in_month` is set to 1 and the period to be prorated falls within the same calendar month, the unit interval is the number of days in the whole calendar month in which the period to be prorated falls. For example, while calculating the prorated fee for Mar 1–Mar 15 (both dates fall in March), BRM calculates the unit interval as 31, because March has 31 days. See "Example 2: Use_number_of_days_in_month Is Set to 1".

Note:

If the period for which BRM is prorating the fees is less than a month but spans across multiple months (for example, Aug 19–Sep 15) or if the cycle fee is for multimonth, `use_number_of_days_in_month` is ignored.

Calculating Unit Interval When Billing Day of Month Is 29, 30, or 31

When the billing day of month (DOM) is set to 29, 30, or 31, the unit interval calculation is based on the option you set for this feature (the forward or back option set in the `/config/business_params` object.) If the option is set to forward and the month does not have the billing DOM, billing will run on the first day of the next month. If the option is set to back and the month does not have the billing DOM, billing will run on the last day of the previous month. This causes the start date of the unit interval to shift based on what option is set.

Examples of Proration

Following are some examples of proration. In each of these examples, the unit interval is calculated differently.
Example 1: Use _number_of_days_in_month_ Is Not Set or Set to 0

In this example, illustrated in Figure 13-1, the prorated cycle fee is calculated for the interval Feb 15–Apr 13 with the billing time as April 22. To calculate the prorated fee for this period, BRM does the following:

Calculates the Unit Intervals for the Period

Starting from the billing time (April 22), BRM calculates the last unit interval by moving to the left one month at a time (on the time axis). Therefore, the last unit interval will be Mar 22–Apr 22. Similarly, it will continue to calculate unit intervals until the start time of the first unit interval (January 22) will be equal to or before the start time of the given period to be prorated (February 15). As a result, it gets the following unit intervals:

- Mar 22–Apr 22 (last unit interval)
- Feb 22–Mar 22 (second unit interval)
- Jan 22–Feb 22 (first unit interval)

Figure 13-1  Example 1

Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:

1. Calculates the scale for the period Feb 15–Feb 22 with the unit interval as Jan 22–Feb 22.
   
   \[ \text{Scale} = \frac{7}{31} = 0.23 \]
   
   where 7 is the number of days between February 15 and February 22 and 31 is the number of days between January 22 and February 22.

2. Calculates the scale for the period Feb 22–Mar 22 with the unit interval as Feb 22–Mar 22. This scale is \( \frac{28}{28} = 1 \) because the period to be prorated is the unit interval.

3. Calculates the scale for the period Mar 22–Apr 13 with the unit interval as Mar 22–Apr 22.
   
   \[ \text{Scale} = \frac{22}{31} = 0.71 \]
   
   where 22 is the number of days between March 22 and April 13 and 31 is the number of days between March 22 and April 22.
4. Calculates the scale for the whole period Feb 15–Apr 13 as the sum of the scales from steps 1, 2, and 3: \(0.23 + 1.0 + 0.71 = 1.94\).

Calculates the Prorated Amount

If the cycle fee amount is $100, BRM calculates the prorated amount for the period Feb 15–Apr 13 as follows:

\[1.94 \times $100 = $194\]

Example 2: Use _number_of_days_in_month_ Is Set to 1

In this example, illustrated in Figure 13-2, the prorated cycle fee is calculated for the interval Feb 15–Apr 13 with the billing time as April 22. Because _use_number_of_days_in_month_ is set to 1, BRM calculates the unit interval based on calendar month for the part of the proration time that falls in the same calendar month. To calculate the prorated fee for this period, it does the following:

Calculates the Unit Intervals for the Period

Starting from the billing time (April 22), BRM calculates the last unit interval by moving to the left one month at a time (on the time axis). Therefore, the unit intervals will be as follows:

Mar 22–Apr 22 (last unit interval)

Feb 22–Mar 22 (second unit interval)

Feb 1–Feb 28 (first unit interval)

Note:

This unit interval is the number of days in February because the entire period Feb 15–Feb 22 falls in February.

Figure 13-2 Example 2

Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:
1. Calculates the scale for the period Feb 15–Feb 22 with the unit interval as Feb 1–Feb 28.

Note:
If the part of the period to be prorated falls in the same calendar month, the unit interval is the number of days in the month. In this example, because the entire period Feb 15–Feb 22 falls in February, the unit interval is 28, the number of days in February.

Scale = 7/28 = 0.25
where 7 is the number of days between February 15 and February 22 and 28 is the number of days in February.

2. Calculates the scale for the period Feb 22–Mar 22 with the unit interval as Feb 22–Mar 22. This scale is 28/28 = 1 because the period to be prorated is the unit interval.

3. Calculates the scale for the period Mar 22–Apr 13 with the unit interval as Mar 22–Apr 22.

Scale = 22/31 = 0.71
where 22 is the number of days between March 22 and April 13 and 31 is the number of days between March 22 and April 22.

4. Calculates the scale for the whole period Feb 15–Apr 13 as the sum of the scales from steps 1, 2, and 3: 0.25 + 1.0 + 0.71 = 1.96.

Calculates the Prorated Amount
If the cycle fee amount is $100, BRM calculates the prorated amount for the period Feb 15–Apr 13 as follows:

1.96 * $100 = $196

Examples Using the 29th, 30th, and 31st for Billing Day of Month

The examples below show the use of special days (29th, 30th, 31st) as the billing DOM because of which the unit interval calculation will be based on the option you set for this feature (the forward or back option set in /config/business_params object). If the option is set to forward and the month does not have the billing DOM, billing is run on the first day of the next month. If the option is set to back and the month does not have the billing DOM, billing is run on the last day of the previous month. This causes the unit interval start date to shift based on the option set.

Example 3a: Use Forward Option with use_number_of_days_in_month Set to 0

This example, illustrated in Figure 13-3, assumes the billing option to be set to forward and use_number_of_days_in_month to be set to 0. To calculate the prorated cycle fee for the period Feb 15–Apr 13 with the billing day of month as April 30, BRM does the following:
Calculates the Unit Intervals for the Period

Starting from the billing time (April 30), BRM calculates the last unit interval by moving to the left (on the time axis) one month at a time. Therefore, the last unit interval will be Mar 30–Apr 30. Similarly, it continues to calculate unit intervals until the start time of the first unit interval (January 30) will be equal to or before the start time of the given period to be prorated (February 15). As a result, BRM gets the following unit intervals:

Mar 30–Apr 30 (last unit interval)
Mar 1–Mar 30 (second unit interval)

Note:
Because February does not have 30 days, and because the billing option is set to forward, BRM calculates this unit interval as beginning on the first day of the next month, which is March 1.

Jan 30–Mar 1 (first unit interval)

Note:
Because January does have 30 days, it calculates this unit interval as beginning on January 30.

Figure 13-3  Example 3a

Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:

1. Calculates the scale for the period Feb 15–Mar 1 with the unit interval as Jan 30–Mar 1 as follows:
   Scale = 14/30 = 0.47
where 14 is the number of days from February 15 and March 1 and 30 is the number of days from January 30 and March 1.

2. Calculates the scale for the period Mar 1–Mar 30 with the unit interval as Mar 1–Mar 30.
   
The scale for this period is $29/29 = 1$ because number of days to be prorated is equal to number of days in the unit interval.

3. Calculates the scale for the period Mar 30–Apr 13 with the unit interval as Mar 30–Apr 30 as follows:
   
   Scale = $14/31 = .45$
   
   where 14 is the number of days from March 30 and April 13 and 31 is the number of days from March 30 and April 30.

4. Adds the scales from the steps above to get the scale for the whole period:
   
   Scale for the period Feb 15–Apr 13 = $.47 + 1.0 + .45 = 1.92$

**Calculates the Prorated Cycle Fee Amount**

If the cycle fee is $100, the prorated cycle fee for the period Feb 15–Apr 13 will be:

$1.92 * $100 = $192$

**Example 3b: Use Forward Option with use_number_of_days_in_month Set to 1**

This example, illustrated in Figure 13-4, assumes the billing option to be set to forward and `use_number_of_days_in_month` to be set to 1. To calculate the prorated cycle fee for the period Feb 15–Apr 13 with the billing DOM as April 30, BRM does the following:

**Calculates the Unit Intervals for the Period**

Starting from the billing time (April 30), BRM calculates the last unit interval by moving to the left one month at a time (on the time axis). Therefore, the last unit interval will be Mar 30–Apr 30. Similarly, it continues to calculate unit intervals until the start time of the first unit interval (January 30) is equal to or before the start time of the given period to be prorated (February 15). As a result, BRM gets the following unit intervals:

- Mar 30–Apr 30 (last unit interval)
- Mar 1–Mar 31 (second unit interval)
- Jan 30–Mar 1 (first unit interval)

**Note:**

Because February does not have 30 days, and because the billing option is set to forward, BRM calculates this unit interval as beginning on the first of the next month, which is March 1. Also, `use_number_of_days_in_month` is set to 1, it calculates this unit interval to be 31, the number of days in March.
Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:

1. Calculates the scale for the period Feb 15–Mar 1 with the unit interval as Jan 30–Mar 1 as follows:
   \[
   \text{Scale} = \frac{14}{30} = 0.47
   \]
   where 14 is the number of days from February 15 and March 1 and 30 is the number of days between January 30 and March 1.

2. Calculates the scale for the period Mar 1–Mar 30 with the unit interval as Mar 1–Mar 31.
   \[
   \text{Scale} = \frac{29}{31} = 0.94
   \]
   where 29 is the number of days between March 1 and March 30 and 31 is the number of days in March.

3. Calculates the scale for the period Mar 30–Apr 13 with the unit interval as Mar 30–Apr 30 as follows:
   \[
   \text{Scale} = \frac{14}{31} = 0.45
   \]
   where 14 is the number of days from March 30 and April 13 and 31 is the number of days between March 30 and April 30.
4. Adds the scales in the steps above to get the scale for the whole period:
   
   Scale for the period Feb 15–Apr 13 = .47 + .94 + .45 = 1.86

Calculates the Prorated Cycle Fee Amount

If the cycle fee is $100, the prorated cycle fee for the period Feb 15–Apr 13 will be:

1.86 * $100 = $186

Example 3c: Use Back Option with use_number_of_days_in_month Set to 1

This example, illustrated in Figure 13-5, assumes the billing option to be set to back and use_number_of_days_in_month to be set to 1. To calculate the prorated cycle fee for the period Feb 15–Apr 13 with the billing DOM as April 30, BRM does the following:

Calculates the Unit Intervals for the Period

Starting from the billing time (April 30), BRM calculates the last unit interval by moving to the left one month at a time (on the time axis). Therefore, the last unit interval will be Mar 30–Apr 30. Similarly, it continues to calculate unit intervals until the start time of the first unit interval will be equal to or before the start time of the given period (February 15) to be prorated. As a result, it gets the following unit intervals:

Mar 30–Apr 30 (last unit interval)
Feb 28–Mar 30 (second unit interval)
Feb 1–Feb 28 (first unit interval)

Note:

Because February does not have 30 days, and because the billing option is set to back, BRM calculates this unit interval as beginning on the last day of the previous month, which is February 28.

Note:

In this example, the first unit interval would have started on January 30, because January does have 30 days, but BRM ignores this and takes the number of days in February as the unit interval because use_number_of_days_in_month is set to 1.
Figure 13-5  Example 3c

Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:

1. Calculates the scale for the period Feb 15–Feb 28 with the unit interval as Feb 1–Feb 28 as follows:
   
   \[
   \text{Scale} = \frac{13}{28} = 0.46
   \]
   
   where 13 is the number of days from February 15 and February 28 and 28 is the number of days in February.

   \[\text{Note:}\]
   
   Because February 15 and February 28 fall in the same calendar month, the unit interval here will be the number of days in February because \texttt{use_number_of_days_in_month} is set to 1.

2. Calculates the scale for the period Feb 28–Mar 30 with the unit interval as Feb 28–Mar 30.
   
   The scale for this period is \(\frac{30}{30} = 1\) because the number of days to be prorated is equal to number of days in the unit interval.

3. Calculates the scale for the period Mar 30–Apr 13 with the unit interval as Mar 30–Apr 30 as follows:
   
   \[
   \text{Scale} = \frac{14}{31} = 0.45
   \]
   
   where 14 is the number of days from March 30 and April 13 and 31 is the number of days between March 30 and April 30.

4. Adds the above three scales to get the scale for the whole period:
   
   \[
   \text{Scale for the period Feb 15–Apr 13} = 0.46 + 1.0 + 0.45 = 1.91
   \]

Calculates the Prorated Cycle Fee Amount

If the cycle fee is $100, the prorated cycle fee for the period Feb 15–Apr 13 will be:

\[1.91 \times $100 = $191\]
Example 3d: Use Back Option with use_number_of_days_in_month Set to 0

This example, illustrated in Figure 13-6, assumes the billing option to be set to back and use_number_of_days_in_month to be set to 0. To calculate the prorated fee for the period Feb 15–Apr 13 with the billing day of month as April 30, BRM does the following:

Calculates the Unit Intervals for the Period

Starting from the billing time (April 30), it calculates the last unit interval by moving to the left one month at a time (on the time axis). Therefore, the last unit interval will be Mar 30–Apr 30. Similarly, it continues to calculate unit intervals until the start time of the first unit interval will be equal to or before the start time of the given period to be prorated (February 15). As a result, BRM gets the following unit intervals:

Mar 30–Apr 30 (last unit interval)
Feb 28–Mar 30 (second unit interval)
Jan 30–Feb 28 (first unit interval)

Note:
Because February does not have 30 days, and because the billing option is set to back, BRM calculates this unit interval as beginning on the last day of the previous month, which is February 28.

Note:
Because January does have 30 days, BRM calculates this unit interval as beginning on January 30.

Figure 13-6  Example 3d
Calculates the Scale

To calculate the scale for the period Feb 15–Apr 13, BRM does the following:

1. Calculates the scale for the period Feb 15–Feb 28 with the unit interval Jan 30–Feb 28:
   \[
   \text{Scale} = \frac{13}{29} = 0.45
   \]
   where 13 is the number of days from February 15 and February 28 and 29 is the number of days from January 30 and February 28.

2. Calculates the scale for the period Feb 28–Mar 30 with the unit interval as Feb 28–Mar 30.
   The scale for this period is \(\frac{30}{30} = 1\) because the number of days to be prorated is equal to number of days in the unit interval.

3. Calculates the scale for the period Mar 30–Apr 13 with the unit interval as Mar 30–Apr 30.
   \[
   \text{Scale} = \frac{14}{31} = .45
   \]
   where 14 is the number of days from March 30 and April 13 and 31 is the number of days from March 30 and April 30.

4. Adds the scales for the above steps to get the scale for the whole period:
   \[
   \text{Scale for the period Feb 15–Apr 13} = .45 + 1.0 + .45 = 1.90
   \]

Calculates the Prorated Cycle Fee Amount

If the cycle fee is $100, the prorated cycle fee for the period Feb 15–Apr 13 will be:

\[1.90 \times 100 = $190\]

Proration for Special Cases

BRM includes proration settings that address customers with cycle arrears fees who purchase, inactivate, and reactivate charge offers within the first cycle.

You choose the rating behavior when such events occur by setting the `cycle_arrear_proration` parameter in the CM configuration (`pin.conf`) file:

\[\text{fm_rate cycle_arrear_proration} = 0|1\]

\[\text{Note:}\]

You must restart the CM after you change this parameter.

When you set up proration in a charge, you choose a proration setting for when the charge offer is purchased and for when it is canceled:

- To use the purchase-time proration setting to rate periods in which a customer both purchases and inactivates a charge offer, set the `cycle_arrear_proration` parameter to 0.
To use the cancellation proration setting to rate periods in which a customer cancels a charge offer, set the cycle_arrear_proration parameter to 1.

Special Cases

Some special proration cases may result in unexpected billing results:

- A customer purchases, inactivates, and reactivates a charge offer with a cycle arrears fee all in the same accounting cycle. The charge offer is set to prorate on purchase and to charge full on cancellation, and the cycle_arrear_proration entry in the CM pin.conf is set to 1.

  In this case, instead of charging a prorated amount based on the total time the charge offer is active during the first cycle, the full amount for the cycle is charged.

- A customer purchases, inactivates, and reactivates a charge offer with a cycle arrears fee all in the same accounting cycle. The charge offer is set to no charge on purchase and to prorate on cancellation, and the cycle_arrear_proration entry in the CM pin.conf is set to 1.

  In this case, instead of being charged nothing for the first accounting cycle, the customer is charged a prorated amount based on the time the charge offer is active, except for the time of the last active period in the cycle.

  For example, if the customer purchases a charge offer, inactivates it after 5 days, reactivates it later for another 5 days before inactivating it, and then reactivates it later for a total of 8 days before the cycle ends, you charge the customer for 5 + 5 or 10 days of use.

- A customer purchases, inactivates, and reactivates a charge offer with a cycle arrears fee all in the same accounting cycle. The charge offer is set to charge full on purchase and to prorate on cancellation, and the cycle_arrear_proration entry in the CM pin.conf is set to 1.

  In this case, instead of being charged for the whole accounting cycle, the customer is charged a prorated amount based on the total time during the cycle that the charge offer is active.

- With a cycle arrears fee, the customer inactivates and reactivates a charge offer in a cycle other than the cycle in which the charge offer is purchased. The charge offer has proration settings of charge full, no charge, or prorate on purchase and charge full or no charge on cancellation.

  In this case, the customer is charged a prorated amount equal to the time the charge offer is active during the accounting cycle from the last reactivation to the end of the cycle. There is no charge for all other active time periods during the accounting cycle.

- With cycle forward billing, the customer purchases, inactivates, and reactivates a charge offer in the same period. The charge offer has proration settings of charge full on purchase and prorate on cancellation.

  In this case, instead of being charged for the whole accounting cycle, the customer is charged for the whole cycle less the period of time that the charge offer is inactivated.
**Note:**

- You can set your policy settings so that customers are not charged for short usage periods. This establishes the possibility of long cycles over 30 days for which a customer is not billed.
- For the proration settings **Charge for the full cycle** on purchase and **Don't charge this cycle** on cancellation, a cycle is always "1" or less, even if a customer purchases and cancels within a long cycle.
- The following proration settings determine what is refunded when a charge offer for cycle forward billing is inactivated or canceled:
  - **Charge for the full cycle**: Nothing is refunded.
  - **Don't charge this cycle**: The total cycle fee is refunded.
  - **Charge based on usage**: The unused portion of the monthly usage fee is refunded.

**Addressing Special Cases**

There are several techniques you can use to address any special cases that may occur when customers inactivate and reactivate an account in the same accounting cycle:

- Inform your customers of the discrepancies.
- Do not allow your customer service representatives (CSRs) to inactivate and reactivate accounts frequently, especially during the first accounting cycle in which a charge offer is purchased.
- Because most special cases occur when you use cycle arrears fees, consider using cycle forward fees instead.

**About 30-Day-Based Proration**

To work in parallel with older legacy billing systems, you can use 30-day-based proration.

In older legacy billing systems, it is common to use 30 days for calculating proration, irrespective of the actual number of days in the month or the billing cycle. By default, BRM calculates proration based on the number of days in the billing cycle. You use 30-day-based proration when you have BRM with an older billing system requiring 30-day-based proration.

**Note:**

Making 30-day-based proration work with a normal calendar year can cause unexpected behavior. See "Special Cases".
Note:

30-day-based proration cannot be used with multimonth billing cycles.

Examples of 30-Day-Based Proration

Based on your proration setting, BRM calculates the prorated cycle amount differently, which may yield different results.

In the examples below, illustrated in Figure 13-7, Figure 13-8, Figure 13-9, and Figure 13-10, February has 28 days and billing occurs on the second day of every month. The monthly cycle forward fee is $30 and `timestamp_rounding` is set to 1 days in the CM `pin.conf` file.

Example 6: Prorated Purchase Fee with 31-day Billing Cycle

![Figure 13-7 Example 6](image)

In this example, BRM calculates the prorated cycle fee as follows:

**Using 30-Day proration**

1. Calculates the scale using 30 days as the base:
   
   \[
   \text{scale} = \frac{21}{30} = .70
   \]
   
   where 21 is the number of days from midnight January 12 to midnight February 2.

2. Calculates the prorated cycle fee:

   \[
   \text{cycle fee amount} \times \text{scale} = \$30.00 \times .70 = \$21.00
   \]

**Using 31 Days in the Billing Cycle**

1. Calculates the scale using 31 days as the base:

   \[
   \text{scale} = \frac{21}{31} = .68
   \]

   where 21 is the number of days from midnight January 12 to midnight February 2.

2. Calculates the prorated cycle fee:

   \[
   \text{cycle fee amount} \times \text{scale} = \$30.00 \times .68 = \$20.32
   \]
Example 7: Prorated Cancel Fee with 31-day Billing Cycle

In this example, BRM calculates the cycle fee refund as follows:

Using 30-Day Proration

1. Calculates the scale using 30 days as the base:
   
   scale = \( \frac{15}{30} = .50 \)
   
   where 15 is the number of days from midnight January 18 to midnight February 2.

2. Calculates the refund amount:
   
   \( \text{cycle fee amount} \times \text{scale} = $30.00 \times .50 = $15.00 \)

Using 31 Days in the Billing Cycle

1. Calculates the scale using 31 days as the base:
   
   scale = \( \frac{15}{31} = .48 \)
   
   where 15 is the number of days from midnight January 18 to midnight February 2.

2. Calculates the refund amount:
   
   \( \text{cycle fee amount} \times \text{scale} = $30.00 \times .48 = $14.52 \)

Example 8: Prorated Purchase Fee with 28-Day Billing Cycle

In this example, BRM calculates the prorated cycle fee as follows:
Using 30-Day Proration

1. Calculates the scale using 30 days as the base:
   \[
   \text{scale} = \frac{15}{30} = .50
   \]
   where 15 is the number of days from midnight February 15 to midnight March 2.

2. Calculates the prorated cycle fee:
   \[
   \text{cycle fee amount} \times \text{scale} = 30.00 \times .50 = 15.00
   \]

Using 28 Days in the Billing Cycle

1. Calculates the scale using 28 days as the base:
   \[
   \text{scale} = \frac{15}{28} = .54
   \]
   where 15 is the number of days from midnight February 15 to midnight March 2.

2. Calculates the prorated cycle fee:
   \[
   \text{cycle fee amount} \times \text{scale} = 30.00 \times .54 = 16.07
   \]

Example 9: Prorated Cancel Fee with 28-Day Billing Cycle

In this example, BRM calculates the cycle fee refund as follows:

Using 30-Day Proration

1. Calculates the scale using 30 days as the base:
   \[
   \text{scale} = \frac{15}{30} = .50
   \]
   where 15 is the number of days from midnight January 18 to midnight February 2.

2. Calculates the refund amount:
   \[
   \text{cycle fee amount} \times \text{scale} = 30.00 \times .50 = 15.00
   \]

Using 28 Days in the Billing Cycle

1. Calculates the scale using 28 days as the base:
   \[
   \text{scale} = \frac{15}{28} = .54
   \]
   where 15 is the number of days from midnight January 18 to midnight February 2.

2. Calculates the refund amount:
   \[
   \text{cycle fee amount} \times \text{scale} = 30.00 \times .54 = 16.07
   \]
Special Cases

When using 30-day proration, there can be unexpected results, as shown in the following examples:

Example 10: Full Purchase Fee Charged When Service Is Provided for 1 Day Less

In this example, illustrated in Figure 13-11, the billing cycle is 31 days. The customer purchases the charge offer on January 3, so the scale is 1. The customer pays the full $30 cycle fee even though the service is available for one day less.

Figure 13-11  Example 10

The prorated cycle fee:

\[ \text{cycle fee} \times \text{scale} = 30.00 \times 1 = 30. \]

Example 11: Full Cancel Fee Refunded When Service Has Been Used for 1 Day

In this example, illustrated in Figure 13-12, the billing cycle is 31 days and the monthly service fee is $30. The customer purchases the charge offer on January 1 and cancels the charge offer on January 2. The scale is 1. The customer gets a full $30 refund even though the customer owned the charge offer for one day.

Figure 13-12  Example 11

The refund amount is:

\[ \text{cycle fee} \times \text{scale} = 30.00 \times 1 = 30. \]
Enabling 30-Day-Based Proration

To enable 30-day-based proration:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`). `BRM_home` is the directory where you installed BRM components.
2. Edit the following entry:
   ```
   -fm_bill enable_30_day_proration 0
   ```
   where:
   - 0 bases proration on the number of days in the billing cycle or number of days in the month. This is the default.
   - 1 bases proration on a 30-day month.
3. Save and close the file.

Using Two Events to Prorate Charges for Charge Offers Whose Validity Ends in First Cycle

When prorating a cycle fee for a charge offer whose cycle validity period ends during the same billing cycle in which it is purchased, BRM can generate one or two events:

- **One event**: By default, BRM generates a single event to charge for the used portion of the cycle.
- **Two events**: BRM generates an event to charge for the entire cycle and then generates an event to refund the charge for the unused portion of the cycle (charge offer cycle validity end date through billing cycle end date).

For example, by default, if the charge offer is purchased on January 1 and its cycle validity period ends on January 16, only one event is generated for a prorated cycle fee for the number of days from midnight January 1 to midnight January 16. If you specify to create two events, the first event is generated for a cycle fee for the number of days from midnight January 1 to midnight February 1, and a second event is generated for a refund for the number of days from midnight January 16 to midnight February 1.

To enable this feature, run the `pin_bus_params` utility to change the `CreateTwoEventsInFirstCycle` business parameter. For information about this utility, see `BRM Developer's Guide`.

To use two events to prorate a cycle fee for a charge offer whose cycle validity period ends in the first cycle:

1. Go to `BRM_home/sys/data/config`.

[Note:]

To avoid such situations, use the number of days in the billing cycle or number of days in the month proration settings, not the 30-day proration setting.

---

Chapter 13
Using Two Events to Prorate Charges for Charge Offers Whose Validity Ends in First Cycle

13-19
2. Create an XML file from the /config/business_params object:
   ```bash
   pin_bus_params -r -c "Subscription" bus_params_subscription.xml
   ```
3. In the XML file, change disabled to enabled:
   ```xml
   <CreateTwoEventsInFirstCycle>enabled</CreateTwoEventsInFirstCycle>
   ```
4. Save the file as bus_params_subscription.xml.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_subscription.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see BRM System Administrator's Guide.

**Prorating Cycle Fees after a Discount Purchase or Cancellation**

When an off-cycle discount is purchased or canceled and the discount is configured to support any of the charge offer's cycle forward event types, the cycle fees are discounted for the duration of the cycle for which the discount is valid.

**Examples of Cycle Fee Proration**

These examples show how cycle fee proration works.

**Example 12: Cycle Fee Is Refunded after a Discount Purchase**

In this example, illustrated in Figure 13-13, a $30 monthly cycle fee is charged on January 1. An off-cycle discount is purchased on January 15 that discounts 10% of the monthly fee. The 10% discount is applied to the prorated monthly charge of $15 from Jan 15–Feb 1. A refund amount of $1.50 is credited to the account or service balance.

**Figure 13-13  Example 12**

![Diagram showing cycle fee proration](image)

**Example 13: Minutes Are Prorated after a Discount Cancellation**

In this example, illustrated in Figure 13-14, a cycle forward monthly event grants 100 minutes. The discount is canceled on January 15. The account made 200 minutes of calls between January 1 and January 15 at $0.10 per minute. The 100 minutes are
prorated between January 1 and January 15. The account is granted 50 minutes and is charged $0.10 per minute for the remaining 150 minutes.

**Figure 13-14  Example 13**

Free minutes are prorated for the period the discount was valid

Example 14: Canceled Discount Proration Is Not Taken into Account When Charge Offer Is Canceled

If you cancel a discount for a cycle fee and then you cancel the charge offer that owns that cycle fee, the prorated cycle fee is refunded without considering any discount that was applied. Therefore, the refunded amount might be more than the charged amount. However, this can be corrected by rerating the cycle event.

In this example, an account owns a charge offer with a cycle fee of $60, cancellation proration settings set to **Do not charge for the cycle**, and a discount of 10% on the cycle fee. At the beginning of the cycle, $54 is charged ($60 cycle fee minus $6 discount). In the middle of the cycle, if the discount is canceled, and $3 is charged back. Later, if the charge offer is canceled and $60 is refunded (because the cancellation proration setting is **Do not charge for the cycle**). Therefore, the net charge for the cycle is a credit of $3. If you rerate the events for the cycle, the expected charge of $60 will be applicable.

**Prorating Cycle Fees When a Discount's Cycle Start or End Date Is Changed**

When a discount's cycle start or end dates are changed, cycle fees may be discounted and refunded or charged depending on the new cycle start and end dates.

**Examples of Cycle Fee Proration**

These examples show how cycle fee proration works.

**Example 15: Cycle Fee Is Refunded When a Discount's Cycle Start Date Is Changed**

In this example, illustrated in **Figure 13-15**, a monthly cycle fee is charged on January 1. The monthly fee is discounted from Jan 20–Feb 1, and the discounted amount is credited to the account balance. When the discount cycle start date is changed to January 10, the monthly fee is discounted from Jan 10–Jan 20, and the discounted amount is credited to the account balance.
Example 16: Cycle Fee Is Charged When a Discount's Cycle Start Date Is Changed

In this example, illustrated in Figure 13-16, a monthly cycle fee is charged on January 1. The monthly fee is discounted from Jan 10–Feb 1, and the discounted amount is credited to the account balance. When the discount cycle start date is changed to January 20, the discounted cycle fee amount is charged from Jan 10–Jan 20 because the discount is no longer valid during this period.

Example 17: Cycle Fee Is Refunded When Discount's Cycle End Date Is Changed

In this example, illustrated in Figure 13-17, a monthly cycle fee is charged on January 1. The monthly fee is discounted from Jan 1–Jan 10. When the discount cycle end date is changed to January 20, the cycle fee is discounted from Jan 10–Jan 20, and the discounted amount is credited to the account balance.
Example 18: Cycle Fee Is Charged When Discount's Cycle End Date Is Changed

In this example, illustrated in Figure 13-18, a monthly cycle fee is charged on January 1. The monthly fee is discounted from Jan 1–Jan 20. When the discount cycle end date is changed to January 10, the discounted cycle fee amount is charged from Jan 10–Jan 20.
Managing Large Billing Runs

This chapter describes how to manage large billing runs in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

• Billing Only Specified Accounts and Bill Units
• Splitting a Billing Run into Multiple Runs
• Managing Bill Due Dates

Implementing these features includes editing the billing run configuration file, `pin_bill_run_control.xml`

When splitting a billing run, or using customized due dates, you need to run the `pin_bill_day` script manually. See "Running the pin_bill_day Script for Bill Run Management".

See also:

• List of Billing Features
• BRM Concepts

Billing Only Specified Accounts and Bill Units

To bill a single account or a limited set of accounts script, you specify the accounts and their bill units in a modified version of the billing run configuration file (`pin_bill_run_control.xml`).

When you specify the accounts to bill, BRM does not perform a database search but retrieves the account and bill unit information directly.

**Note:**

If you specify a paying bill unit that is part of a bill unit hierarchy, ensure that all nonpaying bill units are billed first.

1. Open the `BRM_home/apps/pin_billd/pin_bill_run_control.xml` file.
2. Edit the file to specify the accounts and their bill units to bill:
   
   Add an **Account** and **Billinfo** child element to the **BillingList** parent element for each account and bill unit to include. In the child element, specify the POID ID of the account and bill unit.
To bill specific accounts and bill units, you must include both the account POID and bill unit POID. If only one is specified, the account or bill unit is not billed.

A **BillingList** parent element looks like this:

```xml
<BillingList>
  <Account>Account_POID</Account>
  <Billinfo>Bill_unit_POID</Billinfo>
</BillingList>
```

For example, the following **BillingList** parent element generates bills only for the account with POID ID 55612 and its bill unit with POID ID 34589:

```xml
<BillingList>
  <Account>55612</Account>
  <Billinfo>34589</Billinfo>
</BillingList>
```

3. (Optional) Delete or comment out any billing DOM lists or billing segment lists (specified by **DOMList** and **BillSegmentList** parent elements). If you include accounts and bill units along with DOMs or billing segments, only the account's bill units are processed.

4. Save the file under a different name and close it. For example, when billing a single account, include the account POID ID in the file name (such as `pin_bill_run_account_8445.xml`); if billing a group of accounts, include the account range or reason for billing in the file name.

When you run `pin_bill_day`, the default `BRM_homelapps/pin_billd/business_configuration.xsd` file and filename must be in the same directory.

5. **Manually** run the `pin_bill_day` script using this command:

```
pin_bill_day -file filename
```

where filename is the name and location of the modified version of the billing run configuration file.

- If filename is in the same directory from which you run the `pin_bill_day` script, specify only the file name.
- If filename is in a different directory from which you run the `pin_bill_day` script, you must include the entire path for the file.
Note:

- The -file parameter when used with pin_bill_day, affects only the pin_bill_accts utility; it does not apply to other billing utilities run by the pin_bill_day script. For example, pin_cycle_fees which performs a database search to find all accounts with cycle forward fees that are due, does not use the accounts passed in with the -file parameter.

- Ensure that the accounts specified in the billing run configuration file reside on the same database schema where pin_bill_day is run. If the file contains accounts from different database schemas, pin_bill_day reports an error. See "Setting Up Billing in a Multischema Environment".

Note:

When you run pin_bill_day with a filename, do not run it as a cron job. If you do, depending on the restrictions in filename, some bill units might never be billed.

Splitting a Billing Run into Multiple Runs

When you use the pin_bill_day script to run billing, the billing run includes all bill units whose current accounting-cycle end date is any day before midnight (12:00:00 a.m.) of the day on which the billing run takes place.

To reduce the load and duration of billing runs triggered by the script, you can split large, lengthy billing runs into smaller billing runs based on billing days of month (DOMs) and billing segments. The smaller billing runs can overlap or occur at different times.

Splitting a billing run into multiple runs includes two steps:

2. Configure the split billing run.

Configuring Auto-Triggered Billing for Bill Run Management

To split large billing runs into smaller billing runs, you first disable auto-triggered billing on the days that you execute the smaller billing runs. If auto-triggered billing is enabled on those days, it reduces your ability to control the way your billing load is distributed.

For example, the billing cycle for two million customers ends on August 1. To reduce the number of bills finalized on August 1, you split the billing run into four smaller runs, each of which includes about 500,000 bill units. You execute one smaller billing run per day from August 1 through August 4. If auto-triggered billing is enabled on those days, bill-triggering events might cause BRM to finalize some bills in the smaller billing runs before you execute the runs. For example, bill-triggering events might cause BRM to finalize 1,250,000 bills on August 1 instead of only 500,000.
To configure auto-triggered billing for bill run management:

1. Set a systemwide billing delay. See "Setting Up Delayed Billing".
   By default, auto-triggered billing is always enabled. To disable it, you must set a billing delay in your BRM system.

   **Note:**
   If you use delayed billing, skip this step. A billing delay is already set in your system. For more information, see "Setting up delayed billing".

   By default, after you set a billing delay, auto-triggered billing is enabled for only the delay period and the last two days of each bill unit's accounting cycle.

   For example, if your delayed billing cycle is 7 days long, auto-triggered billing is disabled during the following days in a 31-day accounting cycle as shown in Figure 14-1:

   **Figure 14-1   Auto-Triggered Billing Disabled in Accounting Cycle**

   ![31-day accounting cycle diagram]

   **Note:**
   If a bill-triggering event occurs during the delayed-billing period, the bill is only partially processed: rollovers and cycle fees are applied, but the bill is not finalized. If a bill-triggering event occurs during the last two days of the cycle, the bill is finalized.

2. (Optional) Change the number of days auto-triggered billing is enabled at the end of each accounting cycle. See "Configuring Auto-Triggered Billing".

Configuring a Split Billing Run

To split a billing run into smaller billing runs, you configure the smaller billing runs in multiple versions of the billing run configuration file (pin_bill_run_control.xml). Each of the smaller billing runs is limited to bill units associated with one or both of the following:
Specified accounting days of month (DOMs). Bill units associated with any other billing DOM are excluded from the billing run.

Specified billing segments. Bill units associated with any other billing segment are excluded from the billing run.

To restrict the smaller billing runs to specified billing segments, you must first set up billing segments in your system and then associate bill units with them. See "Load Balancing Billing Runs".

Note:
When you split a large billing run into smaller billing runs:
- Configure the smaller billing runs so that in total they include all due bill units.
- To ensure that no bill unit remains unbilled, periodically run the pin_bill_day script without the -file parameter.
- Do not include the same billing segment in multiple small billing runs. If you do, your accounts receivable (A/R) data may become inaccurate.

To split a billing run into smaller billing runs:
1. Open the pin_bill_run_control.xml file in an XML editor or a text editor. By default, the file is in the BRM_home/apps/pin_bill directory.

Note:
You also edit this file to specify billing run-time due date adjustments. See “Specifying Due Date Adjustments in a Billing Run”.

2. Edit the file to restrict a billing run to bill units associated with one of the following:
   - For specific billing DOMs, see "Including Only Specified Billing DOMs in Billing Runs".
   - For specific billing segments, see "Including Only Specified Billing Segments in Billing Runs".

Note:
Only one smaller billing run at a time can be configured in a billing run configuration file.

3. Save the file.
4. Repeat steps 1 through 3 as often as necessary to configure a set of smaller billing runs that includes all due bill units in a daily billing run.

5. Manually run the `pin_bill_day` script with each version of the XML file by using this command:

   `pin_bill_day -file filename`

   where filename is the name and location of a version of the billing run configuration file.

   - If you copy filename to the same directory from which you run the `pin_bill_day` script, specify only the file name.
   - If you run the command in a different directory from where filename is located, you must include the entire path for the file.
   - In addition, filename must be in the same directory as the default `BRM_home/apps/pin_billd/business_configuration.xsd` file.

---

**Note:**

- The `-file` parameter when used with `pin_bill_day`, affects only the `pin_bill_accts` utility; it does not apply to other billing utilities run by the `pin_bill_day` script. For example, `pin_cycle_fees` which performs a database search to find all accounts with cycle forward fees that are due, does not use the accounts passed in with the `-file` parameter.

- Ensure that the accounts specified in the billing run configuration file reside on the same database schema where `pin_bill_day` is run. If the file contains accounts from different database schemas, `pin_bill_day` reports an error. See "Setting Up Billing in a Multischema Environment".

---

**Note:**

When you run `pin_bill_day` with a configuration file, do not run it as a cron job. If you do, depending on the restrictions in configuration file, some bill units might never be billed.
About Sponsored Charges in Split Billing Runs

If the following conditions occur when you split a billing run, sponsored cycle forward charges might appear in the sharing group owner's bill one cycle late:

- Sharing group owner account and member account have the same billing DOM.
- Sharing group owner account and member account belong to different billing segments.

Including Only Specified Billing DOMs in Billing Runs

To include bill units associated only with specified billing DOMs in a billing run, add a `DOM` child element to the `DOMList` parent element in the billing run configuration file for each day whose bill units you want to include.

A `DOMList` parent element looks like this:

```xml
<DOMList>
  <DOM>---gDay1</DOM>
  <DOM>---gDay2</DOM>
  <DOM>---gDay3</DOM>
</DOMList>
```

where `gDay` is any two-digit value from 01 through 31.

For example, to include only bill units whose billing DOM is 1 or 15, add child elements:

```xml
<DOMList>
  <DOM>---01</DOM>
  <DOM>---15</DOM>
</DOMList>
```

If the `DOMList` parent element is omitted, bill units associated with any billing DOM can be included in the billing run.

Including Only Specified Billing Segments in Billing Runs

To include bill units associated only with specified billing segments in a billing run, add a `BillSegment` child element to the `BillSegmentList` parent element in the billing run configuration file for each billing segment whose bill units you want to include.

A `BillSegmentList` parent element looks like this:

```xml
<BillSegmentList>
  <BillSegment>ID</BillSegment>
  <BillSegment>ID</BillSegment>
  <BillSegment>ID</BillSegment>
</BillSegmentList>
```

where `ID` is the ID of any billing segment defined in the `/config/billing_segment` object in your BRM system.

For example, to include only bill units associated with billing segments 101 or 102, add child elements:

```xml
<BillSegmentList>
  <BillSegment>101</BillSegment>
</BillSegmentList>
```
For information about billing segments, see "Load Balancing Billing Runs".

If the BillSegmentList parent element is omitted, bill units associated with any billing segment can be included in the billing run.

Sample Billing Run Configuration File

The following is a sample pin_bill_run_control.xml file. Billing run loads are restricted to billing DOMs and billing segments specified in the bold elements.

```xml
<DOMList>
  <DOM>---03</DOM>
  <DOM>---07</DOM>
  <DOM>---15</DOM>
</DOMList>

<BillSegmentList>
  <BillSegment>101</BillSegment>
  <BillSegment>102</BillSegment>
</BillSegmentList>

<DueDateAdjustment Length=5 >
  <PaymentTerm id = 1001 />
  <PaymentTerm id = 1002 />
</DueDateAdjustment>

<DueDateAdjustment Length = 7/>
```

For information about the DueDateAdjustment and PaymentTerm elements, see "Specifying Due Date Adjustments in a Billing Run".

Validating Your Billing Run Configuration File Edits

After editing the XML file, you use the file name as a parameter when you run the pin_bill_day script. The script passes the file name to the "pin_bill_accts" utility, which validates the contents of the XML file against its schema definition. If the contents do not conform to the schema definition, the utility returns an error. The schema definition is BRM_home/xsd/pin_bill_run_control.xsd.

The XML file is not directly linked to its schema definition file. Instead, it is linked to the XSD reference file BRM_home/apps/pin_billd/business_configuration.xsd.

For more information about the XSD reference file, see "About Validating XML Configuration Files" in BRM System Administrator's Guide.

Managing Bill Due Dates

This section explains how to manage bill due dates by performing the following tasks:

- **Managing Payment Terms.** Payment terms enable you to use different methods to calculate due dates for bills that have the same payment method.

- **Managing Billing Calendars.** Billing calendars enable BRM to implement payment terms based on business days.

- **Specifying Due Date Adjustments in a Billing Run.** Billing run-time adjustments enable you to accommodate operational delays in your billing process.
By default, the bill due date for any payment method is 30 days after the date the bill is finalized. You can change default due dates only at the payment-type level.

The BRM delayed billing feature enables billing for all the bill units in your system to be run a specified number of days after the end of their billing cycle. If you use delayed billing, be careful to avoid configuring bill due dates that occur before bills are finalized. For example, if your system has a 14-day billing delay and you configure a bill due date that is fewer than 14 days after the end of a bill unit's billing cycle, the bill due date will occur before the bill is finalized.

Managing Payment Terms

A payment term specifies how to calculate the due date of a bill. You use payment terms to set due dates a specified number of days after the billing cycle end date or on a specified day of the month. For example, payment terms can be linked to functions that set bill due dates as follows:

• 21 days after the billing cycle end date
• 15 business days after the billing cycle end date
• The second Tuesday of the month

You can define as many payment terms as you need in your BRM system. Each payment term can be associated with one or more /payinfo objects, but a /payinfo object can be associated with only one payment term at a time. The /payinfo object's payment term applies to all bills associated with the object.

Payment terms enable you to use different methods to calculate due dates for bills that have the same payment method. For example, by default, all bills paid by direct debit are due on the day the bill is finalized. Customer A pays a monthly bill by direct debit and has paychecks automatically deposited every two weeks. However, Customer A wants the account debited on the third Tuesday of every month regardless of when the bill is finalized. To do so without affecting other direct debit customers, you can create a third-Tuesday-of-the-month payment term and associate it with the /payinfo object to which customer A’s bill is linked.

To manage payment terms, you perform these tasks:

• Editing the Payment Terms Configuration File
• Loading Payment Terms
• Assigning Payment Terms to Bill Units

Editing the Payment Terms Configuration File

You configure all the payment terms in your BRM system in the BRM_home/sys/datal config/pin_payment_term.xml file.

In the file, the PaymentTerms parent element must contain a PaymentTerm child element for each payment term in your system. A PaymentTerm child element looks like this:

  <PaymentTerm ID="int">description</PaymentTerm>

where int is a payment term ID. For more information, see Table 14-1.

To create a payment term, add a PaymentTerm child element to the PaymentTerms parent element. In the child element, specify values for the items listed in Table 14-1:
Table 14-1 Payment Term Elements

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the payment term.</td>
<td>Any nonnegative integer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Payment term ID 0 is the default payment term.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Payment term IDs 1 through 1000 are reserved for BRM use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Important:</strong> In the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode, the ID number must be associated with a function that calculates bill due dates. See the chapter about billing in BRM Opcode Guide.</td>
</tr>
<tr>
<td>description</td>
<td>An explanation of the payment term. For example, <strong>3rd Tuesday of the month</strong>.</td>
<td>Any string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This string is mapped to the PIN_FLD_DESCR field in the <strong>config/payment_term</strong> object, which can be used to populate a list of payment terms in a user interface (UI). When creating the string, take any UI length restrictions into account.</td>
</tr>
</tbody>
</table>

Sample Payment Terms Configuration File

The following is a sample **pin_payment_term.xml** file:

```xml
<?xml version="1.0" encoding="UTF-8"?>

    <!-- Sample file. Modify according to guidelines -->
    <PaymentTermConfiguration>
        <PaymentTerms>
            <PaymentTerm ID="1001">17 days</PaymentTerm>
            <PaymentTerm ID="1002">14 business days</PaymentTerm>
            <PaymentTerm ID="1003">3rd Tuesday of the month</PaymentTerm>
        </PaymentTerms>
    </PaymentTermConfiguration>
</BusinessConfiguration>
```

Loading Payment Terms

To set up payment terms in your system, edit the payment terms configuration file (**pin_payment_term.xml**), and then load its contents into the **config/payment_term** object in the BRM database.

1. Open the **pin_payment_term.xml** file in an XML editor or a text editor.

   The file is in the **BRM_home/sys/data/config** directory.

2. Enter the appropriate information into the file. See "Editing the Payment Terms Configuration File".

3. Save the file.
4. Use the following command to run the "load_pin_payment_term" utility from the directory in which the pin_payment_term.xml file is located:

```bash
load_pin_payment_term pin_payment_term.xml
```

If you do not run the utility from the directory in which pin_payment_term.xml is located, include the complete path to the file; for example:

```bash
load_pin_payment_term BRM_home/sys/data/config/pin_payment_term.xml
```

5. Stop and restart the Connection Manager (CM).

6. For each payment term in your system, customize the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode to specify the function and parameters to use to calculate the due dates of bills associated with the payment term. See BRM Opcode Guide.

Updating the pin.conf File to Use Payment Terms

To use configured payment terms, update the pin.conf file.

1. Open the BRM_home/sys/cm/pin.conf file in a text editor.

2. Add the following line:

   ```bash
   - cm_cache fm_cust_pol_paymentterm_cache 256, 40960, 64
   ```

3. Save and close the file.

4. Stop and restart the CM.

Assigning Payment Terms to Bill Units

To assign a payment term to a bill unit, you associate the payment term with the /payinfo object to which the /billinfo object is linked.

To do this at account creation time, call the PCM_OP_CUST_COMMIT_CUSTOMER opcode to put the ID of the payment term into the PIN_FLD_PAYMENT_TERM field of the appropriate /payinfo object. (PCM_OP_CUST_COMMIT_CUSTOMER calls the PCM_OP_CUST_SET_PAYINFO opcode to perform this task.)

Note:

- All the payment terms in a BRM system are stored in the /config/payment_term object. In the object, the PIN_FLD_PAYMENT_TERMS array contains the ID and description (PIN_FLD_DESCR) of each payment term.

- If you do not assign a payment term to a /payinfo object at account creation time, BRM automatically assigns the default payment term ID 0 to the /payinfo object. This occurs even if there are no payment terms in your system.

To do this at account maintenance time, call the PCM_OP_CUST_UPDATE_CUSTOMER opcode to perform the task described in the preceding paragraphs.
Managing Billing Calendars

By default, a billing calendar contains a list of days on which bills cannot be due. Billing calendars enable BRM to implement payment terms based on business days. To calculate due dates for bills associated with such payment terms, BRM must determine which days of the year are considered business days and which are not. To do so, it uses billing calendars to exclude days such as weekends, holidays, and other user-specified nonbilling days from the calculation.

For example, if a billing cycle ends on December 10, 2016, and its payment term adds 15 business days to that date, a billing calendar can be used to prevent the bill's payment being due on New Year's Eve.

You can add multiple billing calendars to your system to accommodate different countries and customers. For example, a US billing calendar would include Thanksgiving and Independence Day (July 4) as nonbilling days when they occur on weekdays.

When calculating due dates for bills associated with payment terms based on business days, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode uses billing calendars to exclude days such as weekends, holidays, and any other user-specified nonbilling day from the calculation.

To manage billing calendars, you perform these tasks:

- Setting Up Billing Calendars
- Editing the Billing Calendar Configuration File
- Updating Billing Calendars
- Associating Billing Calendars with Payment Terms

Setting Up Billing Calendars

To set up billing calendars in your system, edit the billing calendar configuration file (pin_calendar.xml), and then load its contents into /config/calendar objects in the BRM database (each calendar is loaded into a separate object).

1. Open the pin_calendar.xml file.

   By default, the file is in the BRM_home/sys/data/config directory.
2. Enter the appropriate information into the file. See "Editing the Billing Calendar Configuration File ".

Note:

- Bill run management includes a default billing calendar. Its case-sensitive name is default. By default, the PCM_OP_BILL_POL CALC_PYMT_DUE_T policy opcode uses this calendar to calculate due dates for payment terms based on business days.
- When you edit the configuration file, if you unintentionally change or delete the calendar, due date calculations based on business days may be incorrect.

3. Save the file.

4. Use the following command to run the "load_pin_calendar" utility from the directory in which the pin_calendar.xml file is located:

   load_pin_calendar pin_calendar.xml

Note:

- When you run the utility, the pin_calendar.xml and business_configuration.xsd files must be in the same directory. By default, both files are in BRM_home/sys/data/config.
- This utility needs a configuration (pin.conf) file in the directory from which you run the utility. For information about creating configuration files for BRM utilities, see "Creating Configuration Files for BRM utilities" in BRM System Administrator's Guide.

If you do not run the utility from the directory in which pin_calendar.xml is located, include the complete path to the file, for example:

   load_pin_calendar BRM_home/sys/data/config/pin_calendar.xml

5. Stop and restart the CM.

Editing the Billing Calendar Configuration File

You configure all the billing calendars in your BRM system in the BRM_home/sys/data/config/pin_calendar.xml file.
This configuration file contains one predefined billing calendar. Its case-sensitive name is default. By default, the PCM_OP_BILL_POL_Calc_PYMT_DUE_T policy opcode uses this calendar to calculate due dates for payment terms based on business days.

When you edit the configuration file, if you unintentionally change or delete the calendar, due date calculations based on business days may be incorrect.

To edit this configuration file, open it in an XML editor or a text editor.

In the file, the CalendarConfiguration parent element must contain a Calendar child element for each billing calendar in your system. A Calendar child element looks like this:

```xml
<Calendar name="calendar_name">
  <Date>
    <!-- day_description -->
    <Day>---dd</Day>
    <Month>---mm--</Month>
    <Year>yyyy</Year>
  </Date>
</Calendar>
```

For an example, see "Sample Billing Calendar Configuration File".

To add a calendar to the file, see "Adding Calendars".

To add a day to a calendar, see "Adding Days to Calendars".

### Adding Calendars

To create a billing calendar, add a Calendar child element to the CalendarConfiguration parent element. In the child element, specify values for the items listed in Table 14-2:

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
</table>
| name                     | The name of the billing calendar, such as Gregorian, European_Holidays, and Korean. The name of the default billing calendar is default. | Any string. Maximum length is 255 characters. **Note**
|                          |             | • The name is case sensitive. • This string is mapped to the PIN_FLD_NAME field in the I config/calendar object, which can be used to populate a list of billing calendars in a user interface (UI). When creating the string, take any UI length restrictions into account. |
Table 14-2 (Cont.) Calendar Elements

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>A day on which you do not want bill payments to be due. A Calendar child element can have multiple Date elements. For example, in a US billing calendar, you might include a Date element for every US holiday (New Year’s Day, President’s Day, Memorial Day, and so on). <strong>Note</strong>: The default due date function that uses billing calendars (fm_bill_pol_add_n_bus_days) automatically skips weekends, so you do not need to create a Date element for Sundays and Saturdays in calendars used only by that function. See the chapter about billing in BRM Opcode Guide.</td>
<td>See &quot;Adding Days to Calendars&quot;.</td>
</tr>
</tbody>
</table>

Adding Days to Calendars

In a Calendar child element, you must add a Date element for each day on which you do not want bill payments to be due. A Date element looks like this:

```xml
<Date>
  <!-- description_of_day -->
  <Day>---dd</Day>
  <Month>--mm--</Month>
  <Year>yyyy</Year>
</Date>
```

To add a day to a calendar, add a Date element to the Calendar element. In the Date element, specify values for the items listed in Table 14-3:

Table 14-3 Date Element Entries

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>A day of the month.</td>
<td>1 through 31.</td>
</tr>
<tr>
<td>Month</td>
<td>A month of the year.</td>
<td>1 through 12.</td>
</tr>
</tbody>
</table>
| Year                     | The year in which the date is a nonbilling day. | One of the following:
  - To associate the date with one year only, specify the appropriate year in yyyy format (for example, 2005).
  - To associate the date with all years (arecurring nonbilling day), use 0000.
  - For example, to make December 25 (Christmas) a recurring nonbilling day, use these values: Day 25, Month 12, Year 0000. |
Sample Billing Calendar Configuration File

The following is a sample pin_calendar.xml file:

```xml
<?xml version="1.0"?>
  <!-- Sample file. Modify according to guidelines -->
  <CalendarConfiguration>
    <Calendar name="default">
      <!-- holiday specific to indicated year -->
      <Date>
        <!-- Thanksgiving -->
        <Day>---25</Day>
        <Month>--11--</Month>
        <Year>2004</Year>
      </Date>
    </Calendar>
    <Calendar name="Western Australia">
      <!-- recurring holiday on same date. use 0 for year value -->
      <Date>
        <!-- Christmas -->
        <Day>---25</Day>
        <Month>--12--</Month>
        <Year>0000</Year>
      </Date>
      <!-- holiday date specific to indicated year -->
      <Date>
        <!-- Anzac Day -->
        <Day>---26</Day>
        <Month>--04--</Month>
        <Year>2004</Year>
      </Date>
      <Date>
        <!-- Australia Day -->
        <Day>---26</Day>
        <Month>--01--</Month>
        <Year>2005</Year>
      </Date>
    </Calendar>
  </CalendarConfiguration>
</BusinessConfiguration>
```

Validating Your Billing Calendar Configuration File Edits

After editing the contents of the XML file, you use the "load_pin_calendar" utility to load the contents of the file into `config/calendar` objects in the BRM database. See "Setting Up Billing Calendars".

Before loading the contents of the file, the utility validates the contents against the file's schema definition. If the contents do not conform to the schema definition, the load operation fails. The schema definition is in this file:

`BRM_home/xsd/pin_calendar.xsd`

The XML file is not directly linked to its schema definition file. Instead, it is linked to this XSD reference file:
BRM_home/sys/data/config/business_configuration.xsd

For more information, see "About validating XML configuration files" in BRM System Administrator's Guide.

Updating Billing Calendars

To update billing calendars, re-edit the billing calendar configuration file, and then run the "load_pin_calendar" utility to load the updated contents of the file into /config/calendar objects in the BRM database. See "Setting Up Billing Calendars" for the complete procedure.

**Note:**

This utility overwrites existing billing calendars. When updating billing calendars, you cannot load new calendars only. You must load the complete set of billing calendars each time you run the utility.

To add the newly loaded data to the CM cache, stop and restart the CM.

Associating Billing Calendars with Payment Terms

When using payment terms based on business days to calculate bill due dates, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode uses billing calendars to omit nonbilling days from the calculation.

To associate a billing calendar with such payment terms, see the chapter about billing in BRM Opcode Guide.

Specifying Due Date Adjustments in a Billing Run

You use **billing run-time due date adjustments** to add days to the due dates of bills in a billing run. You can add the same number of days to all the bills in the billing run, or you can add different numbers of days to bills whose accounts are associated with different payment terms.

Billing run-time adjustments enable you to accommodate operational delays in your billing process. For example, bills associated with payment term A are due on the third Thursday of the month. On May 3, a problem in your system makes you unable to run billing. On May 10, you fix the problem and run billing for accounts whose billing DOM is 3. As a result, DOM 3 bills are generated a week late. To make up for this delay, you add a due date adjustment of 7 days for payment term A to the billing run. This gives DOM 3 customers associated with payment term A the usual time between receipt of their bill and its due date.

**Note:**

To specify adjustments based on the payment term with which a bill is associated, you must first set up payment terms in your system and associate accounts with them. See "Loading Payment Terms".
To specify billing run-time due date adjustments:

1. Open the billing run configuration file (pin_bill_run_control.xml) in an XML editor or a text editor.

   By default, the file is in the BRM_home/apps/pin_bill directory.

   **Note:**
   
   You also use this file to split your daily billing run into smaller billing runs. See "Configuring a Split Billing Run".

2. Specify the appropriate due date adjustments in the file. See “Editing the Billing Run Configuration File to Adjust Bill Due Dates”.

3. Save the file.

   **Note:**
   
   If you want, you can save a copy of the file under a different name. The pin_bill_day script can take any XML file name as a parameter if the file’s contents conform to the appropriate schema definition. See “Validating Your Billing Run Configuration File Edits”.

4. **Manually** run the pin_bill_day script, using this syntax:

   ```bash
   pin_bill_day -file filename
   ```

   where filename is the name and location of the billing run configuration file.

   - If you copy filename to the same directory from which you run the pin_bill_day script, specify only the file name.
   - If you run the command in a different directory from where filename is located, you must include the entire path for the file.
   - In addition, filename must be in the same directory as the default BRM_home/apps/pin_bill/business_configuration.xsd file.

   **Note:**
   
   - The -file parameter when used with pin_bill_day, affects only the pin_bill_accts utility.
   - Ensure that the accounts specified in the billing run configuration file reside on the same database schema where pin_bill_day is run. If the file contains accounts from different database schemas, pin_bill_day reports an error. See "Setting Up Billing in a Multischema Environment".
When you run `pin_bill_day` with a configuration file, do not run it as a `cron` job. If you do, depending on the restrictions in configuration file, some bill units might never be billed.

Editing the Billing Run Configuration File to Adjust Bill Due Dates

By editing the `BRM_home/apps/pin_billd/pin_bill_run_control.xml` file and then using it as a parameter for the `pin_bill_day` script, you can add days to the due dates of bills in a billing run. The due date adjustment can apply to all bills in the billing run or to bills associated only with specified payment terms.

To edit the file, open it in an XML editor or a text editor, and then perform one or both of these tasks:

- **Associating Due Date Adjustments with Payment Terms**
- **Specifying a Default Due Date Adjustment**

By default, the added days include weekends and nonbusiness days. To add only **business days**, change the following code in the `PCM_OP_BILL_POL_CALC_PYMT_DUE_T` policy opcode from this:

```plaintext
if (due_date_adjust) {
    fm_utils_add_n_days(due_date_adjust, &due_t);
}
```

To this:

```plaintext
if (due_date_adjust) {
    fm_bill_pol_add_n_bus_days(ctxp, n, "billing_calendar_name", &due_t, ebufp);
}
```

For information about the `fm_bill_pol_add_n_bus_days` function, see the chapter about billing in **BRM Opcode Guide**.

**Associating Due Date Adjustments with Payment Terms**

To specify an adjustment for bills associated with a particular payment term, add the appropriate `Payment.Term id` element to a `Due.date.adjustment` element in the billing run configuration file:

```plaintext
<Due.date.adjustment length = n >
    <Payment.Term id = x />
</Due.date.adjustment >
```

where:

- n is a positive integer that represents the number of days to add to the due date of a bill.
- x is the ID of any payment term defined in the `/config/payment_term` object. See "Managing Payment Terms".

You must specify both n and x. For example, the following 5-day adjustment applies only to bills associated with payment terms 1001 and 1002:
Calculating Due Dates Based on Both Payment Terms and Adjustments

When due date adjustments are associated with payment terms, bill due dates are calculated as follows:

- **Payment terms that add days**
  \[ \text{billing cycle end date} + \text{payment term} + \text{adjustment} = \text{due date} \]
  For example, if the billing cycle end date is April 1, 2001, the payment term 1001 is “add 7 days to the billing cycle end date,” and the due date adjustment is 5 days, this calculation is used:
  \[ \text{April 1} + 7 \text{ days} + 5 \text{ days} = \text{April 13} \]

- **Payment terms that specify a particular day of month**
  \[ \text{payment term} + \text{adjustment} = \text{due date} \]
  For example, if payment term 1002 is “second Tuesday of the month,” the due date adjustment is 5 days, and billing is run on April 1, 2004, this calculation is used:
  \[ \text{April 8} (\text{second Tuesday of April 2004}) + 5 \text{ days} = \text{April 13} \]

Specifying a Default Due Date Adjustment

To specify a due date adjustment for bills in a billing run that are not associated with a payment term or whose payment term is not associated with an adjustment, add a default adjustment:

\[ <\text{Due_date_adjustment length = n} /> \]

where \( n \) is a positive integer that represents the number of days to add to the due date of a bill.

For example, if the following adjustments are included in the same XML file, the 7-day adjustment applies to bills associated with any payment term except payment terms 1001 and 1002:

\[ <\text{Due_date_adjustment length = 5} /> <\text{Payment_Term id = 1001} /> <\text{Payment_Term id = 1002} /> </\text{Due_date_adjustment} > <\text{Due_date_adjustment length = 7} /> \]

Sample Billing Run Configuration File

The following sample \text{pin_bill_run_control.xml} file contains these due date adjustments:

- A 5-day adjustment for bill units associated with payment terms 1001 and 1002
- A 7-day adjustment for all the other bill units in the billing run

Due date adjustments are configured in the \textbf{bold} elements.
Validating Your Billing Run Configuration File Edits

After editing the XML file, you use the file name as a parameter when you run the pin_bill_day script. See "Specifying Due Date Adjustments in a Billing Run".

The script passes the file name to the pin_bill_accts utility, which validates the contents of the file against its schema definition. If the contents do not conform to the schema definition, the utility returns an error. The schema definition is in this file:

BRM_home/xsd/pin_bill_run_control.xsd

The XML file is not directly linked to its schema definition file. Instead, it is linked to this XSD reference file:

BRM_home/apps/pin_billd/business_configuration.xsd

For more information, see "About validating XML configuration files" in BRM System Administrator's Guide.
This document describes Oracle Communications Billing and Revenue Management (BRM) bill suppression and explains how to implement it.

See also:
• List of Billing Features
• BRM Concepts

About Suppressing Bills

BRM finalizes a bill at the end of each billing cycle. When a bill is finalized, the status of its bill items is changed from pending to open so that they stop accumulating charges and so that payments can be applied to them. In addition, a due date is added to the bill. A new bill is created to handle bill items for the next billing cycle.

Bill suppression, however, enables you to postpone finalizing a bill until the end of a future billing cycle. When a bill is suppressed, it is extended to include bill items for the next cycle, and the bill continues to accumulate charges until the end of that cycle.

Note:
Charges accrued during all cycles in which a bill is suppressed do not age, get invoiced, go into collections, or have a due date set for them until suppression ends and the bill is finalized.

You use bill suppression to avoid sending out unnecessary bills and incurring wasteful expenses. For example, if the cost to create and mail a bill is greater than the balance due, you can suppress the bill until its balance due is greater than its production costs.

Bills can be suppressed in several ways. See these topics:
• About Automatic Bill Suppression
• About Manual Bill Suppression
• About Manual Account Suppression

All types of bill suppression can be overridden by exceptions. See "Exceptions to Bill Suppression".

About Automatic Bill Suppression

At the end of a billing cycle, BRM can automatically suppress bills whose balance is less than a user-specified minimum required to finalize a bill. Such bills are suppressed for one billing cycle. If their balance is still below the minimum at the end of that cycle, they are suppressed for another billing cycle.
If the number of consecutive billing cycles for which a bill is suppressed reaches your specified maximum number of cycles, the bill is generated even if its balance is still below the minimum. This ensures that an excessive amount of time does not pass between customer bills.

To implement automatic bill suppression, you specify minimum balance amounts and maximum cycle limits for each customer segment that includes accounts whose bills you want to suppress automatically. A customer segment's specifications apply to all the bill units in the accounts that belong to the segment.

For example, a customer segment for low-usage accounts with bad payment histories might have a bill-generation threshold of only $5 and a limit of only three consecutively suppressed cycles, whereas a customer segment for high-usage accounts with good payment histories might have a bill-generation threshold of $15 and a limit of six consecutively suppressed cycles.

If an account belongs to more than one customer segment, the lowest minimum balance and the lowest maximum cycle settings associated with the customer segments apply to the account. These settings can be from different customer segments.

For example, account X belongs to customer segments A and B. Segment A's minimum balance is $5 and its maximum cycle setting is 4. Segment B's minimum balance is $10 and its maximum cycle setting is 2. Thus, account X's minimum balance is $5 (from segment A) and its maximum cycle limit is 2 (from segment B).

For more information, see "Automatically Suppressing Bills".

About Manual Bill Suppression

Manual bill suppression enables you to suppress individual bills programmatically or through a custom user interface on a case-by-case basis.

For example, if you use automatic bill suppression, you can use manual bill suppression to suppress bills whose balance does not qualify for automatic suppression, as in this situation: Customer A's account belongs to customer segment X. The minimum balance required to finalize bills associated with accounts in customer segment X is $10. Midway through the current billing cycle, customer A's balance
is $105, so his bill does not qualify for automatic bill suppression and will be finalized at the end of the billing cycle. Because customer A is having cash flow problems, however, he calls a customer service representative (CSR) and asks her to suppress his bill for two billing cycles. Using an interface that interacts with the manual bill suppression feature, she manually suppresses his bill for the requested number of cycles.

**Note:**
Unlike automatic bill suppression, the default manual bill suppression feature does not use customer segments.

For more information, see the chapter about billing in *BRM Opcode Guide*.

### About Manual Account Suppression

Manual account suppression enables you to suppress accounts on request. With this feature, customers who will not be using their account for an extended period of time can retain all their services and connection IDs without accumulating any of the charges usually associated with their account.

Optionally, charges associated with one account-level bundle can accumulate during account suppression. You can use this bundle to handle any special fees you want to charge while an account is suppressed. For example:

- Charge a purchase fee for suppressing an account.
- Charge a low monthly cycle fee for retaining a suppressed account's services and connection IDs.

**Note:**
Unlike automatic bill suppression, account suppression does not use customer segments.

To prevent owner accounts from being suppressed, add the appropriate logic to the `PCM_OP_BILL_POL_CHECK_SUPPRESSION` policy opcode. For more information, see the chapter about billing in *BRM Opcode Guide*.

### Suppressed Accounts versus Inactive Accounts

A suppressed account differs from an inactive account as shown in Table 15-1:

**Table 15-1  Differences between Suppressed and Inactive Accounts**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Account Type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the account inactive?</td>
<td>Suppressed account</td>
<td>No.</td>
</tr>
<tr>
<td>Is the account inactive?</td>
<td>Inactive account</td>
<td>Yes.</td>
</tr>
</tbody>
</table>
### Table 15-1  (Cont.) Differences between Suppressed and Inactive Accounts

<table>
<thead>
<tr>
<th>Condition</th>
<th>Account Type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are the account's services inactive?</td>
<td>Suppressed account</td>
<td>Yes.</td>
</tr>
<tr>
<td>Are the account's services inactive?</td>
<td>Inactive account</td>
<td>Yes.</td>
</tr>
<tr>
<td>Are the account’s bills finalized?</td>
<td>Suppressed account</td>
<td>No. Accrued charges do not age, get invoiced, or go into collections.</td>
</tr>
<tr>
<td>Are the account’s bills finalized?</td>
<td>Inactive account</td>
<td>Yes. Charges accrued before the account is inactivated age, get invoiced, and go into collections.</td>
</tr>
<tr>
<td>Can new charges accrue in the account?</td>
<td>Suppressed account</td>
<td>Yes. Optionally, charges associated with one account-level bundle can accrue.</td>
</tr>
<tr>
<td>Can new charges accrue in the account?</td>
<td>Inactive account</td>
<td>No.</td>
</tr>
<tr>
<td>Does the status of the account's child accounts change?</td>
<td>Suppressed account</td>
<td>No. Bills for any nonpaying child bill units of the parent account's paying parent bill units are finalized. Their charges continue to accrue in the suppressed parent account's bill that is not yet finalized.</td>
</tr>
<tr>
<td>Does the status of the account's child accounts change?</td>
<td>Inactive account</td>
<td>Yes. All child accounts that have nonpaying bill units are inactivated.</td>
</tr>
<tr>
<td>Is charge sharing affected?</td>
<td>Inactive account</td>
<td>Yes. Charge sharing is suspended. Formerly sponsored charges accrue in member account bills while the owner account is inactive.</td>
</tr>
<tr>
<td>Is discount sharing affected?</td>
<td>Inactive account</td>
<td>Yes. Member account events impact member accounts’ balances, not the inactive owner account’s balance.</td>
</tr>
<tr>
<td>Is charge sharing affected?</td>
<td>Suppressed account</td>
<td>No. Member accounts’ sponsored charges continue to accrue in the suppressed owner account's unfinalized bill.</td>
</tr>
<tr>
<td>Is discount sharing affected?</td>
<td>Suppressed account</td>
<td>No. Member account events continue to impact the balance of the suppressed discount sharing group owner's unfinalized bill.</td>
</tr>
</tbody>
</table>

### Exceptions to Bill Suppression

All types of bill suppression can be overridden by exceptions. The default exceptions are listed in Table 15-2:
### Table 15-2 Exceptions to Bill Suppression

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payment received</td>
<td>The receipt of a payment requires that a bill be finalized to record the payment against the bill. By default, this exception is disabled. To enable it, see the chapter about billing in <em>BRM Opcode Guide</em>.</td>
</tr>
<tr>
<td>Adjustment or credit applied</td>
<td>If an adjustment or a credit is made to an account, the bill is finalized to notify the customer about the change in the balance.</td>
</tr>
<tr>
<td>Maximum consecutive cycle suppressions exceeded</td>
<td>The maximum number of consecutive billing cycles for which a bill can be suppressed is specified for the customer segment to which the bill's account belongs. See &quot;Associating Bill Suppression Information with Customer Segments&quot;.</td>
</tr>
<tr>
<td>First or last bill</td>
<td>An account's first and last bills are always finalized at the end of their billing cycles, even if their balance is below the minimum balance required to finalize bills.</td>
</tr>
<tr>
<td>Account closed</td>
<td>Bills associated with closed accounts cannot be suppressed.</td>
</tr>
</tbody>
</table>

To add or delete exceptions, see the chapter about billing in *BRM Opcode Guide*.

**Note:**

The **Bill Now** functionality and on-purchase billing for bundles and packages do not override bill suppression. Although they finalize a suppressed bill, they do not:

- Reset the counter in the `/billinfo` object that tracks consecutively suppressed billing cycles (PIN_FLD_NUM_SUPPRESSED_CYCLES).
- End manual bill or manual account suppression.

### How Exceptions Affect Manual Bill and Account Suppression

Exceptions do not end manual bill or manual account suppression. Rather, they cause the PCM_OP_BILL_MAKE_BILL opcode to do the following:

- Finalize the manually suppressed bill at the end of its current billing cycle.
- Reset the counter in the `/billinfo` object that tracks the bill's *consecutively suppressed* billing cycles (PIN_FLD_NUM_SUPPRESSED_CYCLES) to 0.
• Decrement the counter in the /billinfo object that tracks the bill's *remaining manually suppressed* cycles (PIN_FLD_SUPPRESSION_CYCLES_LEFT) by 1.

For example, a bill is manually suppressed for 10 billing cycles. At the end of the fifth cycle, however, it is finalized because of an exception. At that time, the bill's *consecutively suppressed* cycles counter is reset to 0, and its *remaining manually suppressed* cycles counter is decremented by 1. Because the latter counter was also decremented by 1 at the end of the four previous suppressed billing cycles, its value is now 5, which indicates that the bill should be suppressed for 5 more cycles.

### Automatically Suppressing Bills

To implement manual bill suppression and manual account suppression, see *BRM Opcode Guide*.

To set up automatic bill suppression:

1. Set up customer segments in your system and add accounts to them. See "Creating and Managing Customer Segments" in *BRM Managing Customers*.

   ![Note:](image)

   Whether or not you set up customer segments, all accounts belong to customer segment 0. Thus, to implement automatic bill suppression *without* creating customer segments, perform step 2 for customer segment 0. The suppression specifications associated with customer segment 0 apply to *all* the accounts in your system.

2. For each customer segment that includes accounts whose bills you want to suppress automatically, specify the following information:
   - Minimum balance required for a bill to be finalized.
   - Maximum number of consecutive billing cycles that a bill can be suppressed.

   See "Associating Bill Suppression Information with Customer Segments".

### Editing the Bill Suppression Configuration File

In the *BRM_home/sys/data/config/pin_bill_suppression.xml* file, you specify the following information for each customer segment that includes accounts whose bills you want to suppress automatically:

- Minimum balance required for a bill to be finalize.
- Maximum number of consecutive billing cycles that a bill can be suppressed.

To edit this configuration file, open it in an XML editor or a text editor.

In the file, the *CustomerSegmentArray* parent element must contain a *CustomerSegment* child element for each customer segment to which you want to add bill suppression information. A *CustomerSegment* child element looks like this:

```
<CustomeSegment ID="int">
  <MinBillAmount>decimal</MinBillAmount>
  <MaxSuppressionCycles>int</MaxSuppressionCycles>
</CustomerSegment>
```
To specify bill suppression information for a customer segment, add a `CustomerSegment` child element to the `CustomerSegmentArray` parent element. In the child element, specify values for the items listed in Table 15-3:

**Table 15-3  Customer Segment Bill Suppression Elements**

<table>
<thead>
<tr>
<th>XML Element or Attribute</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the customer segment. Customer segments are defined in an array in the <code>config/customer_segment</code> object. The index of each array entry is the ID of a customer segment. The IDs of the customer segments to which an account belongs are specified in the <code>PIN_FLD_CUSTOMER_SEGMENT_LIST</code> field of the <code>laccount</code> object.</td>
<td>Any nonnegative integer.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Suppression data associated with nonexistent customer segment IDs is ignored until the IDs are defined in the <code>config/customer_segment</code> object.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Customer segment ID 0 is the default customer segment. All accounts belong to this customer segment.</td>
<td></td>
</tr>
<tr>
<td>MinBillAmount</td>
<td>Minimum balance required to finalize a bill. If the balance is less than this amount, the bill is automatically suppressed.</td>
<td>Any positive number with two decimal places.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Although balances are stored in account currency, this value is not converted to a particular currency. For example, if this value is 5.00, it represents 5 US dollars, 5 Australian dollars, 5 euros, and so on.</td>
<td></td>
</tr>
<tr>
<td>MaxSuppressionCycles</td>
<td>Maximum number of consecutive billing cycles for which a bill can be suppressed.</td>
<td>Any integer greater than 0.</td>
</tr>
</tbody>
</table>

Sample Bill Suppression Configuration File

The following is a sample `pin_bill_suppression.xml` file:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<BusinessConfiguration
    xmlns="http://www.portal.com/schemas/BusinessConfig"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig business_configuration.xsd">

<!-- Sample file, Modify according to guidelines -->
<BillSuppressionConfiguration>
    <CustomerSegmentList>
        <CustomerSegment ID="1001">
            <!-- Bad customer -->
            <MinBillAmount>5.55</MinBillAmount>
            <MaxSuppressionCycles>2</MaxSuppressionCycles>
        </CustomerSegment>
        <CustomerSegment ID="1002">
            <!-- Good customer -->
            <MinBillAmount>99.99</MinBillAmount>
            <MaxSuppressionCycles>5</MaxSuppressionCycles>
        </CustomerSegment>
    </CustomerSegmentList>
</BillSuppressionConfiguration>
</BusinessConfiguration>
```
Validating Your Bill Suppression Configuration File Edits

After editing the contents of the XML file, you use the “load_pin_bill_suppression” utility to load the contents of the file into the `config/suppression` object in the database. See "Associating Bill Suppression Information with Customer Segments".

Before loading the contents of the file, the utility validates the contents against the file’s schema definition. If the contents do not conform to the schema definition, the load operation fails. The schema definition is in this file:

`BRM_home/xsd/pin_bill_suppression.xsd`

The XML file is not directly linked to its schema definition file. Instead, it is linked to this XSD reference file:

`BRM_home/sys/data/config/business_configuration.xsd`

For more information about the XSD reference file, see "About Validating XML Configuration Files" in *BRM System Administrator's Guide*.

Associating Bill Suppression Information with Customer Segments

To implement bill suppression, edit the bill suppression configuration file (`pin_bill_suppression.xml`) and then load its contents into the `config/suppression` object in the BRM database.

1. Open the `pin_bill_suppression.xml` file in an XML editor or a text editor.
   
   By default, the file is in the `BRM_home/sys/data/config` directory, where `BRM_home` is the directory where you installed BRM components.

2. In the file, specify the following information for each customer segment that contains accounts whose bills you want to suppress:
   
   • Minimum balance required for a bill to be finalized.
   
   • Maximum number of consecutive billing cycles that a bill can be suppressed.

   See "Editing the Bill Suppression Configuration File".

3. Save the file.

4. Use the following command to run the “load_pin_bill_suppression” utility from the directory in which the `pin_bill_suppression.xml` file is located:

   `load_pin_bill_suppression pin_bill_suppression.xml`

**Note:**

When you run the utility, the `pin_bill_suppression.xml` and `business_configuration.xsd` files must be in the same directory. By default, both files are in `BRM_home/sys/data/config`. See "Validating Your Bill Suppression Configuration File Edits".
If you do not run the utility from the directory in which `pin_bill_suppression.xml` is located, include the complete path to the file, for example:

```bash
load_pin_bill_suppression BRM_home/sys/data/config/pin_bill_suppression.xml
```

For more information, see "load_pin_bill_suppression".

5. Stop and restart the Connection Manager (CM).

6. To verify that the bill suppression information was loaded, display the `config/suppression` object by using Object Browser or the `robj` command with the `testnap` utility.

For general instructions on using `testnap`, see "Using testnap" in BRM Developer's Guide. For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in BRM Developer's Guide.
Creating Custom Bill Items

This document describes how to create custom bill items and how Oracle Communications Billing and Revenue Management (BRM) assigns custom bill items to events.

See also:
- List of Billing Features
- BRM Concepts

About Custom Bill Items

Bill items enable you to track a customer's balance for a type of event. For example, a bill item tracks all charges for service usage or all charges for cycle fees during a billing cycle.

By default, BRM tracks balances in the following bill items: cycle arrears items, cycle forward items, cycle forward arrears items, cycle tax items, cycle incentive items, and usage items.

You can create custom bill items to further aggregate charges and to provide more descriptive information in your invoices, reports, and CSR applications. For example, if you charge customers for password changes, you can track password changes separately and list the charges on invoices under “password change” rather than “usage.”

About Defining Custom Bill Items

When you create a custom bill item, you define the following:

- **The bill item name.** This is the item name displayed on customer invoices, reports, and CSR applications.

- **How to track charges.** You specify whether a bill item stores one charge only or accumulates multiple charges. See "Tracking Charges in Bill Items".

- **How to store the item in the database.** You can either pre-create a custom /item object in the database or have BRM create one for you. See "About Creating /item Objects".

- **The type of events you want the bill item to track.** You do this by assigning bill items either to an event and service combination or to event attributes. See "About Assigning Custom Bill Items to Events".

Tracking Charges in Bill Items

When you create a custom bill item, you specify whether the item accumulates charges or tracks each charge separately.
Cumulative bill items accumulate charges throughout the billing cycle. All events of the same type are consolidated into a single item object. For example, if a customer has three broadband sessions during a billing cycle, BRM stores all of the charges in one item object. The customer’s invoice also lists one item with the total rolled-up charge for all three sessions, as shown in Figure 16-1:

Figure 16-1 Rolled-Up Broadband Usage Charges on Invoice

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Description</th>
<th>$ Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-7,1</td>
<td>Internet usage</td>
<td>19.50</td>
</tr>
</tbody>
</table>

Individual bill items store a charge for a single event, such as purchasing a charge offer. A separate item object is created for each event of the same type. For example, if a customer purchases three ring tones during a billing cycle, BRM stores the charges in three separate item objects. The customer’s invoice also lists three separate bill items and their charges, as shown in Figure 16-2:

Figure 16-2 Individual Bill Items

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Description</th>
<th>$ Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-7,1</td>
<td>Rng tone purchase</td>
<td>2.50</td>
</tr>
<tr>
<td>B-7,2</td>
<td>Rng tone purchase</td>
<td>2.50</td>
</tr>
<tr>
<td>B-7,3</td>
<td>Rng tone purchase</td>
<td>2.50</td>
</tr>
</tbody>
</table>

You specify whether a custom bill item is cumulative or individual in the config_item_types.xml file. See "Mapping Item Tags to Item Types".

About Creating /item Objects

You create your custom bill items in the database by subclassing the item object. For example, you can create an item/password object for storing password charges.

For usage events, you can specify whether BRM pre-creates the custom bill item before any event occurs or creates it during the rating process. In this case, the item is created when a service object is created for an account and when billing is run.

About Assigning Custom Bill Items to Events

You assign custom bill items to events in either of two ways:
• Assign bill items to a specific event and service combination. See "About Using Event and Service Combinations to Assign Bill Items".

• Assign bill items to events based on event attributes. See "About Using Event Attributes to Assign Bill Items".

About Using Event and Service Combinations to Assign Bill Items

You can assign a bill item to each event and service combination that you support. For example, you can map the event and service combinations to a bill item object as shown in Table 16-1:

Table 16-1  Event and Service Combinations for Bill Item Object

<table>
<thead>
<tr>
<th>Event and Service Combination</th>
<th>Item Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>/event/session/telco/gsm</td>
<td>/item/gsm_usage</td>
</tr>
<tr>
<td>/service/telco/gsm/*</td>
<td></td>
</tr>
<tr>
<td>/event/session/telco/gprs</td>
<td>/item/gprs_usage</td>
</tr>
<tr>
<td>/service/telco/gprs/*</td>
<td></td>
</tr>
<tr>
<td>/event/session/email</td>
<td>/item/email_usage</td>
</tr>
<tr>
<td>/service/email</td>
<td></td>
</tr>
</tbody>
</table>

In Table 16-1, BRM separates charges for GSM usage, GPRS usage, and email usage into three item objects and displays each item separately on customer invoices.

You can assign one bill item to one event and service combination or assign one bill item to multiple event and service combinations. For example, you can assign item/voice to any event and service combination that provides voice service.

You can also create multiple item configurations (sets of item-to-event-and-service mappings) to apply to different types of bill units. This enables you to avoid creating unnecessary bill items in your BRM system. See "Improving Performance by Using Multiple Item Configurations" in BRM System Administrator's Guide.
Note:

A large number of items per account or bill unit (/billinfo object) can decrease system performance. Additionally, account creation and billing failures can occur when there are a large number of item types for an account or service that results in the maximum lengths for PIN_FLD_ITEM_POID_LIST and PIN_FLD_NEXT_ITEM_POID_LIST fields to be exceeded. Item POIDs are appended to PIN_FLD_ITEM_POID_LIST and PIN_FLD_NEXT_ITEM_POID_LIST in the /account object when a new item type is created for an account or service. The following options are recommended when creating custom item types:

- Limit the number of item types such that if a customer uses all the event and service combinations defined in the config_item_tags.xml and config_item_types.xml, the number of item poids for an account or service does not exceed 2000 bytes. See “Assigning Item Tags Based on Event and Service Combinations”.
- Create item types with PRECREATE set to FALSE. By setting PRECREATE to FALSE, the items are created only when the particular event occurs for the service. This enables minimal items to be created during account creation or billing. See “Mapping Item Tags to Item Types”.

To assign event and service combinations to bill items, you perform these tasks:

- Map event and service combinations to item tags by editing the config_item_tags.xml file. See “Assigning Item Tags Based on Event and Service Combinations”.
- Map item tags to item types by editing the config_item_types.xml file. See “Mapping Item Tags to Item Types”.

You can configure how BRM assigns event and service combinations to bill items by adding the – fm_act attach_item_to_event n entry in the CM pin.conf file, where n can be 0, 1, or 2. If the value of n is:

- **0**: BRM does not assign events without balance impact to any item. The events that have balance impact are assigned to items according to the event and service combinations defined in the appropriate config/item_tags object. This is the default value.
- **1**: BRM assigns any event to items. This includes events without any balance impact.
- **2**: BRM assigns any event to items. If the event has a service and the event and service combination is not defined in a config/item_tags object, BRM assigns the event to the default item (item/misc) on the account and not on service.

To add the attach_item_to_event entry:

1. Open the Connection Manager (CM) configuration file (BRM_home/sys/cm/pin.conf, where BRM_home is the directory in which you installed BRM components).
2. Add the following entry:
About Using Event Attributes to Assign Bill Items

You can provide even more granularity in your reports, invoices, and CSR applications by assigning items by event attribute. This enables you to assign multiple items to the same event and service combination.

For example, for the event and service combination of /event/session/telco/gsm and /service/telco/gsm/*, you can separate events by the following:

- Calls that originated in New York in a custom item object named /item/new_york.
- Calls that originated in California in a custom item object named /item/california.

In the example shown in Figure 16-3, the customer’s invoice displays an itemized list of GSM usage:

![Figure 16-3 Itemized List of GSM Usage](image)

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Description</th>
<th>$ Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-7,1</td>
<td>Calls from New York</td>
<td>8.50</td>
</tr>
<tr>
<td>B-7,2</td>
<td>Calls from California</td>
<td>6.00</td>
</tr>
<tr>
<td>B-7,3</td>
<td>Roaming calls</td>
<td>13.07</td>
</tr>
</tbody>
</table>

To assign items based on event attributes, you perform the following tasks:

- Map event attributes to an item tag by customizing a policy opcode. See "Assigning Item Tags Based on Event Attributes".
- Map items tags to item types by editing the config_item_types.xml file. See "Mapping Item Tags to Item Types".

How BRM Assigns Custom Bill Items to Events

BRM assigns bill items to events during the rating process by performing the following tasks:

1. BRM assigns an item tag based on the event and service combination.
2. If customized to do so, the BRM custom API takes the assigned item tag as input and then assigns a different item tag based on the event's attributes.
3. BRM assigns an item type based on the item tag.

Cumulative Custom Item for Taxes

The following example shows a custom item that stores all taxes from all tax suppliers in a single bill item for each billing cycle.

1. Open the $BRM_home/sys/data/pricing/example/config_item_tags.xml$ file.
2. Add the following entry:

   `<ItemTagElement>
   <ItemTag>cycle_tax</ItemTag> <EventType>/event/billing/cycle/tax</EventType> 
   <ServiceType>/account</ServiceType> </ItemTagElement>`

3. Save and close the file.
4. Open the $BRM_home/sys/data/pricing/example/config_item_types.xml$ file.
5. Add the following entry:

   `<ItemTypeElement>
   <ItemTag>cycle_tax</ItemTag> <ItemDescription>Cycle Tax</ItemDescription> 
   <ItemType precreate="false" type="cumulative">/item/cycle_tax</ItemType>
   </ItemTypeElement>`

6. Save and close the file.

Setting Up BRM to Assign Custom Bill Items to Events

To assign items to events and to update the items in the BRM database for billing and tracking:

1. Create the custom bill item in the database by subclassing the `Item` storable class. See "Creating Custom Fields and Storable Classes" in $BRM Developer's Guide$.

   `Note:
   Before creating a bill item, you must know the event and service type whose balance impacts you want stored with the new item.`

2. Map event and service combinations to item tags in the appropriate item configuration. See "Assigning Item Tags Based on Event and Service Combinations".
3. Map event attributes to item tags. See "Assigning Item Tags Based on Event Attributes".
4. Map item tags to item types in the appropriate item configuration. See "Mapping Item Tags to Item Types".
Assigning Item Tags Based on Event and Service Combinations

You map event and service combinations to custom item tags by editing the `config_item_tags.xml` file. You then load the item tags into a `/config/item_tags` object by using the `load_config_item_tags` utility. See "load_config_item_tags".

Every item tag in the item tags file must have a corresponding item type defined in the `config_item_types.xml` file. If you make changes to the `config_item_tags.xml` file after you load it into the database, you must make corresponding changes to the item types and load the `config_item_types.xml` file again. See "Mapping Item Tags to Item Types".

To assign item tags:

1. Open the `BRM_home/sys/data/pricing/example/config_item_tags.xml` file in a text editor.
2. Verify that the value of the following optional tag is correct:

   ```xml
   <Name>Item_Configuration_Name</Name>
   ```

   Where `Item_Configuration_Name` is the name of the item configuration to which the item tags defined in this XML file belong.

   The name of the BRM system's default item configuration is `Default`. If the value of this tag is `Default`, or if this tag is not included in the file, your custom tags are added to the default item configuration.

   To add a new item configuration to your system, enter a unique item configuration name in this XML file and in the corresponding `config_item_types.xml` file.

   To modify an existing item configuration in your system, enter that configuration's name in this XML file and in the corresponding `config_item_types.xml` file.

   For more information about item configurations, see "Improving Performance by Using Multiple Item Configurations" in BRM System Administrator's Guide.

3. Add custom tags to the file by following the instructions in the file.

   **Note:**

   Tag names must be unique.

   For example, to store GSM calls in a custom bill item, add the following entry:

   ```xml
   <ItemTagElement>
     <ItemTag>GSM</ItemTag>
     <EventType>/event/delayed/session/telco/GSM/*</EventType>
     <ServiceType>/service/telco/GSM/*</ServiceType>
   </ItemTagElement>
   ```

   Where:

   - The `ItemTag` element specifies the unique name for the item tag.
   - The `EventType` element specifies the parent `event` storable class.
   - The `ServiceType` element specifies the parent `service` storable class.
4. Save the file. You can save the file with a different name and location or use the original file.

5. Run the following command

```
load_config_item_tags config_item_tags_file
```

Where `config_item_tags_file` is the name and path of your `config_item_tags.xml` file.

![Note:](image)

The `load_config_item_tags` utility replaces the entire contents of the `config/item_tags` object whose PIN_FLD_NAME value matches the Name value in the `config_item_tags.xml` file with the contents of that file. If you are updating a set of item tags, you cannot load new items only. You must load complete sets of items each time you run the `load_config_item_tags` utility.


To verify that the item tags were loaded, use Object Browser in Developer Center or the `testnap` utility's `robj` command to display the specified `config/item_tags` object. See "Reading an Object and Writing Its Contents to a File" in BRM Developer's Guide.

Assigning Item Tags Based on Event Attributes

You assign item tags based on event attributes by customizing the BRM API.

Setting Up Online Charging to Assign Items Based on Event Attributes

You set up online charging to assign items to events based on event attributes by using the `PCM_OP_BILL_POL_GET_ITEM_TAG` policy opcode. By default, this policy opcode does nothing. However, you can customize it to find events with specific flist fields, assign the appropriate item tag, and then return the item tag in the PIN_FLD_ITEM_TAG output flist field. See BRM Opcode Guide.

Mapping Item Tags to Item Types

You map the `item tags` that you defined in the `config_item_tags.xml` file or policy opcode to `item types` by using the `config_item_types.xml` file. You then load the mappings into the appropriate `/config/item_types` object by using the `load_config_item_types` utility.
Note:
The `load_config_item_types` utility replaces the entire contents of the specified `/config/item_types` object whose PIN_FLD_NAME value matches the Name value in the `config_item_types.xml` file with the contents of that file. If you are updating a set of mappings, you cannot load new mappings only. You must load complete sets of mappings each time you run the `load_config_item_types` utility.

Note:
If the `config_item_types.xml` file does not contain a PIN_FLD_NAME value, the `/config/item_types` object whose PIN_FLD_NAME value is Default is replaced.

To map item tags to item types:

1. Open the `BRM_home/sys/data/pricing/example/config_item_types.xml` file in an XML editor or a text editor.
2. Verify that the value of the following tag is correct:
   ```xml
   <Name>Item_Configuration_Name</Name>
   ```
   Where `Item_Configuration_Name` is the name of the item configuration to which the item types defined in this XML file belong.

   The name of the BRM system's default item configuration is Default. If the value of this tag is Default, or if this tag is not included in the file, your custom mappings are added to the default item configuration.

   To add a new item configuration to your system, enter a unique item configuration name in this XML file and in the corresponding `config_item_tags.xml` file.

   To modify an existing item configuration in your system, enter that configuration's name in this XML file and in the corresponding `config_item_tags.xml` file.

   For more information about item configurations, see "Improving Performance by Using Multiple Item Configurations" in BRM System Administrator's Guide.

3. Map the item tags you created to custom item types by following the instructions in the file.

   For example, to map the item tag `new_york` to the item type `/item/new_york`, add the following entry:

   ```xml
   <ItemTypeElement>
     <ItemTag>new_york</ItemTag>
     <ItemDescription>Calls from New York</ItemDescription>
     <ItemType precreate="false" type="cumulative">/item/new_york</ItemType>
   </ItemTypeElement>
   ```

   Where:
The **ItemTag** element specifies the name of the item tag.

The **ItemDescription** element specifies the item name that is displayed in customer invoices, reports, and CSR applications.

The **precreate** element specifies whether BRM pre-creates the item for the service type: **true** specifies to pre-create the item in the database, and **false** specifies to create the item when the event occurs.

**Note:**

BRM pre-creates items for usage events only. It does not pre-create items for purchase or cycle fee events.

The **type** element specifies whether to track balances separately or to consolidate balances: **cumulative** specifies that this bill item stores charges for all events of the same type in a billing cycle; **individual** specifies to create a separate item for each event. See "Tracking Charges in Bill Items" for more information.

**Note:**

If you have set **precreate** to **false** for the usage or delayed events that are rated using Pipeline Manager, do not set the **type** to **individual**.

The **ItemType** element specifies the name of the custom **/item** object. In this example, BRM stores the item charges in the **/item/new_york** object.

4. Save the file. You can save the file with a different name and location or use the original file.

5. Run the following command:

   ```
   load_config_item_types config_item_types_file
   ```

   where **config_item_types_file** is the name and path of your **config_item_types.xml** file.

**Caution:**

The **load_config_item_types** utility replaces the entire contents of the **/config/item_types** object whose PIN_FLD_NAME value matches the Name value in the **config_item_types.xml** file with the contents of that file. If you are updating a set of mappings, you cannot load new item types only. You must load complete sets of mappings each time you run the **load_config_item_types** utility.

6. Stop and restart the CM. See "Starting and Stopping the BRM System" in **BRM System Administrator's Guide**.
To verify that the item tags were loaded, use Object Browser in Developer Center or the testnap utility's robj command to display the specified config/item_types object. See "Reading an Object and Writing Its Contents to a File" in BRM Developer's Guide.

### Assigning Bill Items to Event Balance Impacts

All events contain, or can contain, a balance impact. You can use custom item types to separately track charges in individual balance impacts of an event. For example, even though only one BRM event is recorded for a service, if you charge for both connection time and the amount of bytes transferred during a session, the two charges can be tracked separately.

PIN_FLD_ITEM_TAGS is an array in the output flist of the PCM_OP_BILL_POL_GET_ITEM_TAG policy opcode. The PIN_FLD_ITEM_TAGS enables you to create an item tag for one or more balance impacts. You choose from an event, which balance impacts to use, and what item tags they are assigned to. The item type is assigned using the matching element ID of the item tag and balance impact.

The following example, lists the account POID and custom item tags in flist array format. Balance impacts with element IDs 2 and 3 have item types assigned based on the item tags TransferVolume and ConnectionPeriod. All other balance impacts have an item type assigned based on the item tag SessionUsage.

```
0     PIN_FLD_POID                POID [0] 0.0.0.1 /account 182477 0
0     PIN_FLD_ITEM_TAG             STR [0] SessionUsage
0     PIN_FLD_ITEM_TAGS          ARRAY [2] allocated 20, used 3
1     PIN_FLD_ITEM_TAG             STR [0] TransferVolume
0     PIN_FLD_ITEM_TAGS          ARRAY [3] allocated 20, used 3
1     PIN_FLD_ITEM_TAG             STR [0] ConnectionPeriod
```

**Note:**

If a PIN_FLD_ITEM_TAGS array element is not specified for a balance impact, the balance impact will have an item assigned based on the PIN_FLD_ITEM_TAG element at the top level of the policy opcode output.

To assign bill items to event balance impacts:

1. Create your custom balance impact item tags and types. See "Setting Up BRM to Assign Custom Bill Items to Events".
2. Map your custom balance impact item tags to your custom balance impact item types. See "Mapping Item Tags to Item Types".
3. Customize the PCM_OP_BILLPOL_GET_ITEM_TAG policy opcode with the appropriate business logic to:
   a. Determine the element ID of the required balance impact of the event in the input flist of the PCM_OP_BILL_POL_GET_ITEM_TAG policy opcode.
   b. Create a PIN_FLD_ITEM_TAGS array element in the output flist of the PCM_OP_BILL_POL_GET_ITEM_TAG policy opcode with the element ID being the same as the balance impact ID from step a.
c. Set the field PIN_FLD_ITEM_TAGS of the PIN_FLD_ITEM_TAGS array element to the custom value of the item tag that you require for the balance impact.

For information about customizing policy opcodes, see "Adding and Modifying Policy Facilities Modules" in BRM Developer's Guide.

4. Assign your custom items to event balance impacts during the rating process.

By default, any item tag specified for a sharing group owner's balance impact is ignored and an item of type /item/sponsor is used. A custom item type can be assigned for a sharing group owner's balance impact only in the following situations:

• The SplitSponsorItemByMember business parameter is enabled. See "Splitting Sponsored Charges into Multiple Items”.

• A custom item tag is specified for the sharing group owner's balance impact in the PIN_FLD_ITEM_TAGS array.

Creating Custom Sponsored Bill Items

By default, BRM accumulates the charges for all charge sharing member services and accounts belonging to one owner in a single /item object.

You can create custom sponsored bill items that divide the accumulated charges across the members of the charge sharing group.

To create custom sponsored bill items:

1. Enable the SplitSponsorItemByMember business parameter. See "Splitting Sponsored Charges into Multiple Items”.

The charges are broken down into:

• One /item/sponsor object for each charge sharing member service instance.

• One /item/sponsor object for account-level charges for all member accounts.

The sponsored items point to the owner account object and to the sharing member service. If the shared charges are at the member account level, the service pointer is NULL.

2. Create your custom balance impact /item/sponsor tags and types. See "Setting Up BRM to Assign Custom Bill Items to Events" and "Assigning Bill Items to Event Balance Impacts”.

Note:

When configuring a custom item type for sharing group owner's balance impacts, specify the base type without the component sponsor in the item type string. For example, to use peak_usage for sponsored peak usage charges, configure the tag as /item/peak_usage. BRM automatically uses the correct sponsor subtype /item/sponsor/peak_usage at the time of rating.

3. Assign the custom sponsored items to event balance impacts during the rating process.
Splitting Sponsored Charges into Multiple Items

By default, splitting sponsored charges into multiple sponsored items is disabled. You can enable splitting sponsored charges for online charging, offline charging, or both.

If you split sponsored charges for offline charging, you do not have the option of disabling the pre-updater step of the Rated Event (RE) Loader. This is because the pre-updater stored procedure assigns sponsored items to events when the splitting option is enabled.

To enable this feature, run the `pin_bus_params` utility to change the `SplitSponsorItemByMember` business parameter. For information about this utility, see *BRM Developer’s Guide*.

To enable splitting sponsored charges into multiple items:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   
   ```bash
   pin_bus_params -r SplitSponsorItemByMember -bus_params_billing.xml
   ```

3. In the XML file, change `disabled` to `enabled`:
   
   ```xml
   <SplitSponsorItemByMember>disabled</SplitSponsorItemByMember>
   ```

4. Do one of the following:
   
   - To enable splitting of sponsored charges for both online charging and offline charging, change `disabled` to `enabled`.
   - To enable splitting of sponsored charges for only online charging, change `disabled` to `onlyRealTime`.
   - To enable splitting of sponsored charges for only offline charging, change `disabled` to `onlyBatch`.

   ```xml
   <SplitSponsorItemByMember>disabled</SplitSponsorItemByMember>
   ```

5. Save the file as `bus_params_billing.xml`.

6. Load the XML file into the BRM database:

   ```bash
   pin_bus_params bus_params_billing.xml
   ```

7. Stop and restart the CM.

8. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator’s Guide*. 

Creating Corrective Bills

This document describes how to create corrective bills in Oracle Communications Billing and Revenue Management (BRM).

See also:

- List of Billing Features
- BRM Concepts

About Corrective Bills

Correcting a bill is necessary when a bill or invoice sent to a customer requires an update. Corrections can include:

- Corrections to a bill resulting from changes that do not affect the amount owed by the customer. For example, a change to the invoice address on the customer's account or a correction to the language on the customer's profile.
- Corrections to the charges on a bill resulting from adjustments, for example, settled disputes that require some item adjustment, catalog corrections such as retroactive tariff updates, and so on.

A corrective bill is a bill that is generated after corrections are made to a regular bill or a bill that was previously corrected. You ensure that the adjustments are appropriately allocated and then generate the corrective bill. When a corrective bill is generated, it is assigned a new due date based on the current payment setup for the account.

You can create a series of corrective bills for a bill. For example, you can generate a corrective bill for a regular bill, and then a corrective bill for the first corrective bill, and so on.

A corrected invoice is the invoice associated with a corrective bill and, based on your billing configuration, contains the entire bill information or only the corrections and adjustments. For information on corrected invoices, see BRM Designing and Generating Invoices.

You use Billing Care or Customer Center to generate corrective bills for an individual bill. When you have corrections for a set of bills or accounts, you use the pin_make_corrective_bills utility to create the corrective bills. See "Billing Accounts By Using the pin_make_corrective_bill Utility".

Note:

You cannot use the BRM billing scripts to run corrective billing.

BRM can generate corrective bills for the following:

- Original bills.
• Bills created by using Bill Now
• On-purchase billing.
• Parent and nonpaying child bills in a bill unit (billinfo object) hierarchy.
• A prior corrective bill. That is, you can correct a subsequent corrective bill to address the errors on a prior corrective bill.

After BRM generates a corrective bill for a prior bill, BRM accepts payments it receives for the regular bill, the corrective bill, or both, but it applies the payments to the latest (original or corrective) bill only. For example, BRM has a regular bill B1-124 for which it generates two corrective bills (CB1-28 and CB1-30) in one bill period. For BRM, CB1-30 is the last bill and B1-124 and CB1-28 are prior bills. By default, all payments that BRM receives for B1-124 or CB1-28 or CB1-30 will be applied against CB1-30.

When configuring how to correct bills, note how corrective billing works with the following features:

• **In-progress bills.** BRM creates a corrective bill for a finalized bill only. If a bill is currently in progress, BRM does not permit a corrective bill to be generated for that bill.

• **Bill unit hierarchy.** When charges are corrected for a bill in a bill unit hierarchy, a corrective bill is made for parent and child bill units. A paying parent bill unit includes the corrective bill for a nonpaying child bill unit. You cannot create a corrective bill for only a nonpaying child bill unit.

• **Rerating accounts that use open item accounting.** If you rerate events in billing cycles other than the current billing cycle for accounts which have open item accounting, you should generate corrective bills for those previous bills. In such a scenario, it is recommended that you generate corrective bills and corrected invoices for all the previous bills that were affected by the rerating. When you generate corrected invoices for such corrective bills, the rerated events are displayed in the corrected invoices. They will not be displayed in the regular bill for the current cycle. If you do not create a corrective bill for the previous bill, these rerated events are included in the regular bill for the current cycle.

• **Discount and charge sharing.** You must rerate accounts following updates to the discount and charge sharing for those accounts in a billed period. BRM includes the resulting adjustments in the corrective bills you generate for the billed period.

• **Collections.** BRM does not initiate payment collection actions for bills that have been replaced by a corrective bill. See “Post-Processing Actions for Corrective Bills” for more information.

• **A/R actions.** Disputes, settlements, and write-offs impact the content of a corrective bill. To prevent the generation of corrective bills for certain A/R actions, set up a custom validation policy for BRM to use when you generate the corrective bills. Such a validation could be used to prevent BRM from generating corrective bills when the prior bill is in dispute and a settlement has not been reached.

• **Adjustments created by rerating.** BRM creates unallocated items for any adjustment items that are created as a result of rerating. If you require that corrective bills should contain the aggregation and allocation of automatic adjustments to the items on the original bill, you must enable the AllocateReratingAdjustments business parameter before you rerate the original bills. See the discussion of rerating in *BRM Implementing Charging* for more information.
Corrective Bills and Billing Cycles

You can have multiple corrections for a single bill. BRM generates corrective bills differently based on whether the corrections occur in the same cycle or in multiple cycles.

When there are multiple corrections in a single bill cycle, BRM corrects the most recent corrective bill for the cycle. BRM includes first-time corrections on the most recent corrective bill and also events or items that were previously corrected and billed on previous corrective bills.

When corrections span multiple billed cycles, BRM uses the accounting method specified in the account to determine the cycle for which to generate the corrective bill.

- **Balance forward.** For accounts using the balance forward accounting method, BRM generates a single corrective bill for the last billed cycle.
- **Open item.** For accounts using the open item accounting method, BRM creates corrective bills for each billed cycle which contains corrections.

For example, an account has corrections in the October bill and the November bill. BRM generates two corrective bills, one for October and one for November. If the October bill was already paid, BRM includes the corrections for the October cycle to the next open bill, which in this case is the bill for December. The corrections for the November cycle are included in the November bill because that bill is still pending payment.

Configuring Corrective Billing

Complete the following to set up the corrective billing process in BRM. Some steps are optional.

- Enable corrective billing in BRM. See "Enabling Corrective Billing".
- Restrict CSR permissions, as necessary. See "Restricting Corrective Billing Permissions".
- Set up a numbering scheme for corrective bill numbers. See "Configuring Bill Numbers for Corrective Bills".
- Create custom reasons for the correction. BRM provides default reasons, but you can add to this list. See "Customizing Correction Reasons".
- Specify the threshold amount for corrective bills. See "Specifying the Minimum Threshold Amount for Corrective Bills".
- Do not permit the processing of payment for prior bills. See "Rejecting Payments for Prior Bills".
- Set up the handling of corrections for partially or fully-paid bills. See "Enabling BRM to Create Corrective Bills for Partially or Fully Paid Bills".

Enabling Corrective Billing

To enable corrective billing, run the `pin_bus_params` utility to change the `EnableCorrectiveInvoices` business parameter. For information on this utility, see *BRM Developer's Guide*.
To enable corrective billing:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, change `disabled` to `enabled`:
   
   ```
   <EnableCorrectiveInvoices>enabled</EnableCorrectiveInvoices>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the file into the BRM database:
   
   ```
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

### Configuring Accounts and Bills for Corrective Billing

When you run the `pin_make_corrective_bills` utility, you can use the command line to specify the bill numbers and account numbers that need corrective bills. In addition, you can include account numbers and bill numbers in a file that the `pin_make_corrective_bills` utility reads.

The default file is `pin_bill_run_control.xml` in the `BRM_home/apps/pin_billd` directory. See "Managing Large Billing Runs" for a sample billing run configuration file.

To verify that the selected accounts or bills are eligible for corrective billing, run the following command:

```
pin_make_corrective_bills --validate_only
```

### Restricting Corrective Billing Permissions

By default, BRM provides all CSRs with permission to create corrective bills.

To restrict the permission for CSR roles:

1. In Permissioning Center, restrict the permissions for the required CSR roles. For more information on setting up permissions, see "Setting Up Permissions in BRM Applications" in `BRM System Administrator's Guide`.
2. Stop and restart the CM.

### Configuring Bill Numbers for Corrective Bills

Corrective billing uses a different bill number format. Regular bills use a bill number such as B1-189. A corrective bill uses a number such as CB1-15. A corrective bill applied to a corrective bill uses the same prefix and the next available sequence; for example, CB1-207.

You can configure BRM to use the same bill number format as non-corrective bills. In this case, BRM retains the original bill's prefix and uses the next available sequence number in BRM. For example, for bill number B1-189, the corrective bill number is set to B1-2007.
For subsequent corrections, BRM retains the prefix from the previous corrective bill and assigns the next available sequence number.

To retain the numbering format you used for regular bills:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the file, change `disabled` to `enabled`:
   ```xml
   <GenerateCorrectiveBillNo>enabled</GenerateCorrectiveBillNo>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see `BRM System Administrator's Guide`.

Customizing Correction Reasons

You can specify correction reasons:

- When you create a corrective bill from Billing Care or Customer Center. You can select a reason for the correction.
- When you run the `pin_make_corrective_bills` utility. You provide the correction reason.

BRM provides the following default reasons for correcting a bill:

- Update to the invoice address
- Manual adjustment
- Price correction

You can provide custom corrections reasons for bill correction by modifying an existing reason code or adding a custom reason code to the `reasons.en_US` file. BRM assigns the IDs of 1, 2, and 3 respectively to these reasons and uses 43 in the version field to specify the domain of the reason.

Here is an example code sample to add a reason code:

```sh
DOMAIN = "Reason Codes-Bill Correction Reasons";
STR
  ID = 25;
  VERSION = 43;
  STRING = "Corrective bill request for Error in Promotion ABC";
END
```

When you add or modify a reason code, ensure that `ID` is a number greater than 3 and the value for `version` is 43 (see the preceding example). These values are reserved by BRM.

Use the `load_localized_strings` utility to load the modified `reasons.en_US` file. For instructions on providing custom reason codes, see "String Manipulation Functions" in
Specifying the Minimum Threshold Amount for Corrective Bills

You can specify a minimum amount that the A/R actions for a bill should reach for BRM to generate a corrective bill. If the bill to be corrected belongs to a bill unit hierarchy, BRM checks that threshold against the sum of A/R actions for all bill units that had charges contributing to that bill.

To specify the minimum threshold amount for corrective bills:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. Change 0 to the required value. For example,
   ```
   <CorrectiveBillThreshold>15</CorrectiveBillThreshold>
   ```
4. Save the file as bus_params_billing.xml.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the pin_multidb script with the -R CONFIG parameter. For more information, see BRM System Administrator's Guide.

Rejecting Payments for Prior Bills

Corrections to bill charges and bill payments may not always be synchronized operations. That is, you may receive payments where the bill number associated with a payment does not match the last bill or you may have received a partial or full payment for that original or corrective bill.

You can reject payments for prior bills, and thereby require them to be specific to the bill. To enable this feature, run the pin_bus_params utility to change the RejectPaymentsForPreviousBill business parameter. For information about this utility, see BRM Developer's Guide.

To reject payments for prior bills and require them to be specific to the bill:

1. Go to BRM_home/sys/data/config.
2. Create an XML file from the /config/business_params object:
   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the file, change disabled to enabled:
   ```
   <RejectPaymentsForPreviousBill>enabled</RejectPaymentsForPreviousBill>
   ```
4. Save the file as bus_params_billing.xml.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

### Enabling BRM to Create Corrective Bills for Partially or Fully Paid Bills

If BRM receives a partial or full payment for a bill, it does not enable a corrective bill to be generated for that bill. Instead, BRM assigns all corrections for that bill period to the next open bill.

To enable BRM to generate a corrective bill for a bill that has a full or partial payment, set the `AllowCorrectivePaidBills` parameter in the `billing` instance of the `/config/business_params` object to `enabled`. BRM then generates the corrective bill, and the balance on the corrective bill reflects the payment that was received against the prior bill. The corrective bill also displays the payment details for payments already processed against the prior bill.

To enable BRM to generate corrective bills for fully paid or partially paid bills:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the file, change `disabled` to `enabled`:
   ```xml
   <AllowCorrectivePaidBills>enabled</AllowCorrectivePaidBills>
   ```
4. Save the file as `bus_params_billing.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_billing.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator's Guide*.

### Customizing Corrective Bills

See BRM Opcode for information about how to customize corrective bills. You can do the following:

- **Customize due dates.** You can apply different due dates to corrective bills by customizing the bill due date calculations. Modify the `PCM_OP_BILL_POL_CALC_PYMT_DUE_T` policy opcode to retain the same due date as in the prior bill, or add a specific number of offset days.

- **Change the default invoice type.** Use the `PCM_OP_CUST_POL_PREP_PAYINFO` policy opcode to override the default value (detailed replacement invoice).

- **Define custom validations.** To define custom validations for corrective billing, modify the existing validations in the `PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL` policy opcode or add custom validations.
Billing Accounts By Using the pin_make_corrective_bill Utility

After you have configured corrective billing (see "Configuring Corrective Billing"), you are ready to generate corrective bills.

You can generate corrective bills in one of the following ways:

- Allocate adjustments to a selected bill in Billing Care or Customer Center, and then submit the bill for corrective billing.
- Run the **pin_make_corrective_bills** utility which generates the corrective bill.

The **pin_make_corrective_bills** utility calculates the balance due after allocating the adjustments and A/R actions for each account bill unit, and creates a corrective bill for the balance due. It creates corrective bills for bill units whose bills have corrections that fall within the period you specify.

The balance due amount is the amount requested as a payment by the **pin_collect** utility and the amount that is shown on the corrective invoice.

For information about the **pin_make_corrective_bills** utility syntax, see "pin_make_corrective_bills".

Run the **pin_make_corrective_bills** utility when you have allocated all adjustments to prior bills.

When you generate corrective bills for a bill unit hierarchy, you must run the **pin_make_corrective_bills** utility for the top-most parent bill unit.

You must run the **pin_make_corrective_bills** utility before you run **pin_collect** because **pin_collect** needs the balance due amount collected by the **pin_make_corrective_bill** utility.

After you begin the process of generating a corrective bill, you cannot cancel the process.

To generate a corrective bill using **pin_make_corrective_bills**:

1. Go to `BRM_home\apps\pin_billd`.
2. Run the **pin_make_corrective_bills** utility:

   ```
   pin_make_corrective_bills
   ```

   For example, you can:
   - Use the **-corrective_inv_type** parameter to specify the invoice type (replacement or invoice correction letter).
   - Use the **-threshold_amount** parameter to override the threshold amount for generating a corrective bill
   - Use the **-account_no** parameter to specify the accounts to create corrective bills for.
   - Use the **-create_if_no_corrections** parameter to generate a corrective bill when there are only simple changes such as invoice address changes. Use this parameter with the **-corrective_inv_type R** parameters.
3. Verify the contents of the corrective bills by using Billing Care or Customer Center.

Post-Processing Actions for Corrective Bills

After you generate corrective bills, you can do the following:

- Generate corrective invoices by running the `pin_inv_accts` utility. You can generate summary or detailed corrective invoices for the corrective bills you generated. See "pin_inv_accts" in BRM Designing and Generating Invoices.

- Export corrective invoices for use with custom programs, (such as DOC1) to generate and publish the corrective invoice documents. Use the `pin_inv_export` utility to export invoices to a format you can use with other programs, such as DOC1. See "pin_inv_export" in BRM Designing and Generating Invoices.

- Collect any balance due for accounts that use credit card and direct debit payment methods by running the `pin_collect` utility. See "pin_collect" in BRM Configuring and Collecting Payments.

- Check whether corrective billing affected the state of any prior bills that were in collections by running the `pin_collections_process` utility. For more information, see "pin_collections_process" in BRM Collections Manager.

Corrective Billing for Disputes, Settlements and Write-offs

Disputes, settlements, and write-offs impact the content of a corrective bill. If you must prevent the generation of corrective bills for certain A/R actions, set up a custom validation policy for BRM to use when you generate the corrective bills. Such a validation could be used to prevent BRM from generating corrective bills when the prior bill is in dispute and a settlement has not been reached.

Disputes

Corrective bills can be generated for a dispute entered at the bill level, item level or event level. When BRM generates the corrective bill for a prior bill in dispute, the corrective bill contains the details of the dispute, the disputed bill item under the corrected items, and the bill balance reduced by the disputed amount.

Settlements

Corrective bills can be generated for a dispute settled at the bill level, item level or event level. When BRM generates the corrective bill when a bill item on the prior bill is in settlement, the corrective bill contains the details of the settlement, the settled bill item under the corrected items, and the bill balance reduced or increased by the settlement.

A dispute may be settled for the full amount of the dispute in the customer's favor. If you submitted a corrective bill at the time of such a dispute, the corrective bill at settlement time may become a duplicate as there may not be any additional corrections to the second bill.
Write-Offs

You perform write-offs after the account is past due and considered delinquent in collections. When BRM generates the corrective bill for a write-off on a bill item on the prior bill, the corrective bill contains the details of the write-off, the written-off bill item under the corrected items, and the bill balance reduced by the write-off amount.
Running Trial Billing

This document provides an overview of Oracle Communications Billing and Revenue Management (BRM) trial billing and explains when and how to run trial billing.

See also:

- List of Billing Features
- BRM Concepts

About Trial Billing

*Trial billing* is a process that simulates BRM billing. You use trial billing to validate billing results without billing customers.

The trial billing utility simulates the billing functions of the `pin_bill_accts` utility. It can additionally create and store trial invoices in the BRM database.

You can also collect revenue assurance data for trial bills and display the data by generating Revenue Assurance Billing reports.

In case of billing discrepancies, you can make corrections, such as adjustments or payment allocations, or you can run rerating. When you are satisfied with the results from trial billing, run actual billing to generate the final bills.

To perform trial billing, you run the "`pin_trial_bill_accts`" utility.

Table 18-1 summarizes the similarities and differences between regular billing and trial billing:

<table>
<thead>
<tr>
<th>Billing with <code>pin_bill_accts</code></th>
<th>Trial Billing with <code>pin_trial_bill_accts</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Performs billing on an account's billing day of month.</td>
<td>Performs trial billing before or after an account's billing day of month.</td>
</tr>
<tr>
<td>Creates cycle forward, usage item, and bill objects in the BRM database.</td>
<td>Creates cycle forward and usage item objects, but the objects are not recorded in the BRM database.</td>
</tr>
<tr>
<td>Calculates and totals the balance impacts for the previous billing cycle.</td>
<td>Calculates and totals the balance impacts for any billing cycle.</td>
</tr>
<tr>
<td>Calculates and updates account balances.</td>
<td>Calculates but does not update account balances.</td>
</tr>
<tr>
<td>Does not create invoices.</td>
<td>Creates and stores trial invoices in the BRM database unless you specify not to.</td>
</tr>
<tr>
<td>Balance due in the bill object shows the exact amount that is due.</td>
<td>Balance due in the trial invoices may not show the exact amount due at time of actual billing. This is because events that have a balance impact can occur after you run trial billing.</td>
</tr>
</tbody>
</table>
Note:

With regular billing, invoices are created separately by running the `pin_inv_accts` utility. The trial billing utility simulates billing and additionally creates and stores trial invoices. For this reason, a trial billing run that generates invoices takes longer than actual billing. You have the option to not generate trial invoices if you do not need them. See "Creating Trial Bills without Generating Trial Invoices".

About Trial Billing for Bill Unit Hierarchies and Sharing Groups

Bill unit hierarchies and sharing groups enable customers to pay other customers' bills. Bill unit hierarchies form a parent-child relationship; paying parent bill units and their nonpaying child bill units are on the same billing cycle. Trial billing for bill unit hierarchies is handled differently from regular billing.

In regular billing, charges for nonpaying child bill units are calculated first and then rolled up to the paying parent bill unit. For each nonpaying bill unit, BRM creates a database lock, computes the balance impacts, and then releases the lock.

In trial billing, the paying parent and all its nonpaying child bill units are locked simultaneously. The parent bill unit and the nonpaying child bill units remain locked until the balance impacts of all nonpaying bill units are computed and rolled up to the parent bill unit.

Note:

As with regular billing, the invoicing threshold value for bill unit hierarchies is considered during trial billing. If the threshold value is exceeded (that is, the number of nonpaying bill units is greater than the threshold value), a separate invoice is generated for each nonpaying bill unit by using multiple threads, and the invoice for the paying bill unit does not contain the data.

Note:

When hierarchical bill units are selected for trial billing, access to the parent and child bill units is prevented and system performance is decreased as a result of the database locks.

With sharing groups, sharing occurs when an account is added to a sharing group and a is added to that account. The sharing group owner account and member account can have different billing cycles. When a sharing group owner is selected for trial billing, the trial invoice might not contain all the fees from member accounts.

In regular billing, when a sharing group owner account is billed, BRM processes the member accounts to determine the sponsored fees to apply to the owner account. The sponsored cycle events of member accounts are accumulated in the `item/sponsor` object of the owner account and appear on the owner's bill.
In trial billing, when a sharing group owner account is selected, trial billing is not run for member accounts. In this case, the owner’s trial invoice might not contain the recurring charges for the sponsored cycle events of the member accounts because such charges are generated by running billing. The owner's trial invoice would, however, contain the members’ usage charges because those charges are not generated by billing.

About Trial Invoices

A trial invoice is a statement of charges and the balance that is due. You use trial invoices to validate billing charges before creating final bills for your customers. Trial invoices are generated when you run the trial billing utility (`pin_trial_bill_accts`). For information about creating trial invoices, see "Running Trial Billing".

**Note:**

- Trial invoices are created only for billing cycles that have not been billed yet.
- Trial invoices are optional. If you do not need trial invoices, you can specify not to generate them: for example, when the revenue assurance data collected from trial billing is sufficient for validating the billing results.

**Note:**

Session or activity charges incurred between the dates of actual billing and trial billing do not appear on trial invoices.

You can design your own invoice templates for displaying trial invoices, and you can customize the information displayed on trial invoices in the same ways you do for regular invoices.

You can also purge and export trial invoices. See "Purging Trial Invoices" and "Exporting Trial Invoices".

**Note:**

Trial invoices are always stored in the primary BRM database schema. In a multischema system, regular invoices can be stored in a separate schema.

About Collecting Revenue Assurance Data from Trial Billing

When you enable trial billing to generate revenue assurance data, the data is automatically generated when you run the `pin_trial_bill_accts` utility. You can use this
data to validate overall billing results. To view the data, run a Revenue Assurance Billing Summary report.

You enable trial billing to generate revenue assurance data by setting an entry in the trial billing configuration file \texttt{(BRM\_home/apps/pin\_trial\_bill/pin.conf)}. For more information, see "Enabling Billing Utilities to Generate Revenue Assurance Data" in \textit{BRM Collecting Revenue Assurance Data}.

You can also split revenue assurance data collected from trial billing into more detailed categories by running \texttt{pin\_trial\_bill\_accts} with the \texttt{-split} parameter. The split data is based on the type of billable item and its associated services. To view split revenue assurance data, run a Revenue Assurance Billing Detail report.

You can customize how revenue assurance data is split by configuring item subtypes for the type of revenue, such as one-time usage, recurring charges, adjustments, and payments.

For information about revenue assurance data, see "About Collecting Revenue Assurance Data from Billing" in \textit{BRM Collecting Revenue Assurance Data}.

\section*{Configuring Trial Billing}

You can configure the following options for trial billing:

- Specifying Accounts for Trial Billing
- Specifying Bill Units, Billing Segments, and DOMs for Trial Billing
- Creating Trial Bills without Generating Trial Invoices

\subsection*{Specifying Accounts for Trial Billing}

To run trial billing for a small number of random accounts, set the \texttt{pin\_trial\_bill\_accts} flag in the Connection Manager (CM) configuration file \texttt{(BRM\_home/sys/cm/pin.conf)} to the maximum number of accounts to bill:

\begin{verbatim}
-pin_trial_bill_accts threshold numberOfAccounts
\end{verbatim}

To specify a list of specific accounts to be trial billed, use the \texttt{-f} parameter:

\begin{verbatim}
pin_trial_bill_accts -end 4/1/2002 -f inputFile
\end{verbatim}

\textbf{Note:}

You can also use the \texttt{-f\_control} parameter as an alternative way of specifying accounts for trial billing. See "Specifying Bill Units, Billing Segments, and DOMs for Trial Billing".

The input file lists the account POIDs and bill unit POIDs in flist array format. The flist must use the following structure, where each array element corresponds to one account and bill unit.
• Trial billing stops and reports a warning message when it encounters an account or bill unit with inactive status.
• All accounts listed in the input file will be trial billed.
• If a selected account has a nonpaying bill unit, pin_trial_bill_accts does not create a trial bill for it. Nonpaying bill units are trial billed only when their parent accounts are selected for trial billing.
• If an account that has not been billed for some time is selected for trial billing, a trial invoice is generated for each billing cycle that was skipped. The invoice data is stored in shared memory until all the billing cycles have been processed. If the account skipped many billing cycles, this could result in a PIN_ERR_NO_MEM error, indicating that the system does not have enough shared memory to process that account.

Specifying Bill Units, Billing Segments, and DOMs for Trial Billing

You can run trial billing for bill units associated with specified accounting DOMs, specified billing segments, or a combination of DOMs and segments. You list the DOMs and billing segments in a trial-billing-run configuration file. You can also use this file to specify a list of accounts and bill units for trial billing.

You create a trial-billing-run configuration file by copying and modifying the BRM billing run configuration file (BRM_home/apps/pin_billd/pin_bill_run_control.xml). You then specify this file when you run pin_trial_bill_accts with the -f_control parameter.

To specify accounts and bill units, DOMs, or billing segments for trial billing:

1. Open the pin_bill_run_control.xml file in an XML editor or a text editor. By default, the file is in the BRM_home/apps/pin_billd directory.
2. Edit the file to specify the bill units to trial bill:
   • To specify accounts and their bill units, see “Specifying Bill Units for Trial Billing”.
   • To specify accounting DOMs, see “Specifying Accounting DOMs for Trial Billing”.
   • To specify billing segments, see “Specifying Billing Segments for Trial Billing”.  

Note:

• Using a trial-billing-run configuration file to specify accounts for trial billing is an alternative way of specifying accounts in flist format by using the -f parameter. (See "Specifying Accounts for Trial Billing".)
• The pin_bill_run_control.xml file is the same file used to split a regular billing run into multiple runs. (See "Configuring a Split Billing Run".)
3. Save the file under a different name and close it. Give the file a meaningful name: for example, if trial billing a group of accounts, include the account range or reason for billing in the file name; if trial billing specific DOMs, include the DOM range in the file name.

Note:
When you run `pin_trial_bill_accts`, the configuration file you create and the default `BRM_home/apps/pin_billd/business_configuration.xsd` file must be in the same directory.

4. Run `pin_trial_bill_accts` with the `-f_control` parameter:

```
pin_trial_bill_accts -end 4/1/2002 -f_control filename
```

where filename is the name of the trial-billing-run configuration file that you created.

Note:
- If filename is in the same directory from which you run `pin_trial_bill_accts`, specify only the file name.
- If filename is in a different directory from which you run `pin_trial_bill_accts`, you must include the entire path for the file.

For more information, see "pin_trial_bill_accts".

Specifying Bill Units for Trial Billing

To generate trial bills for a set of accounts and bill units, add a `BillingList` parent element in the trial-billing-run configuration file for each account and bill unit to include. In the `BillingList` parent element, add an `Account` child element that specifies the POID ID of the account, and add a `Billinfo` child element that specifies the POID ID of the bill unit.

Note:
- Trial billing stops and reports a warning message when it encounters an account or bill unit with inactive status.
- To trial bill specific accounts and bill units, you must include both the account POID and bill unit POID. If only one is specified, the account or bill unit is not trial billed.

For example, the following `BillingList` parent element generates trial bills only for the bill unit with POID ID 64295 that belongs to the account with POID ID 17763:

```
<BillingList>
  <Account>17763</Account>
</BillingList>
```
To specify multiple accounts or multiple bill units from the same account, add a **BillingList** parent element for each bill unit. For example, the following entries generate two trial bills for the account with POID ID 17763 and one trial bill for the account with POID ID 25147:

```
<BillingList>
  <Account>17763</Account>
  <Billinfo>64295</Billinfo>
</BillingList>

<BillingList>
  <Account>17763</Account>
  <Billinfo>68439</Billinfo>
</BillingList>

<BillingList>
  <Account>25147</Account>
  <Billinfo>314552</Billinfo>
</BillingList>
```

**Specifying Accounting DOMs for Trial Billing**

To generate trial bills for bill units associated with specified accounting DOMs, add a **DOMList** parent element in the trial-billing-run configuration file. In the **DOMList** parent element, add a **DOM** child element that specifies the accounting DOM for each day whose bill units you want to include.

For example, the following **DOMList** parent element generates trial bills only for bill units whose accounting DOM is 1 or 15:

```
<DOMList>
  <DOM>---01</DOM>
  <DOM>---15</DOM>
</DOMList>
```

If the **DOMList** parent element is omitted, bill units associated with any accounting DOM can be included in the billing run.

**Specifying Billing Segments for Trial Billing**

To run trial billing for specified billing segments, you must first set up billing segments in your system and then associate them with bill units. For more information, see "Load Balancing Billing Runs".

To generate trial bills for bill units associated with specified billing segments, add a **BillSegmentList** parent element in the trial-billing-run configuration file. In the **BillSegmentList** parent element, add a **BillSegment** child element that specifies the billing segment ID for each billing segment whose bill units you want to include. The segment ID is the ID of any billing segment defined in the `/config/billing_segment` object in your BRM system.

For example, the following **BillSegmentList** parent element generates trial bills only for bill units associated with billing segments 101, 102, and 103:

```
<BillSegmentList>
  <BillSegment>101</BillSegment>
  <BillSegment>102</BillSegment>
</BillSegmentList>
```
Running Trial Billing

To run trial billing, use the "pin_trial_bill_accts" utility:

1. Go to BRM_home/apps/pin_trial_bill.
2. Run the pin_trial_bill_accts utility:

   ```
   pin_trial_bill_accts
   ```

   Running the utility with no parameters runs trial billing for all accounts that are due for billing. You can run trial billing for a subset of accounts. For example, you can:

   * Use the -start and -end parameters to run trial billing for bills in a specified date range. See "Running Trial Billing With Date Ranges".
   * Use the -pay_type parameter to run trial billing for a specific payment type.
   * Use the -bill_only to run trial billing without generating trial invoices and collect revenue assurance data.
   * Use the -split parameter to generate detailed revenue assurance data.
   * Use the -active, -inactive, and -closed parameters to run trial billing for a specific account status.

   See "pin_trial_bill_accts".

Running Trial Billing With Date Ranges

The start and end dates determine which accounts are selected for trial billing and which billing cycles for those accounts are trial billed.

The end date is used as the search criteria for retrieving accounts for trial billing. The search selects all accounts with a billing date less than the end date. You can specify either an absolute date, a number of days before or after the current date, or the current date (by specifying 0).

---

**Note:**

If you do not specify an end date, pin_trial_bill_accts uses the current date for the end date.

---

The start date determines the billing cycles for which trial invoices are generated. You can specify either an absolute date, a number of days before or after the current date, or the current date (by specifying 0).

If you specify a start date, trial billing is run only when a billing cycle is completed between the start date and the end date. Trial billing is not run on partial cycles.
If you do not specify a start date or if you specify 0 for the current date, `pin_trial_bill_accts` generates trial invoices for all billing cycles that were completed between the current date and the end date and that have not already been billed. More than one trial invoice might be generated for accounts that have not been billed for one or more billing cycles.

Examples:

- Create trial invoices for accounts whose billing date is on or before 4/1/2002 (current date is 3/15/2002):
  
  ```
  pin_trial_bill_accts -end 04/01/2002
  pin_trial_bill_accts -end +17
  ```

- Create trial invoices for accounts whose billing date is on or before 3/1/2002 (current date is 3/15/2002):
  
  ```
  pin_trial_bill_accts -end 03/01/2002
  pin_trial_bill_accts -end -14
  ```

- Create trial invoices for accounts whose billing date is on or before the current date:
  
  ```
  pin_trial_bill_accts -end 0
  pin_trial_bill_accts
  ```

- Create trial invoices for accounts with complete billing cycles between 4/1/2002 and 5/15/2002 (current date is 5/15/2002):
  
  ```
  pin_trial_bill_accts -start 04/01/2002 -end 5/15/2002
  pin_trial_bill_accts -start -44 -end 0
  ```

Running Trial Billing According to Payment Type

Generates trial invoices for accounts with the specified payment method ID. If you do not specify the payment method ID, the `pin_trial_bill_accts` utility generates trial invoices for all payment methods.

The ID can be any payment method ID from Table 18-2.

### Table 18-2 Payment Method ID

<table>
<thead>
<tr>
<th>Payment Method</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>credit card</td>
<td>10003</td>
</tr>
<tr>
<td>debit card</td>
<td>10002</td>
</tr>
<tr>
<td>direct debit</td>
<td>10005</td>
</tr>
<tr>
<td>guest</td>
<td>10010</td>
</tr>
<tr>
<td>invoice</td>
<td>10001</td>
</tr>
<tr>
<td>prepaid</td>
<td>10000</td>
</tr>
</tbody>
</table>
Table 18-2  (Cont.) Payment Method ID

<table>
<thead>
<tr>
<th>Payment Method</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEPA</td>
<td>10018</td>
</tr>
<tr>
<td>subordinate</td>
<td>10007</td>
</tr>
<tr>
<td>undefined</td>
<td>0</td>
</tr>
</tbody>
</table>

For example, to generate trial invoices for all bills paid by the credit card payment method, use the following syntax:

```
pin_trial_bill_accts -pay_type 10003
```

For hierarchical bill units, you must run nonpaying child bill units before running paying parent bill units. For example, to generate trial invoices for hierarchical bill units whose payment method is by check, use the following syntax:

```
pin_trial_bill_accts -pay_type 10007
pin_trial_bill_accts -pay_type 10012
```

Note these limitations:

- The `-active`, `-closed`, or `-inactive` parameters are not supported for nonpaying child bill units when combined with the `-pay_type` parameter. These parameters limit the number of nonpaying child bill units that are invoiced, which leads to errors when their paying parent bill unit is trial billed.
- Do not set a threshold value. Setting a threshold value limits the number of nonpaying child bill units that are invoiced, which leads to errors when their paying parent bill unit is trial billed.

**Note:**

The threshold entry is located in the `BRM_home/apps/pin_trial_bill/pin.conf` file.

- The `-f`, `-f_control`, and `-bill_only` parameters are not supported for all bill units when used with the `-pay_type` parameter.

Creating Trial Bills without Generating Trial Invoices

If the revenue assurance data collected from trial billing provides enough information for you to validate your billing charges and you do not need the specific information provided in invoices, you can run the trial billing utility with the `-bill_only` parameter. This parameter suppresses the creation of trial invoices. Generating trial invoices takes longer than regular billing; therefore, running trial billing without generating trial invoices improves the performance of trial billing and is equivalent in performance to running regular billing.
Purging Trial Invoices

To purge trial invoices, run the `pin_trial_bill_purge` utility.

By default, `pin_trial_bill_purge` purges invoices with the `/invoice/trial` POID type. If a custom `/invoice` subclass exists, invoices with `/invoice/custom_subclass/trial` POID type are purged.

Note:

If the `/invoice` class is partitioned, running the purge utility (`partition_utils -o`) on `/invoice` purges both regular invoices and trial invoices.

For a description of the syntax and parameters of this utility, see "pin_trial_bill_purge".

For more information about trial invoices, see "About Trial Invoices".

Exporting Trial Invoices

To export trial invoices, run the `pin_inv_export` utility and use the `-trial` parameter.

You can export trial invoices in the same formats that are supported for regular invoices. To export trial invoices, follow the same procedures you use for exporting regular invoices. For more information, see "Exporting Invoices" in *BRM Designing and Generating Invoices*.

For a description of the syntax and parameters of this utility, see "pin_inv_export" in *BRM Designing and Generating Invoices*. 


Suspending Billing

This document describes how to suspend billing for selected accounts in Oracle Communications Billing and Revenue Management (BRM).

See also:
- List of Billing Features
- BRM Concepts

About Suspending Billing of Accounts and Bills

By default, BRM generates bills for all bill units in all accounts. When you run billing in BRM, active accounts as well as inactive and closed accounts are billed. However, you may have accounts or bill units that you do not want to bill. You can suspend billing for those accounts and bill units. You can later resume spending on suspended accounts and bill units.

Note:
By excluding accounts or bill units that do not need to be billed, you can reduce the time it takes to complete your billing run.

Note:
- If you suspend an account, all nonpaying child bill units of the suspended account's paying parent bill units are also suspended.
- To suppress billing, see "About Suppressing Bills". Unlike bill and account suspension, bill suppression does not inactivate bill units.
- For information about another way to reduce the duration of your billing run, see "Splitting a Billing Run into Multiple Runs".

Suspending Billing of Closed Accounts

By default, you can configure BRM to enable or disable billing of closed accounts. When billing of closed accounts is disabled, BRM excludes closed accounts from the billing run when the following conditions are met:
- The account's total balance due for every bill unit is zero.
Note:
The total balance due includes the totals from the nonpaying child bill units of an account's paying parent bill units.

- The account has had no billable activity since the previous bill.

To configure billing of closed accounts:

1. Open the CM configuration file (`BRM_home/sys/cm/pin.conf`) in a text editor.
2. Do one of the following:
   - To disable billing of closed accounts, set the value of `stop_bill_closed_accounts` entry to 1:
     ```
     stop_bill_closed_accounts 1
     ```
   - To enable billing of closed accounts, set the value of `stop_bill_closed_accounts` to 0.
     By default, billing of closed accounts is enabled.
3. Save the file.
4. Stop and restart the CM.

Suspending Billing of an Account's Bill

If you have additional bill units in accounts that you do not want to bill, such as inactive bill units, you can customize BRM to suspend billing of those bill units.
Setting Up Billing to Run in a Multischema Environment

You can run billing in a multischema BRM environment on one database schema at a time by using one instance of the billing utilities or on multiple schemas simultaneously by using multiple instances of the billing utilities. In other words, to bill accounts on a specific database schema, you must run the billing utilities from that database schema.

For instance, to bill accounts that reside on the primary and secondary database schemas, you can run the billing utilities from the primary schema to bill the accounts that reside on the primary database schema and run another instance of the billing utilities from the secondary schema to bill the accounts that reside on the secondary database schema.

Note:

When running the `pin_bill_day` script with the `-file` option, ensure that the accounts specified in the billing run configuration file reside on the same database schema where `pin_bill_day` is run. If the file contains accounts from different database schemas, `pin_bill_day` reports an error.

Running Billing on One Schema at a Time

Running billing utilities on multiple database schemas one at a time requires that you edit the billing utility configuration file before each time you run the billing utilities. Perform the following steps before each time you run billing:

1. Open the billing utility configuration file (`BRM_home/apps/pin_billd/pin.conf`) in a text editor.
2. Change the value of the `userid` entry to the database schema against which you want to run billing. For example, to run billing on the 0.0.0.2 schema, change the `userid` entry as follows:
   - `- userid 0.0.0.2 /service/pcm_client 1`
3. Change the value of the `login_name` entry to an account that resides in the
schema against which you want to run billing. For example, to run billing using the
account `root.0.0.0.2`, change the `login_name` entry as follows:

   `nap login_name root.0.0.0.2`

4. Save the file.

5. Run the billing utilities.

Running Billing on Multiple Schemas Simultaneously

Running billing on multiple database schemas simultaneously requires that you create
parallel instances of the billing utility configuration files, each of which is configured for
a particular schema. Then you run all instances of your billing utilities.

1. For each schema that you want to run billing on, create a subdirectory under
`BRM_home/apps/pin_billd`.

   For example, `BRM_home/apps/pin_billd/db1` for database 1, `BRM_home/apps/pin_billd/db2` for database 2, and so on.

2. Copy the `BRM_home/apps/pin_billd/pin.conf` file into each new subdirectory.

3. In each billing subdirectory, do the following:
   a. Open the `pin.conf` file.
   b. Change the database number in the `login_name` entry to a database account
      that resides in the schema against which you want to run billing. For example,
      to run billing against schema `0.0.0.2`, change the `login_name` entry as follows:

      `nap login_name root.0.0.0.2`

   c. Save the file.

4. Run the billing utilities from the new subdirectories.
21
Billing Utilities

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

Topics in this document:

- load_config_item_tags
- load_config_item_types
- load_pin_bill_suppression
- load_pin_billing_segment
- load_pin_calendar
- load_pin_payment_term
- pin_bill_accts
- pin_cycle_fees
- pin_make_corrective_bills
- pin_rollover
- pin_trial_bill_accts
- pin_trial_bill_purge

load_config_item_tags

Use this utility to load item tags into the BRM database when you create custom bill items. See “Creating Custom Bill Items”.

Location

BRM_home/bin

Syntax

load_config_item_tags [-v] [-d] [-h] config_item_tags_file

Parameters

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

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

```
load_config_item_tags 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.

### config_item_tags_file

The name and location of the file that contains your item tag definitions. The sample *config_item_tags.xml* file is in the *BRM_home/sys/data/pricing/example* directory.

### Results

The `load_config_item_tags` utility notifies you when it successfully creates the `/config/item_tags` object.

If the `load_config_item_tags` utility does not notify you that it was successful, look in the utility log file (*default.pinlog*) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

---

**load_config_item_types**

Use this utility to load bill item tag-to-bill item type mappings into the BRM database when creating custom bill items. See "Creating Custom Bill Items"

### Location

*BRM_home/bin*

### Syntax

```
load_config_item_types [-v] [-d] [-h] config_item_types_file
```

### Parameters

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

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

`load_config_item_types 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.

**config_item_types_file**
The name and location of the file that contains your custom item definitions. You can use the sample file, `BRM_home/sys/data/pricing/example/config_item_types.xml`.

**Results**
The `load_config_item_types` utility notifies you when it successfully creates the `/config/item_types` object.

If the `load_config_item_types` utility does not notify you that it was successful, look in the utility log file (`default.pinlog`) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

**load_pin_bill_suppression**

Use this utility to load bill suppression information for customer segments into the BRM database. See the following topics:

- Associating Bill Suppression Information with Customer Segments
- Editing the Bill Suppression Configuration File

**Location**

`BRM_home/bin`

**Syntax**

`load_pin_bill_suppression -t | -v | -d | -h filename`

**Parameters**

-t
Runs the utility in test mode to validate the XML file against its schema definition (see "Validating Your Bill Suppression Configuration File Edits"). This option does not load data into the `/config/suppression` object or overwrite any existing data in the object.
To avoid load errors based on XML content problems, run the utility with this option before loading data into the object.

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

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

```
load_pin_bill_suppression any_other_parameter -v > filename.log
```

-d
Creates a log file for debugging purposes. If no log file name is specified in the utility’s pin.conf file, names the file default.pinlog. Use this parameter for debugging when the utility seems 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.

FILENAME
The name and location of the bill suppression configuration file. The default file is BRM_homelsys/data/config/pin_bill_suppression.xml, but the utility can take any XML file name as a parameter if the file's contents conform to the appropriate schema definition. See "Validating Your Bill Suppression Configuration File Edits".

If you copy filename to the same directory from which you run the load utility, specify only the file name. If you run the command in a different directory from where filename is located, you must include the entire path for the file.

In addition, filename must be in the same directory as the default BRM_homelsys/data/config/business_configuration.xsd file.

Results
This utility notifies you only if it encounters errors. Look in the default.pinlog file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the bill suppression information was loaded, display the /config/suppression object by using one of the following features:

- Object Browser
- robj command with the testnap utility

For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in BRM Developer's Guide.
load_pin_billing_segment

Use this utility to load billing segment definitions into the BRM database. See "Setting Up Billing Segments".

Location

BRM_home/bin

Syntax

load_pin_billing_segment [-t] [-v] [-d] [-h] filename

Parameters

- **-t**
  Runs the utility in test mode to validate the XML file against its schema definition (see "Validating Your Billing Segment Configuration File Edits"). This option does not load data into the /config/billing_segment object or overwrite any existing data in the object.

  - Note: To avoid load errors based on XML content problems, run the utility with this option before loading data into the object.

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

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

- **-d**
  Creates a log file for debugging purposes. If no log file name is specified in the utility's pin.conf file, names the file default.pinlog. Use this parameter for debugging when the utility seems 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.

filename
The name and location of the billing segment configuration file. The default file is `BRM_home/sys/data/config/pin_billing_segment.xml`, but the utility can take any XML file name as a parameter if the file's contents conform to the appropriate schema definition. See "Validating Your Billing Segment Configuration File Edits".
If you copy `filename` to the same directory from which you run the load utility, specify only the file name. If you run the command in a different directory from where `filename` is located, you must include the entire path for the file.
In addition, `filename` must be in the same directory as the default `BRM_home/sys/data/config/business_configuration.xsd` file.

Results
This utility notifies you only if it encounters errors. Look in the `default.pinlog` file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.
To verify that the billing segment definitions were loaded, display the `/config/billing_segment` object by using one of the following features:
- Object Browser
- `robj` command with the `testnap` utility
For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in `BRM Developer's Guide`.

load_pin_calendar
Use this utility to load billing calendars into the BRM database. See “Editing the Billing Calendar Configuration File ”.

Location
`BRM_home/bin`

Syntax
`load_pin_calendar [\-t] [\-d] [\-v] [\-h] filename`

Parameters
-t
Runs the utility in test mode to validate the XML file against its schema definition (see "Validating Your Billing Calendar Configuration File Edits"). This option does not load data into `/config/calendar` objects or overwrite any existing data in the objects.

Note:
To avoid load errors based on XML content problems, run the utility with this option before loading data into the object.
-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility seems to have run with no errors but the data has not been loaded into the database.

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

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

```plaintext
load_pin_calendar any_other_parameter -v > filename.log
```

-h
Displays the syntax and parameters for this utility.

`filename`
The name and location of the billing calendar configuration file. The default file is `BRM_home/sys/data/config/pin_calendar.xml`, but the utility can take any XML file name as a parameter if the file's contents conform to the appropriate schema definition. See "Validating Your Billing Calendar Configuration File Edits". If you copy `filename` to the same directory from which you run the load utility, specify only the file name. If you run the command in a different directory from where `filename` is located, you must include the entire path for the file. In addition, `filename` must be in the same directory as the default `BRM_home/sys/data/config/business_configuration.xsd` file.

Results
If the utility does not notify you that it was successful, look in the `default.pinlog` file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

**load_pin_payment_term**

Use this utility to load payment terms into the BRM database. See "Editing the Payment Terms Configuration File ".

**Location**
`BRM_home/bin`

**Syntax**

```plaintext
load_pin_payment_term [-t] [-h] [-v] [-d] filename
```
Parameters

-t
Runs the utility in test mode to validate the XML file against its schema definition. This option does not load data into the config/payment_term object or overwrite any existing data in the object.

Note:
To avoid load errors based on XML content problems, run the utility with this option before loading data into the object.

-h
Displays the syntax and parameters for this utility.

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

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

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

load_pin_payment_term any_other_parameter -v > filename.log

filename
The name and location of the payment terms configuration file. The default file is BRM_home/sys/data/config/pin_payment_term.xml, but the utility can take any XML file name as a parameter if the file’s contents conform to the appropriate schema definition.

If you copy filename to the same directory from which you run the load utility, specify only the file name. If you run the command in a different directory from where filename is located, you must include the entire path for the file.

In addition, filename must be in the same directory as the default BRM_home/sys/data/config/business_configuration.xsd file.

Results
If the utility does not notify you that it was successful, look in the default.pinlog file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.
pin_bill_accts

Use this utility as part of your daily billing to calculate the balance due for each account and to create a bill for the balance due. See "Billing Accounts By Using the pin_bill_accts Utility".

Location

BRM_home/bin

Syntax

pin_bill_accts [-active|-inactive|-close]
[pay_type ID]
[ -end mm/dd/yy | numberOfDays]
[-sponsorship | -discount]
[-retry]
[-cycle_charge_only] [-finalize_bill]
[-check_for_subords]
[-file billing_run_config_file_name]
[-split]
[-test]
[-verbose]
[-help]

Parameters

-active|-inactive|-close
Specifies the status of the accounts to be billed.

-pay_type ID
Calculates the balance due for accounts with the specified payment method. You specify the payment method by using the payment method ID, such as 10007 for nonpaying (subordinate) accounts, as shown in Table 21-1.

Note:

If you use -pay_type 10007 Subordinate, you must run the pin_bill_accts utility twice. The first time you run the utility with this parameter to calculate the child account's balance, and the second time you run the utility without the parameter to roll up the balance due to the parent account and then bill the parent account.

This example creates bills by using -pay_type 10007 for nonpaying (subordinate) accounts:

<table>
<thead>
<tr>
<th>Payment Method</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>credit card</td>
<td>10003</td>
</tr>
<tr>
<td>debit card</td>
<td>10002</td>
</tr>
</tbody>
</table>
### Payment Method Table

<table>
<thead>
<tr>
<th>Payment Method</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct debit</td>
<td>10005</td>
</tr>
<tr>
<td>guest</td>
<td>10010</td>
</tr>
<tr>
<td>invoice</td>
<td>10001</td>
</tr>
<tr>
<td>prepaid</td>
<td>10000</td>
</tr>
<tr>
<td>SEPA</td>
<td>10018</td>
</tr>
<tr>
<td>subordinate</td>
<td>10007</td>
</tr>
<tr>
<td>undefined</td>
<td>0</td>
</tr>
</tbody>
</table>

- `mm/dd/yy` or `numberOfDays`:
  - Specifies the due date for accounts to be billed. You can specify a specific due date (for example, `-end 01/31/01` includes accounts with a due date on or before January 31, 2001) or you can specify the due date as number of days from the current date (for example, `-end 5` includes accounts with a due date on or before 5 days from the current date).

  **Note:** The `end` time parameter cannot be greater than the system time. For example, if the current system date is 1/15/2005 and the `end` time specified is 1/31/2005, `pin_bill_accts` fails with “bad config:time” error message.

- `sponsorship`:
  - Specifies how charge sharing groups are billed. When you specify this option, the `pin_bill_accts` behavior is determined by the value of the `BillingFlowSponsorship` parameter, which can be one of the following:
    - 0: Bills owner and member accounts in any order.
    - 1: Bills owner accounts before member accounts.
    - 2: Bills member accounts before owner accounts.
  - Before using this option, ensure that `BillingFlowSponsorship` is set to the correct value. See "Configuring Billing for Groups and Hierarchies".

  **Note:** You cannot use this parameter with the `-discount` parameter.

- `discount`:
  - Specifies how discount sharing groups are billed. When you specify this option, `pin_bill_accts` behavior is determined by the value of the `BillingFlowDiscount` parameter, which can be one of the following:
    - 0: Bills owner and member accounts in any order.
    - 1: Bills owner accounts before member accounts.
    - 2: Bills member accounts before owner accounts.
Before using this option, ensure that **BillingFlowDiscount** is set to the correct value. See "Configuring Billing for Groups and Hierarchies".

**Note:**

You cannot use this parameter with the **-sponsorship** parameter.

**-cycle_charge_only**
Runs billing on accounts with /billinfo bill states 0 and 1. All cycle fees and billing-time discounts are applied, but totals are not calculated, and the bill is not finalized. At the end of the billing run, the /billinfo bill state is set to 2. See "About Bill States and the pin_bill_accts Utility".

**Note:**

- After running **pin_bill_accts** with this parameter, use the **-finalize_bill** parameter to finalize the bill.

**-finalize_bill**
Runs billing on accounts with /billinfo bill state 2. Totals are calculated, and items, bills, and so on are updated. At the end of the billing run, the /billinfo bill state is set to 0. See "About Bill States and the pin_bill_accts Utility".

**Note:**

- Use this parameter to finalize a bill after running the **pin_bill_accts** utility with the **-cycle_charge_only** parameter.

**-retry**
Runs billing for accounts that were previously not billed by **pin_bill_accts** due to some error. After the errors have been resolved, use this option to bill the failed accounts.

Use **pay_type 10007** with the **-retry** option to bill failed accounts that own nonpaying bill units first, and then run it without the **pay_type** option to bill all other failed accounts as follows:

pin_bill_accts -active -retry -pay_type 10007  
pin_bill_accts -active -retry

**-file billing_run_config_file_name**
Specifies the name and location of a billing run configuration file. The default file is **pin_bill_run_control.xml** in the *BRM_home/apps/pin_billd* directory. The billing run configuration file can contain either a list of account and bill unit POIDs or a list of billing segments and accounting days of months (DOMs).
Note:
When you run pin_bill_accts with a billing run configuration file, do not run it as a cron job. If you do, depending on the restrictions in the configuration file, some bill units might never be billed.

Note:
When running pin_bill_accts with -file option, ensure that the accounts specified in the billing run configuration file reside on the same database schema where pin_bill_accts is run. If the file contains accounts from different database schemas, pin_bill_accts reports an error. See "Setting Up Billing in a Multischema Environment".

-split
Generates detail revenue assurance data if you have enabled its collection in the billing utilities configuration file. (See "Enabling Billing Utilities to Generate Revenue Assurance Data" in BRM Collecting Revenue Assurance Data.) You can view the detailed data by generating Revenue Assurance Billing Detail report. The details are based on item types.

Note:
If you specify both the -split and -file parameters and the input file for the -file parameter includes a list of accounts and bill units, the Revenue Assurance Billing Detail report does not segregate the data based on the billing segment and billing day of month (DOM).

For more information, see "About Collecting Revenue Assurance Data from Billing" in BRM Collecting Revenue Assurance Data.

-test
Tests the utility, but does not affect accounts. Use this parameter to see which accounts will be billed, without actually creating bills for the balances due.

-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:

```bash
pin_bill_accts any_other_parameter -verbose > filename.log
```
-help
Displays the syntax and parameters for this utility.

Results
If the pin_bill_accts utility does not notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

When it is called internally by the pin_bill_day script, the pin_bill_accts utility logs error information in the pin_mta.pinlog file.

Error Handling
When the pin_bill_accts utility detects that the cycle fee processing was not completed for the /billinfo object, pin_bill_accts aborts with an error and sets the PIN_FLD_BILLING_STATUS field of the /billinfo object to PIN_BILL_ERROR. In addition, it sets the PIN_FLD_BILLING_STATUS_FLAGS field of the /billinfo object to PIN_BILL_FLAGS_CF_NOT_APPLIED (bit value 0x1000).

Note:
If any nonpaying child bill unit caused the failure, pin_bill_accts updates the billing statuses set in the PIN_FLD_BILLING_STATUS and PIN_FLD_BILLING_STATUS_FLAGS fields of the /billinfo object for both child and parent bill units.

pin_cycle_fees
Use this utility to perform the following tasks:

• Charge cycle forward fees.
• Identify cycle forward fees that have reached the end of free billing periods. For example, if a customer signs up for one month of free service, the pin_cycle_fees utility finds when the free period is over and applies the cycle forward fee balance impact to the customer's account.
• Cancel charge offers that have an expired pending cancellation date. For example, if a charge offer is set to cancel at a future date, the pin_cycle_fees utility cancels the charge offer.
• Bill charge offers with a delayed purchase start time.

See "Prorating Cycle-Forward Fees and Canceling Charge Offers By Using the pin_cycle_fees Utility".

Location
BRM_home/bin

Syntax

pin_cycle_fees  -regular_cycle_fees| -defer_cycle_fees | -defer_cancel | -defer_purchase
[ -start  mm/dd/yy | number_of_days ]
Parameters

-regular_cycle_fees
Charges cycle forward fees.

Note:
The -regular_cycle_fees parameter replaces the functionality of the pin_cycle_forward utility.

-defer_cycle_fees
Identifies and applies cycle forward fees that have reached the end of free billing periods.

-defer_cancel
Cancels expired charge offers.

-defer_purchase
Bills charge offers with a delayed purchase start time.

-start [mm/dd/yy or yyyy|number_of_days]
-end [mm/dd/yy or yyyy|number_of_days]
Start and end dates.
If a start date is specified, the entire day is included. The end date is automatically the current date if you do not specify a value for the -end parameter.
If an end date is specified, that entire day is included, ending at, but not including, the 0th (first) second of the next day (00:00:00 a.m.). The end date cannot be a future date.

-test
Tests the utility, but does not affect accounts. Use this parameter to see which accounts have reached the end of free billing without actually applying cycle forward fees or canceling the expired charge offers.

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

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

```
pin_cycle_fees any_other_parameter -verbose > filename.log
```

-help
Displays the syntax and parameters for this utility.
Results

If the `pin_cycle_fees` utility does not notify you that it was successful, look in the utility log file (`default.pinlog`) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

When it is called internally by the `pin_bill_day` script, the `pin_cycle_fees` utility logs error information in the `pin_mta.pinlog` file.

Error Handling

When the `pin_cycle_fees` utility encounters an error in applying regular, deferred, deferred purchase, or deferred cancellation cycle fees, it sets the PIN_FLD_BILLING_STATUS billing status field of the `/billinfo` object to PIN_BILL_ERROR. In addition, it sets the appropriate bit of the PIN_FLD_BILLING_STATUS_FLAGS field of the `/billinfo` object as follows:

- For regular cycle fees: PIN_BILL_FLAGS_CF_ERROR (bit value 0x100)
- For deferred cycle fees: PIN_BILL_FLAGS_DEF_CF_ERROR (bit value 0x200)
- For deferred purchase cycle fees: PIN_BILL_FLAGS_DEF_PURCHASE_ERROR (bit value 0x400)
- For deferred cancel cycle fees: PIN_BILL_FLAGS_DEF_CANCEL_ERROR (bit value 0x800)

Note:

If any nonpaying child bill unit caused the failure, `pin_cycle_fees` updates the billing statuses set in the PIN_FLD_BILLING_STATUS and PIN_FLD_BILLING_STATUS_FLAGS fields of the `/billinfo` object for both child and parent bill units.

When the status of the `/billinfo` object is set to PIN_BILL_ERROR, the `pin_bill_accts` utility, which runs after `pin_cycle_fees`, does not select this `/billinfo` object for billing. If you rerun `pin_bill_accts` with the `-retry` option, the billing opcode aborts with an error because the cycle fee processing failed for this `/billinfo` object.

Rerun `pin_cycle_fees` (directly or through the `pin_bill_day` script) before you can run billing on this `/billinfo` object.

**pin_make_corrective_bills**

Use the `pin_make_corrective_bills` utility to generate corrective bills for prior bills that have corrections or to process corrections on prior corrective bills. See "Creating Corrective Bills".

This utility should be used for corrective billing only. For regular bills, use "pin_bill_accts".

**Location**

`BRM_home/bin`
Syntax

```bash
pin_make_corrective_bills
  [-bill_no bill1, bill2,...]
  [-account_no acct1, acct2, acct3,...]
  [-start mm/dd/yy|number_of_days]
  [-end mm/dd/yy|number_of_days]
  [-file file_name]
  [-item_type /item/A/R_item]
  [-adj_reason D_id R_id]
  [-threshold_amount min_amt]
  [-correction_reason D_id R_id]
  [-correction_inv_type |R|L|D|S]
  [-no_adj_create [Y | N]]
  [-validate_only]
  [-help]
```

Parameters

- **-bill_no bill1, bill2...**
  Specifies the bill numbers for which corrective bills are required.

- **-account_no acct1, acct2, acct3...**
  Specifies the account numbers for which corrective bills are required.

- **-start mm/dd/yy|number_of_days**
  Specifies the start time to use in selecting the bills for corrective billing. When you specify the start date as a specific date, for example `-start 01/31/01`, the utility attempts to create corrective bills for all bills finalized on or before January 31, 2001. When you specify the start date as a number of days, for example `-end 5`, the utility attempts to create corrective bills for all bills finalized on or in the 5 days before the current date).
  If you omit this parameter, `pin_make_corrective_bills` sets the start date to 1 month before the current date.

- **-end mm/dd/yy|number_of_days**
  Specifies the due date for accounts to be billed. You can specify a specific due date (for example, `-end 01/31/12` includes accounts with a due date on or before January 31, 2012) or you can specify the due date as number of days from the current date (for example, `-end 5` includes accounts with a due date on or 5 days before the current date).
  If you omit this parameter, `pin_make_corrective_bill` sets the end date for the corrective billing for each selected bill to the end date on that bill.

  **Note:**
  The `end` time parameter cannot be greater than the system time. For example, if the current system date is 1/15/2012 and the `end` time specified is 1/31/2012, `pin_make_corrective_bills` fails with “bad config:time” error message.

- **-file file_name**
  Specifies that the list of accounts or the list of bill numbers will be read from a file, and provides the associated file name. The input file should be in XML format.
-item_type/item/A/R_item
Specifies the A/R item allocated to a bill as the criteria for selecting bills to include in the corrective billing. Enter the A/R item for example, item/adjustment, item/disputed, item/writeoff, item/review. Use -item_type/item/any to select all A/R items. If you omit this parameter, pin_make_corrective_bills uses the default value, item/adjustment, to select only those bills with adjustments allocated to them.

-adj_reason D_id R_id
Specifies the event-adjustment reasons associated with a bill as the criteria for selecting bills to include in the corrective billing. If you enter this parameter, you must specify both D_id which is the event adjustment domain number and R_id the reason number.
For example, -adj_reason D_id R_id

-threshold_amount min_amt
Specifies the minimum amount for A/R actions that should be reached in a bill to generate a corrective bill. pin_make_corrective_bills uses the value you input with the -threshold_amount parameter. If you omit this parameter, pin_make_corrective_bills uses the value currently in the CorrectiveBillThreshold business parameter.

-correction_reason D_id R_id
Specifies the correction reason to associated with the corrective bills. You must specify both D_id which is the event adjustment domain number and R_id the reason number.
For example, -correction_reason D_id R_id.

-corrective_inv_type R|L D|S
Specify the type of corrective invoice to associate with each corrective bill. R indicates Replacement Invoice and L indicates Invoice Correction Letter. S indicates Summary and D indicates Detail.
Enter the two values separated by a blank. For example, to generate a detail replacement invoice, you specify -corrective_inv_type R D when you run the pin_make_corrective_bills utility.
If you omit one of the entries (as in -corrective_inv_type R) or omit the blank between the two entries (as in -corrective_inv_type RD), the pin_make_corrective_bills application will fail to generate the corrective bill.

-create_if_no_corrections Y|N
Specifies whether to generate the corrective bill when there are no A/R charges for a bill. The default value is N which results in pin_make_corrective_bills not generating the corrective bill for any bill with no A/R charges.
If you enter Y as the value for -create_if_no_corrections, set the -corrective_inv_type parameter to R (to indicate Replacement Invoice).
pin_make_corrective_bills will not create the corrective bill if you specify -corrective_inv_type as L when you enter Y as the value for -create_if_no_corrections.
If you enter Y as the value for -create_if_no_corrections and you omit the -corrective_inv_type parameter, pin_make_corrective_bills creates the corrective bill if the /payinfo object for the account indicates a replacement invoice as the type of corrective invoice. Otherwise, it fails to create the corrective bill.

-validate_only
Specifies that pin_make_corrective_bills should only validate the eligibility of the bills for corrective billing. When you run pin_make_corrective_bills with the -
validate_only parameter, the utility only verifies that the selected accounts or bills are eligible for corrective billing. It does not generate corrective bills. When you use the -validate_only parameter, pin_make_corrective_bills lists the information for each bill in the default.pinlog log file.

-help
Displays the syntax and parameters for this utility.

Results
The pin_make_corrective_bills utility creates an entry in the default.pinlog utility log file for any errors it encounters in its process. The default.pinlog log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

Error Handling
If billing fails due to errors in bill units, the pin_make_corrective_bills utility automatically updates the billing status field in their /billinfo objects to error. If any nonpaying child bill units caused the failure, the utility updates the billing status for both child and parent bill units.

pin_rollover

The pin_bill_day script runs this Oracle Communications Billing and Revenue Management (BRM) utility to roll over all expired sub-balances that have not been rolled over.

For more information, see "When Rollover Events Occur" in BRM PDC Creating Product Offerings.

Location
BRM_home/bin

Syntax
pin_rollover [-verbose] [-test]

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

-test
Runs in test mode to find the accounts that meet the criteria for roll over, but does not perform any roll over. The test does not affect the balances (currency and noncurrency) of the accounts.

Results
If the pin_rollover utility does not notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.
**pin_trial_bill_accts**

Use this utility to calculate the balance due and create a trial invoice for each account.

---

**Note:**

Trial billing stops and reports a warning message when it encounters an account or bill unit with inactive status.

---

For information about trial billing, see "About Trial Billing".

---

**Note:**

To connect to the BRM database, the **pin_trial_bill_accts** utility needs a configuration file in the directory from which you run the utility. The **pin.conf** file for this utility is in `BRM_home/apps/pin_trial_bill`. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator’s Guide*.

---

**Note:**

Trial billing may stop responding if the Data Manager has too few back ends configured. You should change the default configuration settings for Data Manager and increase the number of back ends. For more information about setting the number of back ends, see "Improving Data Manager and Queue Manager Performance" in *BRM System Administrator’s Guide*.

---

**Location**

`BRM_home/bin`

**Syntax**

```
```
Parameters

- `start mm/dd/yy or yyyy+/- numberOfDays|0`
- `end mm/dd/yy or yyyy+/- numberOfDays|0`

The `start` and `end` dates determine which accounts are selected for trial billing and which billing cycles for those accounts are trial billed. See "Running Trial Billing With Date Ranges".

- `-f inputFile`

Specifies the text file that contains the list of account POIDs for trial billing.

**Example:**
Create trial invoices for accounts in `inputFile` and whose billing date is less than `4/1/2002` (current date is 3/15/2002):

```
pin_trial_bill_accts -end 04/01/2002 -f myListOfAccounts
```

- `-f_control inputFile`

Specifies the name and location of a text file that contains additional criteria for selecting accounts and bill units for trial billing. The default file is `pin_bill_run_control.xml` in the `BRM_home/apps/pin_billd` directory. This file can contain a list of account and bill unit POIDs or a list of billing segments, accounting days of month (DOMs), or both.

For more information, see "Specifying Bill Units, Billing Segments, and DOMs for Trial Billing".

---

**Note:**

You should not specify billing segments or accounting DOMs with bill unit POIDs in the input file. If you do, the utility considers only the bill units for trial billing.

The billing run configuration file can contain either a list of account and bill unit POIDs or a list of billing segments and accounting days of months (DOMs)

- `-active|-inactive|-closed`

Searches for accounts whose status is active, inactive, or closed. By default, all accounts are searched.

---

**Note:**

The `active`, `inactive`, or `closed` parameter does not apply when you use the `-f` parameter.

- `-bill_only`

Performs trial billing without generating trial invoices and collects revenue assurance data if you have enabled its collection in the trial billing utility configuration file. For more information, see "About Collecting Revenue Assurance Data from Trial Billing".

- `-pay_type ID`

Generates trial invoices for accounts with the specified payment method ID. If you do not specify the payment method ID, the `pin_trial_bill_accts` utility generates trial
invoices for all payment methods. See "Running Trial Billing According to Payment Type".

-retry
Runs trial billing for hierarchical bill units that were previously not billed during a trial invoicing run because of one or more errors. After the errors are resolved, use this parameter to rebill the unbilled hierarchical bill units. You must run nonpaying child bill units before running paying parent bill units. You must also regenerate trial invoices on the parent bill units that did not contain an error. For example, to run trial billing on bill units that contained an error and whose payment method is by check, use the following syntax:

```
pin_trial_bill_accts -pay_type 10007 -retry
pin_trial_bill_accts -pay_type 10012 -retry
pin_trial_bill_accts -pay_type 10012
```

For more information, see the discussion on working with hierarchical bill unit trial billing errors in *BRM Designing and Generating Invoices*.

-split
Generates detail revenue assurance data if you have enabled its collection in the trial billing utility configuration file. This parameter is valid only if Revenue Assurance Manager is installed.

**Note:**
If you use this parameter, trial invoices are generated unless you also specify the **bill_only** parameter.

You can view the detailed data by generating Revenue Assurance Billing Detail report. The details are based on item types.

**Note:**
If you specify both the **split** and **f_control** parameters and the input file for the **f_control** parameter includes a list of accounts and bill units, the Revenue Assurance Billing Detail report does not segregate the data based on the billing segment and billing day of month (DOM).

For more information, see "About Collecting Revenue Assurance Data from Trial Billing".

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

-help
Displays the syntax and parameters for this utility.
Results

If the pin_trial_bill_accts utility does not notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

pin_trial_bill_purge

Use this utility to purge trial invoices created with the "pin_trial_bill_accts" utility.

For information about trial billing, see "Running Trial Billing".

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

Location

BRM_home/bin

Syntax

pin_trial_bill_purge [-start mm/dd/yy|+- number_of_days|0] [-end mm/dd/yy|+- number_of_days|0] [-active|-inactive|-closed] [-f inputFile|-all] [-verbose] [-help]

Parameters

-start mm/dd/yy or yyyy|+- number_of_days|0
The start and end dates determine the billing cycles for which trial invoices are purged. Billing cycles must fall entirely within the range created by the start and end dates, except fall between the start and end dates. You can specify an absolute date or number of days before or after the current date.

Note:
If you do not specify a start date, pin_trial_bill_purge will purge trial invoices for all billing cycles before the end date for the account. If end date is not specified, it uses the current date.

-end mm/dd/yy or yyyy|+- number_of_days|0
The end date is used as the search criteria for retrieving accounts for purging trial invoices. It selects all accounts whose billing date is less than the end date. You can specify an absolute date or number of days before or after the current date.

Note:
If you do not specify an end date, pin_trial_bill_purge uses the current date for the end date.

Examples:
Purge trial invoices for accounts whose billing date is on or before 4/1/2002 (current date is 3/15/2002):
```
pin_trial_bill_purge -end 4/1/2002
```
```
pin_trial_bill_purge -end +16
```
Purge trial invoices for accounts whose billing date is on or before 3/1/2002 (current date is 3/15/2002)
```
pin_trial_bill_purge -end 3/1/2002
```
```
pin_trial_bill_purge -end -14
```
Purge trial invoices for accounts whose billing date is on or before the current date:
```
pin_trial_bill_purge -end 0
```
```
pin_trial_bill_purge
```
```
pin_trial_bill_purge -start 2/1/2002 -end 3/15/2002
```
```
pin_trial_bill_purge -start -45 -end 0
```

**-active|-inactive|-closed**
Searches for accounts whose status is active, inactive, or closed. By default, all accounts are searched.

**Note:**
The `active`, `inactive`, or `closed` parameter does not apply when you use the `-f` parameter.

**-f inputFile**
Specifies the text file that contains the list of account POIDs.
Example:
Purge trial invoices for accounts in the inputFile and whose billing date is less than 4/1/2002 (current date is 3/15/2002):
```
pin_trial_bill_purge -end 4/1/2002 -f myListOfAccounts
```

**-all**
Purges all trial invoices in the database.

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

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

**-help**
Displays the syntax and parameters for this utility.
**Note:**

If you do not specify a start or end date or a file with list of accounts, `pin_trial_bill_purge` selects all accounts whose billing date is on or before the current date.

**Results**

If the `pin_trial_bill_purge` utility does not notify you that it was successful, look in the utility log file (`default.pinlog`) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.