Oracle® Communications Billing and Revenue Management

Collections Manager

Release 7.5

E16698-10

August 2016

ORACLE®

Oracle Communications Billing and Revenue Management Collections Manager, Release 7.5

E16698-10

Contents

3 Setting Up Collections Manager

Preface

This document provides an overview of using Oracle Communications Billing and Revenue Management (BRM) Collections Manager to manage collections for your business.

Audience

This document is intended for anyone who installs, configures, administers, or customizes BRM, and particularly those who manage collections for your business.

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 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.

Related Documents

For more information, see the following documents in the BRM Release 7.5 documentation set:

- *BRM Release Notes*
- *BRM Concepts*

Document Revision History

The following table lists the revision history for this book.

Version	Date	Description
E16698-01	November 2011	Initial release.
E16698-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ■ Minor formatting and text changes.

Version	Date	Description
E16698-03	August 2012	Documentation updates for BRM 7.5 Patch Set 2. ■ Added the following sections: Promise-to-Pay Agreements About Defining Rule-Based Collections Scenario Assignment About Collections Scenario Replacement Creating Dependencies between Collections Actions in a Scenario Loading Additional Parameters for Scenario Assignment Increasing the Size of the CM Cache for Scenario Data Getting All Valid Collections Scenarios Replacing a Collections Scenario Managing Collections Sharing Groups Configuring Promise-to-Pay Agreements Assigning Bill Units to a Debt Collections Agency Customizing to Which Collections Group Members to Apply Collections Actions
E16698-04	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ■ Updated the **pin_collections_process** utility configuration in "Creating CollectionsInfoChange Business Events"
E16698-05	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ■ Minor formatting and text changes.
E16698-06	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16698-07	October 2013	Documentation updates for BRM 7.5 Patch Set 6. ■ Updated "Removing a Member from a Collections Sharing Group".
E16698-08	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ■ Minor formatting and text changes.
E16698-09	October 2014	Documentation updates for BRM 7.5 Patch Set 10. ■ Updated the "Promise-to-Pay Agreements" section.
E16698-10	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ■ Made minor formatting and text changes

1

Understanding Collections Manager

This chapter provides an overview of using Oracle Communications Billing and Revenue Management (BRM) Collections Manager to manage collections for your business.

Important: Collections Manager is an optional product that you download separately.

Before using Collections Manager, should be familiar with the following:

- Basic BRM concepts. See "Introducing BRM" in *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- If you plan to use invoice reminders, the Universal Messaging Store (UMS). See "Using BRM Messaging Services" in *BRM Developer's Guide*.

About Collections

Collections, or debt management, is a proactive process used by businesses to collect overdue payments from their customers. Collections includes identifying accounts that have overdue balances, determining whether those accounts meet predefined criteria for action, and then taking those actions.

Every business defines its own processes, but collections generally involves a series of increasingly serious steps taken against the customer:

- There is usually a minimum amount due that qualifies for collections. Very small amounts may not be worth the effort required to collect them.
- In most cases, businesses allow a grace period after payment is due. If payment is received within the grace period, no action is taken.
- After the grace period ends, a first action such as a reminder on an invoice, a dunning letter, or a phone call takes place. A late fee may also be assessed.
- If no payment is received as the result of the first collections action, additional steps can be taken. These actions range from additional letters or phone calls to stopping service to referring the case to a collections agency.
- An account is removed from collections when the overdue amount and any late fees are paid or when the debt is written off.

There is a great deal of variation in collections policies. For example, businesses define different criteria for entry into collections, different sequences of actions, different time periods before the debt is written off, and so on.

Policies may also vary within a business. For example, some businesses adopt more aggressive policies against accounts with low credit scores and follow a less aggressive course for accounts with high credit scores. Similarly, accounts that have never been in collections before may receive more favorable treatment.

About Collections Manager

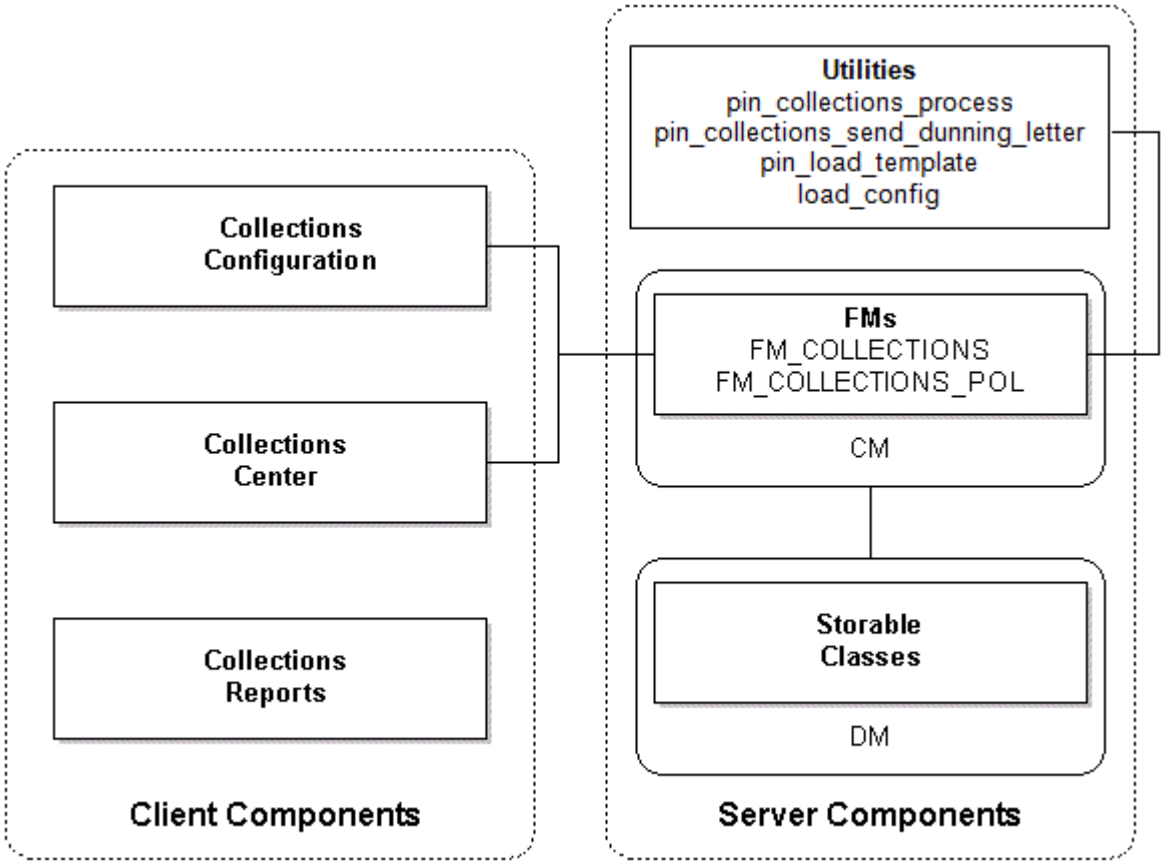
Collections Manager is an optional product that manages in-house debt collection. Collections Manager performs all of its activities at the bill unit (**/billinfo** object) level rather than at the account level. When an account has multiple bill units, Collections Manager targets only those bill units that have overdue balances, but it ignores the account's bill units that do not qualify for collections.

Collections Manager provides tools that enable you to:

- Configure collections criteria
- Determine which bill units have overdue balances
- Apply your collections criteria to those bill units
- Perform automatic collections activities, such as imposing late fees and sending dunning letters
- Manage manual collections activities, such as phone calls by collections agents, or converting payments to installments
- Replace a billing scenario based on the severity

Collections Manager includes client and server components that work together, as shown in Figure 1–1.

Figure 1–1 Collections Manager Components



Collections Manager includes these main components:

- **Collections Configuration**: A GUI application used to define collections scenarios and actions, set up aging buckets, and define user roles for collections personnel. See "Setting Up Collections Manager".
- **Collections Center**: A GUI application used by collections agents and collections managers. Collections agents use it to view, add, and cancel collections actions for bill units. Collections managers use it to oversee the collections process and to assign bill units to agents. See "Managing Bill Units in Collections".
- **Collections Reports**: A comprehensive reporting feature that you use to display information such as the bill unit status, the delinquent amount, and the total balance for all bill units in collections. Reports can be displayed in summary or detailed form. See "Collections Manager Reports" in *BRM Reports*.
- **Utilities**: Command-line applications that perform specific collections tasks.
  - The **pin_collections_process** utility is a daily batch process that identifies bill units that should be placed in collections and performs automatic collections actions. **pin_collections_process** uses criteria set up in Collections Configuration to identify bill units.
  - The **pin_collections_send_dunning** utility is a daily batch process that generates dunning letters based on data collected by **pin_collections_process**. Dunning letters can be sent via email or printed.

  - The **pin_load_template** utility loads dunning letter templates into the database.
  - The **load_config** utility loads additional criteria to be used for assigning a bill unit to a collections scenario.

- **Collections Manager Functional Modules (FMs)**: Opcodes that are called by the collections applications and utilities. There are standard opcodes that perform the basic collections functionality and policy opcodes that you use to customize that functionality. See "Collections Manager FM Standard Opcodes" and "Collections Manager FM Policy Opcodes" in *BRM Developer's Reference*.
- **Storable classes**: New and extended classes, including **/config** subclasses, that store collections-related data.

> **Note:** You can also integrate Collections Manager with a custom client application. For information, see "About Integrating Collections Manager with Custom Client Applications".

See "Installing Collections Manager" for information about installing the client and server components.

How Collections Manager Processes Bill Units

The main Collections Manager process (triggered by the **pin_collections_process** utility) searches the BRM database to find bill units with overdue balances and bill units that are already in collections.

> **Note:** You should run **pin_collections_process** every day to ensure that bill units enter collections promptly. The timing of actions, such as invoice reminders, dunning letters, and late fees, is based on the day the bill unit enters collections. See "About Overdue and Entry Dates".

Collections Manager processes bill units in two phases:

- During collections preprocessing, the bill unit is evaluated to determine if it should enter, exit, or remain in collections.
- During scenario processing, a bill unit that enters or remains in collections is processed based on its collections profile. These profiles are used only for collections and are not the same as the customer profiles that can be displayed in Customer Center.

Collections Manager Preprocessing

During preprocessing, Collections Manager determines whether an account's bill unit should enter, exit, or remain in collections. Preprocessing differs depending on whether the bill unit is already in collections.

- **For bill units already in collections**, the process compares the current overdue balance to the exit criteria you defined in the collections scenario (see "About Scenario Processing"). If the overdue balance is less than or equal to the exit criteria for the bill unit's scenario, the bill unit is removed from collections. If the overdue balance is greater than the exit criteria, the bill unit remains in collections and enters scenario processing.

For example, if the exit criteria is an overdue balance less than or equal to $10, a bill unit with an overdue balance of $8 would be removed from collections, and a bill unit with an overdue balance of $20 would remain in collections.

- **For bill units not in collections**, the process compares the current overdue balance to the minimum collections value that you defined. (See "Setting the Minimum Overdue Balance to Process".)
  - If the overdue balance is less than the minimum collections value, the bill unit remains out of collections and no further action takes place.
  - If the overdue balance is greater than the minimum collections value, the bill unit is assigned to a collections profile based on criteria you defined in advance. By default, all bill units belong to a single collections profile, but you can create any number of custom profiles. For example, you can assign bill units to profiles based on credit score or payment history. See "Setting Up Collections Profiles".

After a bill unit is assigned to a collections profile, the collections process checks if the bill unit meets the entry criteria for any of the scenarios that are valid for the bill unit's profile. The default entry criteria are based on severity, overdue balance and the number of days overdue. For example, you can specify an overdue balance of $100 that is at least 30 days old. You can define additional parameters for the entry criteria. See "About Collections Scenarios".

- If the bill unit does not meet the entry criteria for any of the scenarios for its collections profile, it remains out of collections and no further action takes place.
- If the bill unit meets the entry criteria for a scenario, it is assigned to that scenario, enters collections, and proceeds to scenario processing.
- If the bill unit meets the entry criteria for more than one scenario, it is assigned to the scenario with the highest entry criteria or the highest severity.

  For example, you might have the following three scenarios for a collections profile:
  - $50 entry requirement and severity 1
  - $100 entry requirement and severity 1
  - $100 entry requirement and severity 2

  A bill unit with an overdue balance of $101 is assigned to the scenario with the $100 requirement with severity 1 because it has the highest entry criteria and the highest severity.
- If you have configured additional parameters for the entry criteria, the bill unit is evaluated for the new scenario. If the new scenario is different from the existing scenario, the bill unit is exited from the existing scenario and is assigned to the new scenario. For example: while in collections, the customer makes a partial payment that reduces the overdue balance sufficiently to allow a scenario replacement to something less severe. The customer service representative (CSR) triggers an automated scenario replacement process that reevaluates the scenario assignment logic on the bill unit and assigns the bill unit to a new scenario.

About Scenario Processing

A collections scenario defines severity, entry and exit criteria as well as the sequence of actions that take place while an account's bill unit is in collections. See "About Collections Scenarios".

During scenario processing, Collections Manager first determines if any actions are due for this bill unit today. The timing of actions is based on the number of days since the bill unit entered the scenario. (For more information about the dates used in scenario processing, see "About Overdue and Entry Dates".) If an action is due today, Collections Manager then determines whether the action has a **Pending** status. (See "About Collections Action Dependencies".)

- If no actions are due, the process moves to the next bill unit.
- If the scenario includes actions to be carried out today but they have a **Waiting For Dependents** status, the process moves to the next bill unit.
- If the scenario includes actions to be carried out today and they have a **Pending** status, these actions are executed. Some actions happen automatically, some require manual intervention.
  - Automatic actions are performed by Collections Manager. They include predefined actions, such as sending dunning letters or invoice reminders, as well as custom actions that you define. See "Understanding System Actions" and "Understanding Custom Actions".
  - Manual actions, such as making phone calls, or converting payments to installments, are performed by collections agents. Collections agents use Collections Center to track the bill units and actions, either mandatory or optional, for which they are responsible. See "Understanding Manual Actions" and "Managing Bill Units in Collections".

After all scheduled actions have taken place, Collections Manager updates the bill unit's balance and checks to see if the bill unit now meets the exit criteria for the scenario. (It is possible that a payment was received as part of a manual action or that the scenario calls for writing off the balance.)

- If the new overdue balance is less than or equal to the amount specified in the scenario's exit criteria, the bill unit exits the scenario and leaves collections.
- If the overdue balance is still greater than the exit criteria, the bill unit remains in collections.

About Collections Scenarios

Collections scenarios determine the way bill units are handled in Collections Manager. Scenarios define the entry criteria that determine whether a bill unit will be handled by the collections system, the sequence of actions that take place when a bill unit is in collections, the exit criteria that determine when a bill unit leaves collections, and the severity level that determines priority of the scenario assignment to a bill unit when a bill unit (**/billinfo** object) is eligible for two scenarios that have the same entry criteria.

For example, you can define a scenario with the following characteristics:

- A bill unit enters collections with a minimum overdue balance of $100 that is at least 30 days old.
- A bill unit leaves collections when the overdue balance falls below $25.
- The customer receives a courtesy phone call after 10 days in collections.
- A late fee is assessed and a dunning letter is sent after 30 days in collections.
- A second dunning letter is sent after 45 days.
- The bill unit is closed and the debt written off after 120 days.
- The bill unit is inactivated after 180 days.

- The severity level that determines the priority of the scenario assignment.

Each scenario includes a list of actions that are performed either:

- On their due date. For example, a manual phone call is made 5 days after the bill unit enters collections, and a dunning letter is sent 10 days after the bill unit enters collections.
- In a specified order. For example, a CSR must make a courtesy phone call to the customer before the system refers the account to a collections agency.

About Overdue and Entry Dates

There are two important dates that collections managers use when processing bill units:

- The *overdue date* is the due date of the bill that causes the bill unit to enter collections. By default, Collections Manager uses the due date of the latest overdue bill at the time that the bill unit enters collections.

  You can configure Collections Manager to use the earliest due date instead. See "Configuring How Collections Manager Determines Dates".

- The *entry date* is the date on which the bill unit enters a collections scenario. By default, Collections Manager defines the entry date as the overdue date plus the number of days specified in the scenario's entry criteria. For example, if the bill due date is June 15 and the scenario specifies that the bill must be at least 10 days overdue, the entry date is June 25. You can configure Collections Manager to use the actual date on which the bill unit is processed and enters collections. See "Configuring How Collections Manager Determines Dates".

> **Note:** Do not schedule actions for the same calender date as the entry date. Scheduling actions on the entry date can cancel the actions.

Collections Manager stores the overdue date in the PIN_FLD_OVERDUE_T field of the **/collections_scenario** object.

Collections Manager uses the overdue date to determine the aging buckets into which a bill unit falls during scenario processing. See "Defining Aging Buckets".

Collections Manager uses the entry date to determine when the actions specified in a scenario take place. For example, you can define a scenario so that a dunning letter is sent 30 days after the bill unit enters the scenario.

> **Important:** If you run Collections Manager irregularly or at long intervals, the overdue and entry dates for a bill unit could be misleading. For example, if you do not run collections for several months, a bill unit that has not paid its bill during this period will enter collections with overdue and entry dates based on the latest of several overdue bills. As a result, collections actions will begin after a lengthy period of non-payment. To avoid this risk, you should run collections regularly, preferably daily.

About Collections Action Dependencies

Collections action dependencies ensure that all of a scenario's actions are performed in order and maintain the interval between actions. To do this, Collections Manager:

- Determines whether a scenario's preceding action has been completed or canceled before performing an action on its due date.
- After an action is completed or canceled, reschedules the due date of all of the scenario's remaining actions.

For example, assume a collections scenario includes the following actions:

- Action 1: Collections agent makes a courtesy phone call two business days after the bill unit enters collections.
- Action 2: Collections Manager sends an invoice reminder four business days after the bill unit enters collections.
- Action 3: Collections Manager applies a late fee six business days after the bill unit enters collections.

On day four, Collections Manager determines whether Action 1 has been completed or canceled and, since Action 1 is still open, does not send an invoice reminder. After the collections agent makes the courtesy phone call on day five, Action 1 is marked as completed and then Collections Manager reschedules Action 2 for seven business days after the bill unit enters collections and Action 3 for nine business days after the bill unit enters collections. This ensures that there are two business days between the courtesy phone call and the invoice reminder, and four business days between the courtesy phone call and the late fee.

Collections Manager creates dependencies between actions in a scenario by using action status settings:

- **Pending**: This status specifies that the action can be executed on its due date. All preceding actions have been canceled or completed.
- **Waiting For Dependents**: This status specifies that the action cannot be performed until the preceding action is canceled or completed. If an action is set to this status on its due date, Collections Manager does not perform the action and then moves to the next bill unit.

When a bill unit first enters a collections scenario, Collections Manager sets the first action's status to **Pending** and all other actions' status to **Waiting For Dependents**. After the first action is completed or canceled, Collections Manager updates the second action's status from **Waiting For Dependents** to **Pending** and then reschedules the due date of all remaining actions. After the second action is completed or canceled, Collections Manager updates the third action's status from **Waiting For Dependents** to **Pending** and reschedules the due date of all remaining actions. This process continues for all of the scenario's remaining actions.

About Collections Sharing Groups

Some countries have legal requirements that prevent actions, such as collections, from being taken against minors. This law applies to minors even when they have their own accounts and pay their own bills. Collections sharing groups allow you to define the legal entity, such as a parent or guardian, who is responsible for a bill if it enters into collections.

When you create an account and its bill units in Customer Center, you can specify whether collections actions must be taken against another account and bill unit. To do this, you create a collections sharing group, which consists of the following:

- A parent bill unit: During collections processing, Collections Manager enters only the parent bill unit into a collections scenario. The parent bill unit is assigned a collections scenario based on:
  - The combined overdue balance from all bill units in the collections group. For example, if one child bill unit is $100 past due and a second child bill unit is $50 past due, the overdue balance is $150.
  - The oldest due date from all bill units in the collections group. For example, if one child bill unit had a due date of June 1 and the second child bill unit had a due date of June 10, the overdue date is June 1.
- One or more child bill units: During collections processing, Collections Manager skips all child bill units in the group. The child bill unit and parent bill unit do not need to be linked hierarchically.

A bill unit can belong to only one collections sharing group, and a parent bill unit cannot be a child bill unit of the same collections sharing group. You can change a collections sharing group's parent and add or remove child bill units at any time. For more information about creating, modifying, and deleting collections sharing groups, see the Customer Center Help.

When you define a collections action in Collections Configuration Center, you can specify whether the action applies to the parent bill unit, to all the child bill units, or to the parent and all the child bill units. For example, you can specify that all "Referring the Account to a Collections Agency" actions apply to the parent bill unit only, and all "Sending a Dunning Letter" actions apply to the parent and all child bill units.

Corrective Billing and Collections

When BRM generates a corrective bill for a bill that is not yet in collections, the Collections Manager uses the sum of the adjustments for the bill to determine whether an account's bill unit should enter collections.

If BRM generates a corrective bill for a bill already in collections, Collections Manager takes the following actions based on the sum of the adjustments for the bill:

- If the sum of the adjustments for the bill goes below the threshold, BRM moves that bill out of collections.
- If the sum of the adjustments for the bill remains above the threshold, BRM maintains the account's bill unit in collections but replaces the bill number of that bill with the bill number of the corrective bill.

Promise-to-Pay Agreements

When a bill unit (**/billinfo** object) is in collections, the customer can request a promise-to-pay agreement. This agreement allows you to defer running the collections actions for a bill unit, allowing the customer to pay the due amount at a date beyond the original payment due date. The customer can pay the due amount in installments.

Using Collections Center, you can add or update a customer's payment method and reschedule payment dates and amounts for **Promise to Pay** and **Collect Payment** actions. See the Collections Center Help for more information.

To use these options in Collections Center, you must set the following permissions through Permissioning Center:

- **/collectionapps/collections/updatepayment**: To allow you to update promise-to-pay payment details in Collections Center.

- **/collectionapps/collections/newcard**: To register a new credit card or direct debit account details.
- **/collectionapps/collections/maskcarddetails**: To see all the digits of the credit card or the direct debit account.

> **Note:** If credit card tokenization is enabled, you can view only the last 4 digits of the credit card. For more information about credit card tokenization, see "About Credit Card Tokenization" in *BRM Configuring and Collecting Payments*.

By default, the **Promise to Pay** option is available in the Collections Center **Actions** menu. To disable the **Promise to Pay** option in the Collections Center **Actions** menu, use the **/collectionapps/collections/promisetopay** permission.

See the Permissioning Center Help for more information.

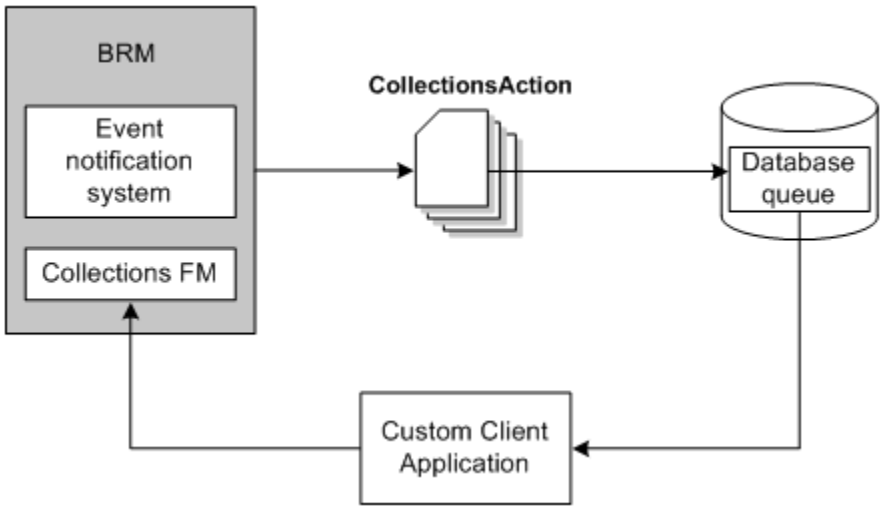
You can customize the system's behavior using the policy opcodes. See "Configuring Promise-to-Pay Agreements" for more information.

About Integrating Collections Manager with Custom Client Applications

Collections Manager can be integrated with a custom client application, such as Siebel CRM, so customer service representatives (CSRs) can track and manage collections activities directly in the custom client application.

The custom client application manages collections activity, such as preparing dunning letters or applying late fees, by calling the Collections Manager API. Collections Manager notifies the custom client application when a collections activity occurs, such as an account entering collections, by using the event notification system. Figure 1–2 shows how data is passed.

Figure 1–2 How Custom Client Applications Communicate with Collections Manager



To integrate Collections Manager with a custom client application:

- Configure the custom client application to call the Collections Manager API. See "About Calling the Collections Manager API".

- Configure Collections Manager to notify the custom client application when collections information changes. See "About Notifying Custom Client Applications when Collections Activity Occurs".

About Calling the Collections Manager API

CSRs need to perform the following collections tasks in their custom client application:

- Manage the collection of overdue balances
- Retrieve information about collections activities
- Perform manual collections activities
- Prepare dunning letters for customers

To perform these tasks, you must configure your client application to accept the required information and pass it to the Collections Manager API. You can determine the information required by looking at the target opcode's input flist. For more information about calling the Collections Manager API, see "Using the Collections Manager API".

Note: You can pass information to the Collections Manager API either directly from the custom client application or through Oracle Application Integration Architecture (AIA). For more information about AIA, see "Using BRM With Oracle Application Integration Architecture" in *BRM Concepts*.

About Notifying Custom Client Applications when Collections Activity Occurs

Collections Manager uses event notification to alert custom client applications when collections activities occur, such as an account entering collections or a dunning letter being sent. Collections Manager sends an alert in the form of an **/event/audit/collections/action** notification event, which includes information about the collections action, such as:

- The POID of the **/collections_action** object
- The action type: manual, custom, or system
- A description of the action
- The action status

For a comprehensive list of information in the notification event, see the **/event/audit/collections/action** specification file.

The following shows how BRM generates the **/event/audit/collections/action** notification event:

1. When you run the **pin_collections_process** utility, one of the following collections activities occurs:
   - An account enters or exits collections
   - A **/collections_action** object is created
   - A collections action's status is updated in BRM
   - A manual collections action is required
   - BRM creates a **/schedule** object

2. A Collections Manager FM opcode generates the **/event/audit/collections/action** event.
3. The event notification system detects the **/event/audit/collections/action** event and calls the opcode specified in the **/config/notify** object. By default, it calls the PCM_OP_COLLECTIONS_PUBLISH_EVENT opcode.
4. PCM_OP_COLLECTIONS_PUBLISH_EVENT calls the PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT policy opcode.
5. The PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT policy opcode adds required fields to **/event/audit/collections/action**.
6. PCM_OP_COLLECTIONS_PUBLISH_EVENT passes **/event/audit/collections/action** to the Payload Generator External Module (EM).
7. The Payload Generator EM collects events in its payload until they compose a complete **CollectionsAction** business event. When the business event is complete, the Payload Generator EM sends it to the Synchronization Queue Data Manager (DM).
8. The Synchronization Queue DM publishes the **CollectionsAction** business event to the database queue. See "Understanding the Synchronization Queue Data Manager" in *BRM Synchronization Queue Manager* for more information.
9. The client application retrieves the business event from the database queue and updates the information in its system.

To notify custom client applications when collections activity occurs:

- Configure BRM to publish **/event/audit/collections/action** notification events. See "Sending Collections Notifications to Custom Client Applications".
- Customize your client application to retrieve notification events from the database queue.

About Defining Rule-Based Collections Scenario Assignment

By default, Collections Manager assigns the collections scenario to the bill unit based on the overdue balance and the number of overdue days. You can use the **load_config** utility to load additional parameters to be used for the collections scenarios assignment. The utility loads the additional parameters into the **/config/collections/scenario_params** object that is used to define an evaluation formula. Each formula is in turn linked to the collections scenario assigned to the bill unit. See "Loading Additional Parameters for Scenario Assignment" for more information.

Using Collections Configuration, you can add, modify, or delete the value of the additional parameters from the scenario definitions. See the Collections Configuration Help for more information on how to configure the additional parameters for a collections scenario.

About Collections Scenario Replacement

When a bill unit meets the entry criteria for a collections scenario, it is assigned to that scenario, enters collections, and proceeds to scenario processing. If there is a change in the criteria, such as the customer making a partial payment, the bill unit may not meet the entry criteria for the collections scenario. Collections Manager reevaluates the entry criteria based on additional parameters and replaces the assigned collections scenario with a new collections scenario. See "Replacing a Collections Scenario" for

more information.

When a bill unit is in collections, you can use Collections Center to replace the collections scenario assigned to the bill unit. See the Collections Center Help for more information.

About Brands in Collections Manager

Collections Manager fully supports branding. You can create different collections policies for different brands. For example, the scenarios and actions you define in Collections Configuration are specific to the active brand that you select. In Collections Center, collections agents and collections managers see bill unit information for the brands they select. In both applications, users have access only to the brands allowed by their access privileges.

See "About Branding" in *BRM Managing Customers* for general information about brands.

What's Next?

To install Collections Manager, see "Installing Collections Manager".

To set up Collections Manager and learn more about Collections Configuration, see "Setting Up Collections Manager".

To learn more about Collections Center, see "Managing Bill Units in Collections".

2

Installing Collections Manager

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Collections Manager software. Installation includes both client and server components.

> **Note:** Collections Manager Reports are installed separately. See "Installing BRM Reports" in *BRM Reports*.

Before you read this chapter, you should be familiar with BRM concepts and architecture. See "Introducing BRM", "BRM System Architecture" in *BRM Concepts*, and "Understanding Collections Manager".

Installing Server Components

The BRM Collections Manager server components include opcodes, utilities, and storable classes. You install these components in the *BRM_Home* directory.

System Requirements

The Collections Manager server components are supported on the HP-UX IA64, Solaris, Linux, and AIX operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Software Requirements

Before installing the BRM Collections Manager server components, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- Oracle database software.

Installing BRM Collections Manager Server Components

> **Note:** If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install BRM Collections Manager server components:

1. Download the software to a temporary directory (*temp_dir*).

   Important:

   - If you download to a Windows workstation, use FTP to copy the .bin file to a temporary directory on your UNIX server.
   - You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. For information, see "Increasing Heap Size to Avoid “Out of Memory” Error Messages" in *BRM Installation Guide*.

2. Go to the directory where you installed the Third-Party package and source the **source.me** file.

   Caution: You must source the **source.me** file to proceed with installation, otherwise "suitable JVM not found" and other error messages appear.

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

3. Go to the temp_dir directory and enter this command:

   ```
   7.5.0_CollectionsMgr_platform_opt.bin
   ```

   where *platform* is the operating system name.

   Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

4. Follow the instructions displayed during installation. The default installation directory for Collections Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or Collections Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the Collections Manager package and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

C shell:

```
source source.me.csh
```

6. Go to the *BRM_Home*/**setup** directory and run the **pin_setup** script.

   Note: The **pin_setup** script starts all required BRM processes.

7. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home*/**apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

8. Start all BRM processes. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Your Collections Manager installation is now complete.

Configuring the pin_bill_day Script to Run Collections

To run collections automatically, you need to edit the **pin_bill_day** script.

1. Open the **pin_bill_day** script in a text editor. The script is located in *BRM_Home*/**bin**.
2. Find the Collections Manager section of the script. If the following lines are commented out, remove the comment marks:

   ```
   cd ${PINDIR}/apps/pin_collections
   pin_collections_process
   pin_collections_send_dunning
   ```

3. Save and close the file.

Uninstalling BRM Collections Manager Server Components

To uninstall BRM Collections Manager, run the *BRM_Home*/**uninstaller/CollectionsMgr/uninstaller.bin**.

Installing BRM Collections Manager Client Components

BRM Collections Manager includes two client applications:

- Collections Configuration
- Collections Center

The client applications are supported on Windows platforms only. You can install them in multiple locations, in any combination. Neither application requires the other.

System Requirements

You can install the client applications on Windows.

Hardware Requirements

Table 2–1 shows the disk space required by client applications to download, extract, and install the software:

Table 2–1 Disk Space Requirements

Application	Required Space (MB)
Collections Configuration	110
Collections Center	110

Note: Collections Configuration and Collections Center require the Java Runtime Environment (JRE). It is included in both application packages and is approximately 50 MB. When the JRE gets installed with a BRM client application, it is not installed again.

Software Requirements

Before installing BRM Collections Manager, you must install:

- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- BRM Collections Manager server components. See "Installing Server Components".
- An application such as WinZip for extracting compressed files.

Installation Procedures

You install the BRM Collections Manager client components in the same way that you install other BRM clients. Both components are installed separately. See "Installing BRM Client Applications on Windows" in *BRM Installation Guide*.

Uninstalling BRM Collections Manager Client Components

To uninstall BRM Collections Configuration, choose **Start - Programs - Portal - Uninstall Collections Configurations** and answer the prompts.

To uninstall BRM Collections Center, choose **Start - Programs - Portal - Uninstall Collections Center** and answer the prompts.

What's Next?

After you install BRM Collections Manager, you need to set it up. See "Setting Up Collections Manager".

3

Setting Up Collections Manager

This chapter explains how to set up Oracle Communications Billing and Revenue Management (BRM) Collections Manager.

Before you read this chapter, you should be familiar with the following:

- BRM concepts and architecture. See *BRM Concepts*.
- BRM Collections Manager concepts. See "Understanding Collections Manager".
- Modifying BRM configuration files. See *BRM System Administrator's Guide*.

If your setup requires the modification of policy opcodes, C programming skills and knowledge of BRM development concepts and procedures are necessary. See *BRM Developer's Guide*.

For installation information, see "Installing Collections Manager".

About Setting Up Collections Manager

After you install Collections Manager, you must set it up to meet your business needs:

- Configure how Collections Manager processes bill units with overdue balances. See "Configuring Collections Manager".
- Integrate Collections Manager with your custom client application, if you are using a custom client application. See "Integrating Collections Manager with Custom Client Applications".
- Customize the punitive actions that Collections Manager performs, such as assigning late fees and finance charges. See "Customizing Collections Manager".

Configuring Collections Manager

To configure Collections Manager, you perform the following general tasks:

- Specify the custom pay types to process, if applicable. See "Adding Custom Pay Types".
- Set the minimum overdue balance for entering collections. See "Setting the Minimum Overdue Balance to Process".
- Set the number of bill units that Collections Manager retrieves during step searches. See "Setting the Number of Bill Units Retrieved during Step Searches".
- Set up Collections Manager to send invoice reminders. See "Setting Up Invoice Reminders".

- Create dependencies between collections actions in a collections scenario. See "Creating Dependencies between Collections Actions in a Scenario".
- Set up Collections Manager to send dunning letters. See "Setting Up Dunning Letters".
- Add or modify the Collections Center reason codes. See "Adding Custom Reason Codes for Collections Center".
- Define your aging buckets, collections scenarios, and collections actions. See "Defining Collections Features".
- Specify how Collections Manager delivers dunning letters for non-invoice bill units. See "Setting Dunning Letter Delivery Preferences for Non-Invoice Bill Units".
- Specify how Collections Manager determines the overdue and entry dates. See "Configuring How Collections Manager Determines Dates".

Adding Custom Pay Types

When you run the **pin_collections_process** utility, it considers only bill units (**/billinfo** objects) with the default pay types, such as 10001, 10002, 10003, 10004, 10005, and 10006.

To add custom pay types:

1. Open the *BRM_home/apps/pin_collections/pin.conf* file in a text editor, where *BRM_home* is the directory in which BRM is installed.
2. Add the following entry:

   - pin_collections_process pay_type *PaymentType1[, PaymentType2...]*

   Note: Valid pay type codes are in the *BRM_home***/include/pin_cust.h** file, under PAY_TYPE.

   For example, to add the wire transfer payment type:

   ```
   - pin_collections_process pay_type 10013, 10014
   ```

3. Save and close the file.

Setting the Minimum Overdue Balance to Process

When you run **pin_collections_process**, only bill units (**/billinfo** objects) with overdue balances greater than or equal to the amount you specify are considered.

To set the minimum overdue balance for collections:

1. Open the *BRM_home***/apps/pin_collections/pin.conf** file in a text editor.
2. Set the **minimum_due** entry to the minimum overdue balance:

   - pin_collections_process minimum_due *Amount*

   For example, to set the minimum overdue balance to $25:

   ```
   - pin_collections_process minimum_due 25.00
   ```

   Note: The default is **0.0**.

Important: The value you specify is a system-wide minimum value. Collections scenarios cannot have entry criteria lower than this value. For example, if the minimum value is $25 and a collections scenario has an entry criteria of $10, a bill unit will enter collections only if its overdue balance is $25 or more.

3. Save and close the file.

Setting the Number of Bill Units Retrieved during Step Searches

The Collections Functional Module (FM) opcodes use step search when searching for bill units. The step size determines the number of bill units retrieved at one time.

To specify the number of bill units to retrieve in step searches:

1. Open the *BRM_home*/**sys/cm/pin.conf** file in a text editor.
2. Set the **account_search_batch** entry to the number of bill units:

   ```
   - fm_collections account_search_batch Amount
   ```

   For example, to retrieve 500 bill units with each step of the search.

   ```
   - fm_collections account_search_batch 500
   ```

   Note: The default is **1000**.

3. Save and close the file.
4. Stop and restart the Connection Manager (CM).

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Setting Up Invoice Reminders

Collections Manager uses the Universal Message Store (UMS) framework to deliver invoice reminders. See "Using BRM Messaging Services" in *BRM Developer's Guide* for general information.

If you plan to use invoice reminders as part of the collections process, complete the following tasks related to the UMS framework:

- Enable messaging by modifying the PCM_OP_INV_POL_PREP_INVOICE policy opcode. See "Enabling Messaging" in *BRM Developer's Guide*.
- Load a message style sheet (**message.xsl**) that makes it possible to display messages in invoices. See "Loading the Message Style Sheet" in *BRM Developer's Guide*.
- Load invoice reminder templates into **/string** objects in the BRM database. See "Creating and Loading Message Templates" in *BRM Developer's Guide*. Collections Manager includes a sample invoice reminder template that you can modify and rename as necessary. The file is *BRM_home*/**sys/msgs/messages/message_invoice_reminder.en_US**.

Creating Dependencies between Collections Actions in a Scenario

By default, Collections Manager performs automated collections actions on their due date, regardless of whether the preceding collections actions have been completed or canceled. This can cause collections actions to be completed out of order.

You can configure Collections Manager to create dependencies between collections actions in a scenario and thus perform collections actions in order.

When so configured, Collections Manager performs the following additional steps during scenario processing:

- Performs a collections action on its due date only if the action's status is set to **Pending**.
- After completing or canceling a collections action, Collections Center asks whether to reschedule the due dates of any outstanding collections actions in the scenario. When set, Collections Manager calculates the difference in days between the action's completed or canceled date and the action's due date, and then moves the due dates of all outstanding actions by the calculated number of days. For example, assume a collections scenario contains two collections actions:
  - Action 1: A manual phone call from an agent 5 days after the bill unit enters collections.
  - Action 2: Referring the case to an outside collections agency 10 days after the bill unit enters collections.

  If Action 1 does not occur until day 7, Collections Manager changes the due date of Action 2 from day 10 to day 12. This maintains the interval between Action 1 and Action 2. All outstanding actions are also rescheduled accordingly to maintain the configured intervals. If rescheduled Action 2 falls on a weekend, it is processed the next working day, and outstanding actions are again rescheduled to maintain the interval between the actions. Rescheduling prevents two actions from occurring on the same day unless the actions are configured to do so.

To create dependencies between collections actions in a collections scenario:

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

   ```
   - fm_collections collections_action_dependency 1
   ```

   Note: The default is **0**, which disables dependencies between collections actions.

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

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Setting Up Dunning Letters

To set up Collections Manager to generate dunning letters, you must run the Email Data Manager (DM) and load your dunning letter templates.

Running Email Data Manager for Dunning Letters

The Email DM must be running before you can send dunning letters, even if you plan to print and mail the letters. For information about configuring the Email DM, see "Sending Email to Customers Automatically" in *BRM Managing Customers* and "Configuring the Email Data Manager for Printing" in *BRM Designing and Generating Invoices*.

Loading Dunning Letter Templates

You load dunning letter templates by using the **pin_load_template** utility. These dunning letter templates are used in Collections Configuration to set the dunning letters action and the invoice reminders action.

Dunning letter templates are XSL documents that provide the basic content of the letter, leaving placeholders for specific information such as the customer's name, overdue balance, and so on. You can design and load multiple dunning letter templates for different brands, scenarios, and situations. For example, if a given brand has three scenarios, you could load two dunning letters for each scenario.

Collections Manager includes two sample dunning letter templates: **dunning_first.xsl** and **dunning_last.xsl**. Both templates are stored in *BRM_home*/**sys/data/config/stylesheets**. You can use a third-party XSL editor to modify these templates or to create your own templates.

After creating your template, load it into the BRM database by using the **pin_load_template** utility. Templates are available in Collections Configuration immediately after you load them. See "Defining Collections Actions".

To load a dunning letter template:

1. Go to a directory that contains a valid configuration (**pin.conf**) file.

   Tip: A configuration file is installed automatically in *BRM_home*/**apps/pin_collections**.

   The **pin_load_template** utility uses information in the configuration file to connect to the BRM database. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

2. Load the XSL template by entering the following command:

   ```
   pin_load_template -brand brand_POID - name template_name -type dunning -locale
   locale_name -template file_name -usexsl
   ```

   where:

   - *brand_POID* is the Portal object ID (POID) of a brand account or the root account. Use a brand account POID to assign the template to a specific brand. Use the root account POID if your system does not use brands or if you want the template to be available to multiple brands. If the brand account POID contains spaces, use quotation marks.
   - *template_name* is the name of the template as it should appear in Collections Configuration.
   - *locale_name* is the BRM locale of the template. See "Locale Names" in *BRM Developer's Guide*.
   - *file_name* is the name and full path of the template file.

For example, the following command loads the **dunning_first.xsl** template for the root account:

```
pin_load_template -brand "0.0.0.1/account 1" -name DunningTemplate -type dunning
-locale en_US -template BRM_home/config/dunning_first.xsl -usexsl
```

Adding Custom Reason Codes for Collections Center

In Collections Center, you can perform operations such as inserting new actions, rescheduling actions, and exempting bill units from collections. See "Managing Bill Units in Collections" for an overview.

For many of these operations, you choose a reason that is associated with the action. Collections Center includes a default set of reasons, which you can modify.

To modify the default reasons, you can add or change reason codes in each **reasons.locale** file.

Defining Collections Features

Many Collections Manager configuration tasks are completed with Collections Configuration. Collections Configuration is a GUI application that you can use for:

- Branding and Collections Configuration
- Defining Aging Buckets
- Setting Up Collections Profiles
- Defining Collections Actions
- Defining Collections Scenarios
- Assigning Collections Personnel Roles

> **Important:** Some configuration tasks involve additional tasks outside Collections Configuration.

See "Installing Collections Manager" for information about installing Collections Configuration.

Branding and Collections Configuration

Collections Configuration supports branding. You can choose the active brand to which your configuration choices apply. You have access only to the brands allowed by your access privileges. (See the Collections Configuration Help for information about choosing the active brand.)

For example, if the active brand is Brand A, the profiles, scenarios, and actions you create during your Collections Configuration session will be valid only for Brand A. You must use a different login name to make configuration choices for other brands.

There are exceptions to this rule. The following actions are predefined by BRM, cannot be modified, and are valid across all brands:

- Inactivate bill unit
- Refer to collections agency
- Close bill unit
- Write off balance

Defining Aging Buckets

Aging buckets define time periods for overdue balances. Each bucket lasts a certain number of days, after which a balance falls into the next bucket. You can create up to 10 aging buckets, which apply systemwide to all scenarios.

For example, you could define 30-day, 60-day, and 90-day buckets. The 30-day bucket holds balances that are 1 to 30 days overdue. The 60-day bucket holds balances that are 31 to 60 days overdue, and the 90-day bucket holds balances that are 61 to 90 days overdue. Balances older than 90 days fall into an implicit fourth bucket.

Aging buckets are used to display information about the age of overdue balances. This information is displayed in Collections Center.

See the Collections Configuration Help for information about defining aging buckets.

Setting Up Collections Profiles

Collections profiles define categories into which you organize bill units for collections purposes. For example, you could organize bill units by credit score. Bill units with a score below a certain number are assigned to one collections profile, and bill units with higher scores are assigned to a different profile.

A bill unit's collections profile helps determine to which collections scenario a bill unit is assigned. (Scenarios define entry and exit criteria for collections as well as what happens to a bill unit while it is in collections. See "Defining Collections Scenarios".)

For example, bill units in the low-credit-score profile might be assigned to scenarios with more aggressive collections actions and lower entry criteria. Customers in the high-credit-score profile could be assigned to scenarios with more lenient policies.

By default, all bill units belong to a single collections profile. If you do not need a more complex system, you do not have to define any additional profiles. You can map all your scenarios to the default profile.

To set up additional profiles, you complete two sets of tasks:

1. Define names and descriptions for the profiles by using Collections Configuration. See the Collections Configuration Help for step-by-step instructions.

   When you define a new profile, Collections Configuration creates a **/config/collections/profile** object, which stores definitions of collections profiles.

2. Customize the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode. You need to write code that defines the characteristics of the collections profiles and maps bill units to them.

   Important: This step requires programming.

Configuring the Currency Used in Collections

The default profile uses US dollars for the currency. To use a different currency for collections, edit the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode.

If a scenario is configured in any other currency, you must customize the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode to consider this scenario for collections.

Increasing the Size of the CM Cache for Profiles Data

When setting up additional profiles, you might need to increase the space allocated to the profiles data in the CM cache.

To increase the CM cache size for profiles data:

1. Open the *BRM_home*/**sys/cm/pin.conf** file in a text editor.
2. Increase *cache_size* in the following entry:

   ```
   - cm_cache fm_collections_config_profiles_cache number_of_entries, cache_size,
   hash_size
   ```

   Note: The default is 20480 bytes.

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

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Defining Collections Actions

Collections actions are individual steps that are performed in the process of collecting overdue balances. An action might be applying a late fee or making a phone call.

You can define as many actions as you need. For example, you can define different phone call actions for different situations: courtesy calls, warning calls, and so on. The same action can be included in multiple scenarios. See "Defining Collections Scenarios".

When you define a collections action, you specify the following in Collections Configuration:

- The collections action name
- The action type: manual, one of several system types, and custom
- For bill units in a collections sharing group, the target bill units: the specified bill unit only (Self), the parent and all child bill units (All Members and Parent), or all child bill units (All Members)

Note: The action target defined for each of the collections actions for a customer is the same for all scenarios. When the action target mode is set for an action, the action target cannot be changed for any newly defined scenarios.

Understanding Manual Actions

Manual actions are completed by collections agents. When a scenario calls for a manual action, BRM saves information about the action in the database. Collections Center displays this information so that a collections agent can complete the action.

For example, suppose that a scenario calls for the customer to receive a courtesy phone call when the account's bill unit is 10 days into the collections process. When a bill unit enters collections, this phone call is included in the list of actions for the bill unit. On the day when the phone call is due, the phone call appears in the list of tasks for the collections agent assigned to the bill unit.

The most common manual action is a phone call, but you can create any others that your business requires.

For more information about manual actions, see the Collections Configuration Help.

Understanding System Actions

System actions are performed automatically by BRM when either of the following two utilities is run:

- **pin_collections_process**
- **pin_deferred_act** (as a part of the **pin_bill_day** script)

If an error occurs while performing any collections action when **pin_collections_process** is run, the status of the collections action object (**/collections_action**) and the schedule object (**/schedule**) corresponding to the collections action is updated to **Error**.

If an error occurs while performing any collections action when **pin_deferred_act** is run, the status of the collections action remains **Pending** but the schedule object corresponding to the collections action is updated to **Error**. When **pin_collections_process** is run and the collections actions is processed again, if an error occurs, the collections action object is updated to **Error**.

The status of collections actions is updated to **Completed** on successful completion of the actions.

By default, Collections Manager supports the following types of system collections actions:

- **Charging a late fee**. For late fees, you specify the currency type, such as US dollars, and the late fee type, either a specified amount or a percentage of the overdue amount.
- **Adding a finance charge**. For finance charges, you specify to charge a certain percentage of the overdue amount.
- **Sending a dunning letter**. For dunning letters, you specify the dunning letter template to use. You can have different templates for different purposes. For example, you could define several dunning letter templates with varying levels of severity. These templates could be used at different times in the same scenario or in separate scenarios.
- **Sending an invoice reminder**. For invoice reminders, you specify the invoice reminder template to use. You can have different templates for different purposes. For example, you could define several invoice reminder templates with varying levels of severity. These templates could be used at different times in the same scenario or in separate scenarios.

You can customize each system action's behavior by modifying policy opcodes:

- To customize the way late fees are assessed, modify the PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES policy opcode.
- To customize the way finance charges are calculated, modify the PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES policy opcode.
- To customize the way dunning letter content is gathered, modify the PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA policy opcode.
- To customize the way invoice reminder content is gathered, modify the PCM_OP_INV_POL_PREP_INVOICE policy opcode.

Important: Modifying policy opcodes requires programming.

For more information about system actions, see the Collections Configuration Help.

Performing Scheduled Collections Actions after Reinstalling Collections Manager

After reinstalling Collections Manager, running **the pin_collections_process** utility does not perform the collections actions that have a due date prior to the date on which you reinstalled Collections Manager.

You can perform all the scheduled collections actions after reinstalling Collections Manager and running the **pin_collections_process** utility irrespective of whether the collections actions have a due date prior to the date or after the date on which you reinstalled Collections Manager and ran **pin_collections_process**.

To perform all scheduled collections actions irrespective of whether they have a due date prior to the date or after the date on which you reinstalled Collections Manager and ran **pin_collections_process**:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Add the following entry:

   ```
   - pin_collections_process execute_all_actions value
   ```

   where *value* is:

   - **1** to perform all collections actions irrespective of whether they have a due date prior to the date or after the date on which you ran **pin_collections_process** after reinstalling Collections Manager.
   - **0** to perform only those collections actions that have a due date later than the date on which **pin_collections_process** is run after reinstalling Collections Manager. This is the default.
3. Save and close the file.
4. Stop and restart the CM.

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Understanding Custom Actions

Custom actions are system actions that you create. You define custom actions and descriptions in Collections Configuration and implement the actions by modifying the PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION policy opcode.

See the Collections Configuration Help for information about defining custom actions and see PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION for information about modifying the policy opcode.

Important: Modifying policy opcodes requires programming.

Increasing the Size of the CM Cache for Actions Data

When defining actions, you might need to increase the space allocated to the action data in the CM cache.

To increase the CM cache size for actions data:

1. Open the *BRM_home*/**sys/cm/pin.conf** file in a text editor.
2. Increase *cache_size* in the following entry:

   ```
   - cm_cache fm_collections_config_actions_cache number_of_entries, cache_size,
   hash_size
   ```

   Note: The default is 40960 bytes.

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

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Defining Collections Scenarios

When you create a new scenario, you decide which collections profile it applies to. You can define multiple scenarios with different characteristics for the same profile. For example, you could set up one scenario for relatively small overdue balances and another for larger balances.

A single scenario cannot be used with multiple collections profiles. If you want to implement scenarios with the same functionality across several profiles, you can define multiple scenarios with the same characteristics but different names.

A scenario includes an ordered list of actions that are taken against the bill unit. When you define a scenario, you select from a list of actions that you have previously defined and then specify the following characteristics:

- The order in which the actions should take place
- The number of days after entering collections that the action should be completed; for example, 5 days after entering collections.
- Whether an action is optional or mandatory

Note: If you define a collections scenario that includes both a writeoff and a bill unit inactivation, the writeoff must come before the inactivation. When a bill unit is inactivated, an event with a credit balance is created to cancel the remaining service for the bill unit. The cycle fees are then prorated and the remaining credit is returned to the bill unit, in some cases as a separate item. If the bill unit has already been inactivated, the writeoff will fail because of the unallocated item. In this case, a customer service representative (CSR) must manually allocate the item so that it can be written off.

For more information about defining scenarios, see the Collections Configuration Help.

Loading Additional Parameters for Scenario Assignment

You load additional parameters for a collections scenario assignment by using the **load_config** utility. See "load_config" in *BRM Developer's Guide* for more information.

The utility takes an XML file as input and creates or updates the **/config/collections/scenario_params** object in the BRM database. Collections Configuration reads the **/config/collections/scenario_params** object and displays the

parameters while defining the evaluation formula to associate with a collections scenario.

For more information about defining scenarios, see the Collections Configuration Help.

To load additional collections scenario parameters:

1. Open the *BRM_home*/**sys/data/config/config_collections_scenario_params.xml** file in a text editor.
2. Edit the file, which includes examples and instructions. Table 3–1 describes the parameters in this file:

Table 3–1 Elements Used to Store Additional Scenario Parameters

Element	Description
PARAM_NAME	Name of the parameter.
STRING_ID	The string ID of the parameter name. This field is used for localization.
STR_VERSION	The version of the localized string within the domain. This field is used for localization.
TYPE	The type of value that the parameter can be assigned, from one of the following types: ■ **1**: STR (alphanumeric) ■ **2**: INT (integer) ■ **3**: ENUM (indicating that the parameter is one of an ordered list of possible values. An array of values must be provided for this selection.) ■ **4**: DECIMAL ■ **5**: TSTAMP (timestamp) For example, to provide a set of possible values, you set the **<TYPE>** element to **3**, and enter an array of values for this parameter in the **<VALUES>** element
CLASS_NAME	The storable class object associated with the parameter.
FIELD_NAME	The field inside the storable class referenced in the **<CLASS_NAME>** element. To enter the field name of a field present under an array or a substruct, use the format <FIELD_NAME>*array.field*</FIELD_NAME >, which includes the parent field of the attribute separated by periods (.). For example: <FIELD_NAME>PIN_FLD_COLLECTIONS_PARAMS.PIN_FLD_CREDIT_PROFILE</FIELD_NAME >.
VALUES	An array list of values that the parameter can assume. The **<VALUES>** array list is present only if the selection for the **<TYPE>** element is **3**.

For example, the following entry defines a new parameter called **Payment Method**:

```
<PARAMS elem="0">
     <PARAM_NAME>Payment Method</PARAM_NAME>
     <STRING_ID>1</STRING_ID>
     <STR_VERSION>1</STR_VERSION>
     <TYPE>3</TYPE>
     <CLASS_NAME>/billinfo</CLASS_NAME>
     <FIELD_NAME>pay_type</FIELD_NAME>
     <VALUES elem="0">
               <NAME>CC</NAME>
               <VALUE>10003</VALUE>
               <STRING_ID>100</STRING_ID>
               <STR_VERSION>1</STR_VERSION>
     </VALUES>
     <VALUES elem="1">
               <NAME>DD</NAME>
```

```
                <VALUE>10005</VALUE>
                <STRING_ID>101</STRING_ID>
                <STR_VERSION>1</STR_VERSION>
     </VALUES>
     <VALUES elem="2">
                <NAME>CASH</NAME>
                <VALUE>10011</VALUE>
                <STRING_ID>102</STRING_ID>
                <STR_VERSION>1</STR_VERSION>
     </VALUES>
     <VALUES elem="3">
                <NAME>CHECK</NAME>
                <VALUE>10012</VALUE>
                <STRING_ID>103</STRING_ID>
                <STR_VERSION>1</STR_VERSION>
     </VALUES>
</PARAMS>
```

In this example:

- The name of the parameter is **Payment Method**.
- The type of value is **3** (which is ENUM, and so an array of values follows.)
- The storable class is **/billinfo**.
- The field inside the storable class object is **pay_type**.
- The **Values** array lists the four possible payment methods: **CC**, **DD**, **Cash**, and **Check**.

For example, the following entry defines a parameter called **Credit Profile**:

```
<PARAMS elem="3">
     <PARAM_NAME>Credit Profile</PARAM_NAME>
     <STRING_ID>4</STRING_ID>
     <STR_VERSION>1</STR_VERSION>
     <TYPE>2</TYPE>
     <CLASS_NAME>/profile/collections_params</CLASS_NAME>
     <FIELD_NAME>PIN_FLD_COLLECTIONS_PARAMS.PIN_FLD_CREDIT_PROFILE</FIELD_NAME>
</PARAMS>
```

In this example:

- The name of the parameter is **Credit Profile**.
- The type of value is **2** (which is INT.)
- The storable class is **/profile/collections_params**.
- The field including the path to the storable class object is PIN_FLD_ COLLECTIONS_PARAMS.PIN_FLD_CREDIT_PROFILE.

By default, the **config_collections_scenario_params.xml** file contains sample data for the following parameters:

- Payment Method (Credit Card, Direct Debit, cash, check) from the **/billinfo** class
- Account Status from the **/account** class
- Account Type from the **/account** class

To define additional data during scenario evaluation, you can create and maintain the required data (parameters and their values) in a customized data class or by

extending the **/profile** class. This data class should have a link to the bill unit to which the data is applicable. There can be only one record per bill unit. The parameter used should be at the 0th level of the storable class.

3. Save and close the file.
4. Open the *BRM_home*/**apps/load_config/pin.conf** file in a text editor.
5. Uncomment the following entry:

   ```
   - load_config validation_module libLoadValidCollections  LoadValidCollections_
   init
   ```

6. Save and close the file.
7. Load the updated file by running the following command:

   ```
   load_config -v config_collections_scenario_params.xml
   ```

   Important:

   - The **load_config** utility requires a configuration (**pin.conf**) file.
   - If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file. For example,

     ```
     load_config -v BRM_home/sys/data/config/config_collections_
     scenario_params.xml
     ```

   For more information on the **load_config** utility, see *BRM Developer's Guide*.

8. Stop and restart the CM.

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

   To verify that the updated collections parameter configurations were loaded, you can display the **/config/collections/scenario_params** object by using Object Browser, or use the **robj** command with the **testnap** utility.

   For more information on the **/config/collections/scenario_params** object, see *BRM Storable Class Reference*.

Increasing the Size of the CM Cache for Scenario Data

When evaluating scenarios, you might need to increase the space allocated to the scenario data in the CM cache.

To increase the CM cache size for scenario data:

1. Open the *BRM_home*/**sys/cm/pin.conf)** file in a text editor.
2. Increase *cache_size* in the following entry:

   ```
   - cm_cache fm_collections_config_scenario_cache number_of_entries, cache_size,
   hash_size
   ```

   Note: The default is 40960 bytes.

If the file does not have this entry, add it.

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

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Assigning Collections Personnel Roles

If you use Collections Center to manage collections activity, you need to assign personnel roles to collections agents and collections managers. You assign collections roles in Collections Configuration. The tasks that collections personnel can perform and the information they can see depend on their role.

Collections Configuration displays a list of all CSR accounts. You select the accounts that should have a collections role.

See "Creating a Customer Service Representative Account" in *BRM System Administrator's Guide* for more information about CSR accounts. The login permissions defined for each CSR account determine which brands the collections agents and collections managers have access to. See "Setting Up Access to Brands" in *BRM System Administrator's Guide*.

See the Collections Configuration Help for detailed instructions about assigning collections personnel roles.

Setting Dunning Letter Delivery Preferences for Non-Invoice Bill Units

You specify how to deliver dunning letters to *non-invoice* bill units by using the **pin_collections** configuration file (*BRM_home*/**apps/pin_collections/pin.conf**).

For non-invoice bill units, you can set the following delivery options:

- Whether to deliver the dunning letter as hardcopy or by email.
- For email delivery, whether to send the dunning letter as part of the message body or as an attachment.
- For dunning letters sent as email attachments, the file path to use for providing customized content in the email body.

> **Note:** To set delivery options for *invoice* bill units, use Customer Center.

Setting the Delivery Option

By default, dunning letters are printed and sent as hardcopy.

To set the delivery option for dunning letters:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Change the value of the **delivery_preference** entry:
   - To deliver dunning letters as hardcopy, enter a non-zero value. This is the default.
   - To deliver dunning letters by email, enter **0**.

     For example:

     ```
     - pin_collections_send_dunning delivery_preference 0
     ```

3. Save and close the file.

Setting the Email Delivery Preference

By default, dunning letters are delivered in the body of the email.

To set the email delivery preference for dunning letters:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Change the value of the **email_option** entry:
   - To deliver dunning letters as email attachments, enter **1**.
   - To deliver dunning letters in the email body, enter **0**. This is the default.

     For example:

     ```
     - pin_collections_send_dunning email_option 0
     ```
3. Save and close the file.

Specifying a File for the Email Body

When you send dunning letters as email attachments, you can include customized information in the email body. You specify the path to a text file for the content of the message.

To specify the path to the message content:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Set the path in the **email_body** entry:

   ```
   - pin_collections_send_dunning email_body Path
   ```

   For example, the following entry specifies the path to the **letter** file:

   ```
   - pin_collections_send_dunning email_body ./letter
   ```

   Important: This option takes effect only if **email_option** is set to **1**.
3. Save and close the file.

Configuring How Collections Manager Determines Dates

You can configure how Collections Manager determines the overdue and entry dates it uses (described in Table 3–2). For information about how Collections Manager uses these dates, see "About Overdue and Entry Dates".

Table 3–2 Overdue and Entry Dates

Option	Description
Overdue date	■ (Default) The overdue date is the due date of the latest overdue bill when the bill unit enters collections. This date is not updated as long as the bill remains in collections. ■ The overdue date is the due date of the oldest overdue bill when the bill unit enters collections, even if the overdue balance from that bill is not sufficient by itself to meet a scenario's entry criteria. See "Configuring the Overdue Date" for instructions.

Table 3–2 (Cont.) Overdue and Entry Dates

Option	Description
Entry date	■ (Default) The entry date is the overdue date plus the number of days specified in the entry criteria of the scenario. This date is updated if the overdue date changes. ■ The entry date is the actual date on which a bill unit is processed and enters collections. This date is not updated as long as the bill unit remains in collections. If Collections Manager is run at irregular or long intervals with this setting, the entry date can differ significantly from the overdue date. See "Configuring the Entry Date" for instructions.

The following examples illustrate overdue and entry dates based on the four possible combinations of configuration options. The examples are based on the following assumptions:

- There is a monthly charge of $15, due on the 15th of the month.
- The collections scenario specifies a minimum overdue balance of $20, at least 10 days late.
- No payments are received in January, February, or March, but a $15 payment is received before the due date in April.
- Collections Manager runs every day.
- The tables show the overdue and entry dates as of the end of each month.

Case 1: Default Settings for Both Dates

Table 3–3 shows the results of the default setting for both dates. The overdue and entry dates stay the same as long as the bill unit remains in collections.

Table 3–3 Default Date Example

Option	Jan.	Feb.	Mar.	Apr.
Payment received	None	None	None	15.00
Overdue amount	15.00	30.00	45.00	45.00
Overdue date	None	Feb. 15	Feb. 15	Feb. 15
Entry date	None	Feb. 25	Feb. 25	Feb. 25

Case 2: Optional Settings for Both Dates

Table 3–4 shows the results of the optional setting for both dates. When the bill unit enters collections, the overdue date is January 15 even though the overdue balance for that month was not sufficient by itself to enter the scenario. Also note that the overdue date changes when a payment eliminates the overdue balance for January.

Table 3–4 Optional Date Example

Option	Jan.	Feb.	Mar.	Apr.
Payment received	None	None	None	15.00
Overdue amount	15.00	30.00	45.00	45.00
Overdue date	None	Jan. 15	Jan. 15	Feb. 15
Entry date	None	Feb. 25	Feb. 25	Feb. 25

Case 3: Default Setting for Overdue Date and Optional Setting for Entry Date

In Table 3–5, the dates are identical with those for Case 1, but the entry date is determined by the actual processing date. If the collections process was not run daily, the entry date would not necessarily be the same as the date determined by the scenario's entry criteria.

Table 3–5 Processing Date Example

Option	Jan.	Feb.	Mar.	Apr.
Payment received	None	None	None	15.00
Overdue amount	15.00	30.00	45.00	45.00
Overdue date	None	Feb. 15	Feb. 15	Feb. 15
Entry date	None	Feb. 25	Feb. 25	Feb. 25

Case 4: Optional Setting for Overdue Date and Default Setting for Entry Date

In Table 3–6, the entry date changes when the overdue date is updated to reflect the payment. This would have the effect of delaying any remaining collections actions defined in the scenario.

Table 3–6 Varied Date Settings

Option	Jan.	Feb.	Mar.	Apr.
Payment received	None	None	None	15.00
Overdue amount	15.00	30.00	45.00	45.00
Overdue date	None	Jan. 15	Jan. 15	Feb. 15
Entry date	None	Jan. 25	Jan. 25	Feb. 25

Configuring the Overdue Date

To configure how Collections Manager determines the overdue date:

1. Open the *BRM_home*/**sys/cm/pin.conf** file in a text editor.
2. Change the value of the **old_overdue_behavior** entry:
   - To use the latest overdue bill date, enter **0**. This is the default.
   - To use the earliest overdue bill date, enter **1**.

     For example:

     ```
     - fm_collections old_overdue_behavior 1
     ```
3. Save and close the file.
4. Stop and restart the CM.

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Configuring the Entry Date

To determine how Collections Manager determines the entry date:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Change the value of the **use_current_time** entry:
   - To use the entry date, enter **0**. This is the default.

- To use the current date, enter **1**.

  For example:

  ```
  - pin_collections_process use_current_time 1
  ```

3. Save and close the file.

Integrating Collections Manager with Custom Client Applications

To enable a custom client application to manage collections activities:

- Configure BRM to send collections notification events to your custom client application. See "Sending Collections Notifications to Custom Client Applications".
- Customize your client application to call the Collections FM opcodes. See "Using the Collections Manager API".
- (Oracle Application Integration Architecture systems only) Configure the **pin_collections_process** utility to publish a **CollectionsInfoChange** business event. See "Creating CollectionsInfoChange Business Events".

Sending Collections Notifications to Custom Client Applications

To configure BRM to send collections notifications:

1. Install the EAI Framework on your system. See "Integrating BRM With Enterprise Applications" in *BRM Developer's Guide*.
2. Install the Synchronization Queue DM on your system. See "Installing and Configuring the Synchronization Queue DM" in *BRM Synchronization Queue Manager*.
3. Configure the Synchronization Queue DM to publish the **CollectionsAction** business event to the database queue by doing the following:
   - Merge the *BRM_home***/sys/eai_js/payloadconfig_collections.xml** file with your existing EAI payload configuration file.
   - Verify that the *BRM_home***/sys/eai_js/Infranet.properties** file points to the correct EAI payload configuration file. You set the path and file name in the **infranet.eai.configFile** entry.
   - Merge the *BRM_home***/sys/data/config/pin_notify_collections** file with your existing *BRM_home***/sys/data/config/pin_notify** file. Load the merged file into the database by using the **load_pin_notify** utility.
   - Uncomment the **CollectionsAction** business event in your *BRM_home***/sys/dm_aq/aq_queuenames** file.

   For more information, see "Installing and Configuring the Synchronization Queue DM" in *BRM Synchronization Queue Manager*.
4. Start the Synchronization Queue DM.

   See "Starting and Stopping the Synchronization Queue DM" in *BRM Synchronization Queue Manager*.

BRM publishes the **CollectionsAction** business event to the database queue whenever an **/event/audit/collections/action** notification event occurs.

Using the Collections Manager API

You can add the following functionality to your custom client application by using the Collections Manager API:

- Manage the collection of overdue balances. See "Managing Overdue Balance Collection".
- Retrieve collections information. See "Retrieving Collections Information".
- Perform system collections activities. See "Performing System Collections Actions".
- Perform manual collections activities. See "Performing Manual Collections Actions".
- Group bill units for collections actions. See "Managing Collections Sharing Groups".
- Configure promise-to-pay agreements. See "Configuring Promise-to-Pay Agreements".

Managing Overdue Balance Collection

Collections Configuration calls the following opcodes to configure collections scenarios, actions, and profiles and to configure user roles for collections personnel. To add this functionality to a custom client application, customize it to accept the information required by the target opcode's input flist and pass it to the appropriate opcode:

- PCM_OP_COLLECTIONS_CONFIG_SET_ACTION. See "Creating or Updating Collections Actions".
- PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE. See "Creating or Updating Collections Profiles".
- PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS. See "Getting All Currently Defined Collections Actions".
- PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES. See "Getting All Currently Defined Collections Profiles".
- PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS. See "Getting All Currently Defined Collections Scenarios".
- PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL. See "Getting Details of a Collections Scenario".
- PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES. See "Getting a List of Message Templates".
- PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION. See "Deleting an Existing Collections Action".
- PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE. See "Deleting an Existing Collections Profile".
- PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO. See "Deleting an Existing Collections Scenario".

Creating or Updating Collections Actions

Use the PCM_OP_COLLECTIONS_CONFIG_SET_ACTION opcode to create or update a collections action.

Collections Configuration calls this opcode when a user creates a new action or modifies an existing action.

As input, this opcode takes the POID of the **/config/collections/action** object to create or update and it takes the action's name and type. This opcode uses the POID ID to determine whether to create a new action object or update an existing action object:

- If the PIN_FLD_POID field is set to **-1**, this opcode creates a new **/config/collections/action** object.
- If the PIN_FLD_POID field is set to the POID of an existing action object, this opcode updates the object.

This opcode stops processing if the POID of the **/config/collections/action** object in the input flist is missing or invalid.

If successful, this opcode returns the POID of the **/config/collections/action** object that was created or updated.

Creating or Updating Collections Profiles

Use the PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE opcode to create or update a collections profile.

Collections Configuration calls this opcode when a user creates or modifies a profile.

As input, this opcode takes the POID of the **/config/collections/profile** object to create or update and it takes the profile's name, description, and the currency used for the profile. This opcode uses the POID ID to determine whether to create a new profile object or update an existing profile object:

- If the PIN_FLD_POID field is set to **-1**, this opcode creates a new **/config/collections/profile** object.
- If the PIN_FLD_POID field is set to the POID of an existing profile object, this opcode updates the object.

This opcode stops processing if the POID of the **/config/collections/profile** object in the input flist is missing or invalid.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns the POID of the **/config/collections/profile** object that was created or updated.

Getting All Currently Defined Collections Actions

Use the PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS opcode to get a list of all currently defined collections actions.

Collections Configuration calls this opcode to retrieve a list of actions and their definitions.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/action** object that contains definitions of collections actions. The object is created when new actions are defined in Collections Configuration.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID, name, description, and type of the actions returned. It also returns the PIN_FLD_OPCODE field, which shows the opcode to use to execute the actions. If any action involves sending a dunning letter, the output flist returns the POID of the template used by this action.

Getting All Currently Defined Collections Profiles

Use the PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES opcode to retrieve a list of currently defined collections profiles.

Collections Configuration calls this opcode to display all currently defined profiles.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/profile** object. This object is created when a new collections profile is defined in Collections Configuration.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID, name, description, and the currency used for each profile found.

Getting All Currently Defined Collections Scenarios

Use the PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS opcode to get a list of all collections scenarios and associated profiles in the current brand.

Collections Configuration calls this opcode to list all currently defined scenarios and profiles.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/scenario** object. This object is created when a new collections scenario is defined in Collections Configuration.

This opcode also retrieves the **/config/collections/profile** object, which contains definitions of profiles that are associated with scenarios.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID and name of each scenario and the POID of the brand to which each scenario belongs. It also returns the POID and name of the profile associated with each scenario.

Getting Details of a Collections Scenario

Use the PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL opcode to get details of a particular collections scenario.

Collections Configuration calls this opcode to display details of the selected scenario.

As input, this opcode takes the POID of the **/config/collections/scenario** object.

This opcode stops processing if the POID of the **/config/collections/scenario** object in the input flist is missing or invalid.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns a results array that contains the POID of the **/config/collections/scenario** object, the name of the scenario, and the POID of the profile associated with the scenario.

It also returns an array that contains details for the action associated with the scenario, including the action object POID, name, description, type, and the PIN_FLD_OPCODE field, which shows the opcode to use to execute the action. If the action involves sending a dunning letter, the output flist returns the POID of the template used by this action.

Getting a List of Message Templates

Use the PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES opcode to get a list of templates for the current brand.

When a user creates or updates an action that requires a template, Collections Configuration calls this opcode to display a list of available templates.

As input, this opcode takes the action type, the template's location, and a dummy POID used to get the database ID.

The opcode performs the following:

- Reads the action type from the input flist. The PIN_FLD_ACTION_TYPE field specifies DUNNING_LETTER.
- Reads the **/config/invoice_templates/dunning** objects to retrieve dunning letter templates. BRM creates these objects and automatically creates subclasses of **/config/invoice_templates** when loading the dunning letter templates.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns an array containing the POID, brand, and name of each template in the current brand that matches the locale and action type specified in the input flist.

Deleting an Existing Collections Action

Use the PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION opcode to delete an existing collections action.

Collections Configuration calls this opcode when a user deletes a collections action.

This opcode takes the POID of the **/config/collections/action** object to delete, verifies that the object is not used by any scenario, and deletes the object.

This opcode stops processing if:

- The POID of the **/config/collections/action** object in the input flist is missing or invalid.
- The action object is used by a scenario.

If this opcode is not successful, it sets the PIN_FLD_RESULT field to **Fail** in the output flist. If the scenario is in use, an appropriate message is displayed in Collections Configuration.

If successful, this opcode returns the POID of the **/config/collections/action** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Deleting an Existing Collections Profile

Use the PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE opcode to delete an existing collections profile.

> **Note:** Users cannot delete a default profile created during the installation of Collections Configuration.

Collections Configuration calls this opcode when a user deletes a collections profile.

This opcode takes the POID of the **/config/collections/profile** object to delete, verifies that the object is not used by any scenario, and deletes the object.

This opcode stops processing if:

- The POID of the **/config/collections/profile** object in the input flist is missing or invalid.
- The object is used by a scenario.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/config/collections/profile** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Deleting an Existing Collections Scenario

Use the PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO opcode to delete an existing collections scenario.

Collections Configuration calls this opcode when a user deletes a scenario.

This opcode takes the POID of the **/config/collections/scenario** object to delete, verifies that the object is not used by any bill units that are currently in collections and was not used by any bill units that were previously in collections, and deletes the object.

This opcode stops processing if:

- The POID of the **/config/collections/scenario** object in the input flist is missing or invalid.
- The object is currently used by one or more bill units in collections.
- The object was previously used by one or more bill units that entered into collections. The collections scenario cannot be deleted even after a bill unit has exited from collections because the association between the collections scenario and the collections objects still remains.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/config/collections/scenario** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Retrieving Collections Information

Collections Center calls the following opcodes to retrieve information about collections actions. To add this functionality to a custom client application, customize it to accept the information required by the target opcode's input flist and pass it to the appropriate opcode.

- PCM_OP_COLLECTIONS_GET_BILLINFOS. See "Retrieving a List of Bill Units in Collections".
- PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS. See "Retrieving Aging Buckets Information".
- PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL. See "Retrieving Scenario Information".
- PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS. See "Getting All Valid Collections Scenarios".
- PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS. See "Retrieving a List of Collections Actions".
- PCM_OP_COLLECTIONS_GET_ACTION_HISTORY. See "Retrieving Collections Action History Information".

Retrieving a List of Bill Units in Collections

Use the PCM_OP_COLLECTIONS_GET_BILLINFOS opcode to get a list of bill units (**/billinfo** objects) that are in collections.

Collections Center calls this opcode to display the bill units in collections that meet the criteria defined by a CSR. The search criteria passed in by Collections Center are included in the optional fields that determine the scope of the search.

The input flist must contain a dummy POID, which is used to get the database ID for the search. To retrieve bill units based on specific criteria, you can use the following optional input:

- The ID of the bill unit
- The status of the bill unit
- The payment type of the bill unit
- A date range; this retrieves bill units whose bills were generated between the specified dates
- The status of the account
- The account number (PIN_FLD_ACCOUNT_NO); this narrows the search to a particular account whose bill units are in collections
- The name of the company to which an account belongs
- The name of the customer on the account
- The account POID of the agent assigned to the bill units
- The name of the collections scenario assigned to the bill units
- The name of the collections profile associated with the bill units
- The overdue amount range
- The overdue days range; this narrows the search to bill units that are overdue for the specified number of days
- A flag indicating whether to get assigned bill units or unassigned bill units

If the opcode fails, a NULL flist is returned.

If successful, this opcode returns a list of bill units that meet the search criteria and details for the fields specified on the input flist.

> **Note:** This opcode uses step search when searching for bill units. See "Setting the Number of Bill Units Retrieved during Step Searches" to specify the number of bill units retrieved in each step.

Retrieving Aging Buckets Information

Use the PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS opcode to retrieve a bill unit's aging buckets details.

Collections Center calls this opcode when displaying the distribution of a bill unit's overdue balance over a number of aging buckets.

This opcode takes the POID of the bill unit and performs the following actions:

- Retrieves aging bucket information, which includes the total number of buckets and the number of overdue days per bucket from the **/config/collections/aging_buckets** object.
- Determines the exact number of overdue days per each aging bucket for the specified bill unit.
- Calculates the amount due in each bucket.

This opcode stops processing if the bill unit information in the input flist is missing or invalid.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/billinfo** object, an array of the aging buckets, the number of overdue days for each bucket, and the amount due in each bucket.

Retrieving Scenario Information

Use the PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL opcode to retrieve details about a bill unit's collections scenario.

Collections Center calls this opcode to display the collections activity details that CSRs use to work with the collections actions that are scheduled for a bill unit.

This opcode takes the POID of the **/collections_scenario** object and performs the following actions:

- Reads the **/collections_scenario** object information.
- Retrieves a list of actions for the specified scenario.
- Retrieves history information for each action.

This opcode stops processing if the input flist does not include a valid POID for a **/collections_scenario** object.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the output flist that contains the POID of the bill unit, the POID of the scenario assigned to the bill unit, scenario details, and an array listing the POID and status of each scheduled action.

Getting All Valid Collections Scenarios

Use the PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS opcode to evaluate all the collections scenarios for the bill unit using the default entry criteria (**entry_amount** and **entry_days**), the severity attribute, and additional configurable parameters and list the valid scenarios for the bill unit.

Collections Center calls this opcode to display the list of valid scenarios during scenario assignment or replacement.

This opcode takes the entry criteria (**entry_amount** and **entry_days**), the severity attribute, and additional configurable parameters and performs the following actions:

- Searches for collections scenarios matching the criteria.
- For each collections scenario, calls the PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS policy opcode. The policy opcode takes the scenario as input, reads the **/config/collections/scenario_params** object for the additional parameters, and validates whether the scenario is valid for the bill unit in collections. The valid collections scenario is added to the list of valid collections scenarios. If the collections scenario is not valid, the policy opcode evaluates the next collections scenario.
- Returns a list of all the valid scenarios.

Retrieving a List of Collections Actions

Use the PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS opcode to retrieve a list of collections actions assigned to collections agents.

Collections Center calls this opcode when collections managers request an overview of the workload for the collections agents they supervise.

The input flist contains the following:

- The agent's account POID. This returns a list of actions that are currently assigned to the specified agent. If you specify the account POID as type-only, this opcode retrieves the actions for all collections agents.
- PIN_FLD_START_T and PIN_FLD_END_T are optional. You specify one of these fields to limit the actions returned to those that are due either after the start time or before the end time.
- The PIN_FLD_THRESHOLD field specifies the number of actions to retrieve. The default is all actions.

This opcode returns an array of details for the collections action. The return data includes the agent's name and account POID, the POID of the action object, the action's name and due date, and the opcode to execute for the action.

> **Note:** This opcode uses step search when searching for bill units. See "Setting the Number of Bill Units Retrieved during Step Searches" to specify the number of bill units retrieved in each step.

Retrieving Collections Action History Information

Use the PCM_OP_COLLECTIONS_GET_ACTION_HISTORY opcode to find past information about a particular collections action.

Collections Center calls this opcode to display details about when an action was assigned to a collections agent, reassigned, rescheduled, and so on.

As input, this opcode takes the POID of the **/collections_action** object for which to retrieve historic data.

This opcode stops processing when the input flist does not include a valid POID for a **/collections_action** object.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/collections_action** object and a results array with history details for the collections action. The array includes the action's status and due date, the assigned agent's name and account POID, the date when the last change occurred, and the description of the change.

Performing System Collections Actions

Collections Manager performs a variety of actions on bill units with overdue balances. Some actions happen automatically, and others require manual intervention.

Automatic actions are performed by the system. They include calculating overdue balances, applying late fees, and updating bill unit balances.

Collections Manager uses the following opcodes to perform system actions on the bill units in collections:

- PCM_OP_COLLECTIONS_PROCESS_BILLINFO. See "Executing Automatic Collections Actions".
- PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER. See "Preparing Invoice Reminders".
- PCM_OP_COLLECTIONS_SET_DUNNING_LETTER. See "Gathering and Storing Data for Dunning Letters".
- PCM_OP_COLLECTIONS_GET_DUNNING_LETTER. See "Retrieving Dunning Letters".
- PCM_OP_COLLECTIONS_TAKE_ACTION. See "Executing Pending Actions for a Bill Unit".
- PCM_OP_COLLECTIONS_PUBLISH_EVENT. See "Passing Information to Custom Client Applications".

Executing Automatic Collections Actions

Use the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode to:

- Determine whether an account's bill unit should enter or exit the collections process.
- Perform collections actions, such as sending notices to customers or applying late charges.

The **pin_collections_process** utility calls this opcode for each bill unit with an overdue balance that meets a minimum amount (see "Setting the Minimum Overdue Balance to Process"). This opcode then processes each bill unit to evaluate its collections status and perform any necessary collections actions.

When this opcode receives a bill unit, it performs the following actions:

- Calculates the exact overdue balance and days late for the bill unit.
- Checks the bill unit's current collections status.

  If the bill unit is *not* in collections, this opcode:

  - Calls the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode to assign the bill unit to a collections profile. See "Mapping Bill Units to Collections Profiles".
  - Maps the collections profile to a collections scenario.
  - Creates a **/collections_scenario** object and a **/collections_action** object for each collections activity that needs to be performed.
  - Sets the status of all actions in the collections scenario. If collections action dependencies are enabled, sets the first action's status to **Pending** and all subsequent actions' status to **Waiting For Dependents**. If multiple actions are scheduled for the first due date, sets the status of those actions to **Pending** and sets the status of actions scheduled for later due dates to **Waiting For Dependents**.

    If collections action dependencies are disabled, sets the status of all actions to **Pending**. See "Creating Dependencies between Collections Actions in a Scenario".
  - Generates the **/event/audit/collections/action** notification event.

  - Calls the PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT policy opcode to assign a collections agent to the bill unit. See "Assigning Bill Units Automatically".

  If the bill unit *is already* in collections, this opcode sets the bill unit's action status to REMAIN_IN and updates the bill unit's overdue balance and days late.

- Determines whether the bill unit meets the exit criteria:

  If it *meets* the exit criteria, this opcode changes the bill unit's action status to **Completed**, generates the **/event/audit/collections/action** notification event, and calls the PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO policy opcode to perform any custom actions that you specify. (See "Performing Custom Actions when a Bill Unit Leaves Collections".) Processing is complete.

  If it *does not meet* the exit criteria, this opcode:

  - Finds all of the bill unit's actions that are set to **Pending**.
  - If a *manual* action is required, generates the **/event/audit/collections/action** notification event. Processing is complete.
  - If a *system or custom* action is required, calls the PCM_OP_COLLECTIONS_TAKE_ACTION opcode to execute the action. See "Executing Pending Actions for a Bill Unit".

- Evaluates the bill unit to determine if it now meets the exit criteria for the scenario:

  If the bill unit *does not meet* the exit criteria, processing ends for this bill unit.

  If the bill unit *does meet* the exit criteria, this opcode:

  - Changes the bill unit's action status to **Completed**.
  - Calls the PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO policy opcode to perform any custom exit actions that you specify. See "Performing Custom Actions when a Bill Unit Leaves Collections".
  - Generates the **/event/audit/collections/action** notification event.

Preparing Invoice Reminders

Use the PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER opcode to prepare a reminder message for customers with overdue payments. This message is later added to your invoices or custom documents via the Universal Message Store (UMS) framework. See "Using BRM Messaging Services" in *BRM Developer's Guide*.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode when a collections scenario calls for an invoice reminder.

This opcode performs the following actions:

- Calls PCM_OP_UMS_SET_MESSAGE to create a **/message** object.
- Returns the POID of the **/collections_action/invoice_reminder** object and the status to PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Gathering and Storing Data for Dunning Letters

Use the PCM_OP_COLLECTIONS_SET_DUNNING_LETTER opcode to gather and store data for dunning letters.

This opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode when a collections scenario requires a dunning letter.

This opcode performs the following actions:

- Calls the PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA policy opcode to retrieve data for the dunning letter and to perform any customizations that you implement. See "Customizing Dunning Letters".
- Converts the data into XML format and saves it as a buffer field in the **/collections_action/dunning_letter** object for this letter.
- Returns the POID of the **/collections_action/dunning_letter** object and the status to the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode.

Retrieving Dunning Letters

Use the PCM_OP_COLLECTIONS_GET_DUNNING_LETTER opcode to retrieve dunning letters from the BRM database.

This opcode is called directly by the **pin_collections_send_dunning** utility.

This opcode performs the following actions:

- Retrieves the **/collections_action/dunning_letter** object from the BRM database. This object contains the XML data to be included in the letter.
- Retrieves the **/config/invoice_templates/dunning** object from the BRM database. This object contains the XSLT style sheet used to format the dunning letter.

> **Note:** Dunning letter templates are stored as **/config/invoice_templates/dunning** objects. The **/config/invoice_templates** class is automatically subclassed when you load a dunning letter template.

- Calls the PCM_OP_INV_FORMAT_INVOICE opcode to create the final dunning letter.

Executing Pending Actions for a Bill Unit

Use the PCM_OP_COLLECTIONS_TAKE_ACTION opcode to execute pending actions for a bill unit.

This opcode is called by either the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode or the **pin_deferred_act** utility to execute actions.

As input, this opcode takes the POID of the action to execute, the POID of the bill unit to process, and the POID of the account that owns the bill unit. This opcode then determines the type of action that needs to be executed:

- If it requires a *manual* action, this opcode stops processing.
- If it requires a *system* action, this opcode calls other opcodes to execute the desired action:
  - Prepare a dunning letter. See "Gathering and Storing Data for Dunning Letters".
  - Apply finance charges. See "Applying Finance Charges".
  - Apply late fees. See "Applying Late Fees".
- If it requires a *custom* action, this opcode calls the PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION policy opcode to perform any custom actions that you specify. See "Performing Custom Collections Actions".

Then, this opcode sets the action's status to **Completed** and updates the bill unit's overdue date and overdue amount. This opcode generates an **/event/audit/collections/action** notification event.

If collections action dependencies are enabled, this opcode also changes the subsequent action's status from **Waiting For Dependents** to **Pending** and reschedules the due dates of any outstanding collections actions in the scenario.

> **Note:** If multiple actions are scheduled for the same day and are all set to **Pending**, this opcode waits until all of those actions are completed or canceled before changing the subsequent action's status from **Waiting For Dependents** to **Pending**.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/collections_action** object for this action and the status of action.

Passing Information to Custom Client Applications

Use the PCM_OP_COLLECTIONS_PUBLISH_EVENT opcode to append additional information to the **/event/audit/collections/action** notification event before it is passed to your custom client application.

This opcode is called by the event notification system whenever the **/event/audit/collections/action** notification event occurs.

This opcode calls the PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT policy opcode and then publishes the notification event to the Payload Generator External Module (EM).

For information about the PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT policy opcode, see "Adding Information that Is Passed to Custom Client Applications".

Performing Manual Collections Actions

Collections Manager performs a variety of actions on bill units with overdue balances. Some actions happen automatically; others require manual intervention.

Manual actions are performed by collections agents or collections managers who use Collections Center to work with bill units. The actions include adding actions to a bill unit scenario, exempting bill units from collections, and assigning bill units to agents. You can configure your client application to call the opcodes to execute manual actions.

Collections Center calls the following opcodes after a manual action is performed. To enable your custom client application to update Collections Manager after manual actions occur, customize it to accept the information required by the target opcode's input flist and pass the information to the target opcode.

- PCM_OP_COLLECTIONS_ASSIGN_AGENT. See "Assigning Bill Units to a Collections Agent".
- PCM_OP_COLLECTIONS_ADD_ACTION. See "Adding Actions to a Collections Scenario".
- PCM_OP_COLLECTIONS_EXEMPT_BILLINFO. See "Exempting Bill Units from Collections".
- PCM_OP_COLLECTIONS_RESCHEDULE_ACTION. See "Rescheduling an Action Scheduled for a Bill Unit".
- PCM_OP_COLLECTIONS_SET_ACTION_STATUS. See "Changing the Status of a Collections Action".

- PCM_OP_COLLECTIONS_REPLACE_SCENARIO. See "Replacing a Collections Scenario".

Assigning Bill Units to a Collections Agent

Use the PCM_OP_COLLECTIONS_ASSIGN_AGENT opcode to assign bill units to a collections agent.

Collections Center calls this opcode when a collections manager assigns a bill unit to a particular agent.

This opcode takes as input the POID of the agent's account and the PIN_FLD_BILLINFO array, which specifies the bill units to assign to the agent.

This opcode performs the following actions:

- Assigns each bill unit in the PIN_FLD_BILLINFO array to the agent.
- Updates bill unit scenarios with the agent's account POID.
- Returns the account POID of the agent responsible for the bill units and an array of the bill units assigned to the specified agent.

This opcode stops processing if the bill unit information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the account POID of the agent responsible for the bill units and an array of the bill units assigned to the agent.

Adding Actions to a Collections Scenario

Use the PCM_OP_COLLECTIONS_ADD_ACTION opcode to add collections actions to a bill unit's collections scenario.

Collections Center calls this opcode when CSRs add actions to a collections scenario.

This opcode takes as input a dummy POID, the **/collections_scenario** POID, the CSR's login account POID, the new action object's POID, and the action's due date.

This opcode performs the following actions:

- Checks the CSR account POID to make sure the user is authorized to add actions.
- Adds a new action to a scenario and updates the output flist with the new action POID.
- Sets the new action's status. If collections action dependencies are enabled, checks whether the subsequent action's status is set to **Pending**. If so, sets the new action's status to **Pending** and updates the subsequent action's status to **Waiting For Dependents**. If not, sets the new action's status to **Waiting For Dependents**.

  Note: Actions cannot be inserted before, between, or on the same day as any canceled or completed actions.

- Checks to see if the PIN_FLD_DAYS field is present in the input flist. If so, this opcode checks the contents of the field to determine whether the actions that follow need to be rescheduled.
  - If PIN_FLD_DAYS is equal to **0**, the actions are not rescheduled.
  - If PIN_FLD_DAYS is less than **0**, the actions are canceled.

  - If PIN_FLD_DAYS is greater than **0**, the value specified indicates the number of days the actions will be delayed.
- Checks to see if the type of collections action is **Collect Payment** and the Credit Card or Debit Card credentials are specified and does the following:
  - Creates a **/payinfo** object with the credentials in the input flist.
  - Creates the **/collections_action/collect_payment** action to hold the amount to be paid and a link to **/payinfo** object.
- Checks to see if the type of collections action is **Promise to Pay** and does the following:
  - Validates if the promise-to-pay agreement is already called for the bill unit and the due date is not before INVOKE_T of the active promise.
  - Updates the outstanding amount under the **/collections_scenario** object with the amount for the new payment milestone.
- Updates the output flist with the new data for each rescheduled action.

This opcode stops processing if:

- The CSR account POID does not match the POID of a user who has permission to add actions.
- The action information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns PIN_FLD_POID set to the POID of the new action added and updates the output flist with the new data for each rescheduled action.

Exempting Bill Units from Collections

Use the PCM_OP_COLLECTIONS_EXEMPT_BILLINFO opcode to exempt bill units from collections.

Collections Center calls this opcode when a CSR exempts a bill unit from collections.

This opcode takes as input the POID of the bill unit to exempt from collections and the POID of the account that owns the bill unit.

If the bill unit is already in collections, this opcode performs the following actions:

- Calls the PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO policy opcode to remove the bill unit from the scenario.
- Cancels all pending and error actions scheduled for this bill unit.
- Sets the bill unit's PIN_FLD_SCENARIO_OBJ field to NULL to indicate that this bill unit is not in collections.
- Sets the exemption flag so that the bill unit never enters collections, even if it has an overdue balance.

If the bill unit is not in collections, this opcode sets the exemption flag so that the bill unit never enters collections, even if it has an overdue balance.

This opcode stops processing if the input flist does not include a valid bill unit POID.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the exempted **/billinfo** object and the POID of the account to which the bill unit belongs.

Rescheduling an Action Scheduled for a Bill Unit

Use the PCM_OP_COLLECTIONS_RESCHEDULE_ACTION opcode to reschedule an action that was scheduled by a bill unit's collections scenario.

Collections Center calls this opcode when a CSR reschedules a particular action to be performed on a bill unit. For example, if a customer promises payment by a certain day, the CSR can reschedule the inactivation of the bill unit.

This opcode takes as input the POID of the action to be rescheduled, the account POID of the CSR that is rescheduling the action, and the new due date for the action.

This opcode performs the following actions:

- Checks the account POID to verify that the CSR is authorized to reschedule the action. If the account POID matches the account of the agent who originally scheduled the action or if it is the account of a manager, the opcode allows rescheduling.
- Reads the PIN_FLD_DUE_T input field to obtain a new due date for the action.
- Checks to see if the PIN_FLD_DAYS field is present in the input flist. If so, this field determines whether the actions that follow are postponed.
  - If PIN_FLD_DAYS is equal to **0**, the actions are not postponed.
  - If PIN_FLD_DAYS is less than **0**, the actions are canceled.
  - If PIN_FLD_DAYS is greater than **0**, the value indicates the number of days to postpone the actions.
- Updates the output flist with the new data for each action.

This opcode stops processing if:

- The CSR's account POID does not match the POID of an authorized user.
- The action information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the POID of the rescheduled action and updates the output flist with the new data for each action affected by the rescheduling.

Changing the Status of a Collections Action

Use the PCM_OP_COLLECTIONS_SET_ACTION_STATUS opcode to change the status of a manual collections action.

Collections Center calls this opcode when a CSR updates the status of a manual action. For example, after finishing a phone call, an agent must update the action's status to **Completed**.

This opcode takes as input the POID of the action object and the new status for the action.

If collections action dependencies are enabled, this opcode performs the following actions:

- Checks whether the action's current status is set to **Pending**. If so, it changes the current action's status and then updates the subsequent action's status from **Waiting For Dependents** to **Pending**. If not, it generates an error.

- If the action's status is changed to **Cancelled**, checks the value of the PIN_FLD_FLAGS flist field and the **/collections_action** object's PIN_FLD_ACTION_MODE field:
  - If PIN_FLD_ACTION_MODE is set to **0**, the action is mandatory. The action cannot be canceled and system actions cannot be completed from Collections Center. The opcode generates an error.
  - If PIN_FLD_ACTION_MODE is set to **1** and PIN_FLD_FLAGS is set to **0**, the opcode changes the action's status to **Cancelled**.
  - If PIN_FLD_ACTION_MODE is set to **1** and PIN_FLD_FLAGS is set to **1**, the opcode changes the action's status to **Cancelled** and the status of all actions that follow to **Cancelled**.
- If the action's status is changed to **Completed**, checks the status of the PIN_FLD_FLAGS flist field:
  - If it is set to **0**, the opcode changes the action's status to **Completed** and reschedules the due dates of all outstanding actions in the scenario.
  - If it is set to **2**, the opcode changes the action's status to **Completed** and leaves the due dates of all outstanding actions unchanged.

If collections action dependences are disabled:

- Changes the current action's status.
- If the action status is changed to **Cancelled**, checks the value of the PIN_FLD_FLAGS flist field:
  - If it is set to **0**, the opcode changes the action's status to **Cancelled**.
  - If it is set to **1**, the opcode changes the action's status to **Cancelled** and the status of all actions that follow to **Cancelled**.

This opcode stops processing if the POID of an action is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the POID of the action whose status was changed.

Replacing a Collections Scenario

Use the PCM_OP_COLLECTIONS_REPLACE_SCENARIO opcode to replace the existing collections scenario for a bill unit with a new collections scenario. You can replace the scenario of a bill unit that is already in collections.

Collections Configuration calls this opcode to replace the scenario of a bill unit that is already in collections.

As input, this opcode takes the POID of the bill unit already in collections.

This opcode creates the **/event/activity/collections/replace_scenario** activity event object to store the bill unit details such as the bill unit POID, the current collections scenario POID, and the new collections scenario POID. If the new collections scenario object is not specified, this opcode calls the PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS opcode to get a list of all the valid collections scenarios applicable for the bill unit.

After the new scenario is identified, this opcode exits out of the existing collections scenario, cancels all the pending actions, deletes the scheduled objects for the pending actions, and updates the bill unit with the POID of the new collections scenario.

Managing Collections Sharing Groups

Customer Center calls the following opcodes to group bill units into a collections sharing group. For more information, see "About Collections Sharing Groups". To add this functionality to a custom client application, customize it to accept the information required by the target opcode's input flist and pass it to the appropriate opcode:

- PCM_OP_COLLECTIONS_GROUP_CREATE. See "Creating a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER. See "Adding Members to a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER. See "Removing a Member from a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_DELETE. See "Deleting a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO. See "Retrieving a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_MODIFY. See "Modifying a Collections Sharing Group".
- PCM_OP_COLLECTIONS_GROUP_SET_PARENT. See "Changing the Parent of a Collections Sharing Group".

Creating a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_CREATE opcode to group bill units into a collections sharing group.

This opcode is called directly by Customer Center.

This opcode takes as input the **/account** POID and the **/billinfo** POID of the group owner, the **/billinfo** POIDs of the members of the group, and the name of the collections sharing group.

This opcode performs the following actions:

- Validates that the parent bill unit is not already a parent or member of another collections sharing group.
- Calls the PCM_OP_GROUP_CREATE_GROUP opcode to create the **/group/collections_targets** object. When a group is created, events are created.

  The **/event/group/collections_targets** event contains details of a parent and members in the PIN_FLD_MEMBERS array.

  The **/event/group/parent** event contains details of a parent.
- Calls the PCM_OP_GROUP_ADD_MEMBER opcode to add members to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, this opcode returns the POID of the **/group/collections_targets** object.

Adding Members to a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER opcode to add a child member to an existing collections sharing group.

This opcode is called directly by Customer Center.

This opcode takes as input the POID of the **/group/collections_targets** object, the **/billinfo** POID of the new member, and the name of the collections sharing group.

This opcode performs the following actions:

- Validates that the new child member is not a parent or member of the existing collections sharing group.
- Calls the PCM_OP_GROUP_ADD_MEMBER opcode to add the bill unit to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, this opcode returns the POID of the **/group/collections_targets** object.

Removing a Member from a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER opcode to remove a member from an existing collections sharing group.

This opcode is called directly by Customer Center.

This opcode takes as input the POID of the **/group/collections_targets** object and the POID of the **/billinfo** object to remove the member, and returns the POID of the **/group/collections_targets** object.

Deleting a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_DELETE opcode to delete an existing collections sharing group.

This opcode is called directly by Customer Center.

This opcode takes as input the POID of the **/group/collections_targets** object to delete, calls PCM_OP_GROUP_DELETE_GROUP opcode to delete the specified object, and returns the POID of the **/group/collections_targets** object.

If the collections group is already in collections, the opcode throws an error.

Retrieving a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO opcode to retrieve:

- Whether the specified bill unit is a member of a collections sharing group.
- The pending receivables for each bill unit. If a bill unit is a parent of a collections sharing group, it also provides the name of the collections sharing group and its child member bill units.

This opcode is called directly by Collections Center.

This opcode takes as input the **/account** or **/billinfo** POID and performs the following actions:

- If the **/billinfo** POID is not passed in, retrieves all **/billinfo** objects associated with the **/account** object.
- For each **/billinfo** object, reads the associated **/group/collections_targets** object to determine whether the bill unit belongs to a collections sharing group and, if it does, whether it is a parent bill unit or a child bill unit.
  - If it is not a member of a collections sharing group, returns the **/group/collections_targets** POID and the PIN_FLD_BOOLEAN field set to **0**.
  - If it is a child in a collections sharing group, returns the **/group/collections_targets** POID and the PIN_FLD_BOOLEAN field set to **1**.

- If it is a parent in a collections sharing group, returns the **/group/collections_targets** POID, the PIN_FLD_BOOLEAN field set to **1**, and the pending receivables of all group members.

Modifying a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_MODIFY opcode to modify an existing collections sharing group, such as renaming the collections sharing group, changing the parent member, and replacing all of the existing child members.

This opcode is called directly by Customer Center.

This opcode takes as input the POID of the **/group/collections_targets** object and the information to change, modifies the collections sharing group, and returns the POID of the **/group/collections_targets** object.

Changing the Parent of a Collections Sharing Group

Use the PCM_OP_COLLECTIONS_GROUP_SET_PARENT opcode to change a collections sharing group's existing parent bill unit.

This opcode is called directly by Customer Center.

This opcode takes as input the POID of the **/group/collections_targets** object, the name of the calling program, and the POID of the new parent bill unit.

This opcode performs the following actions:

- Validates that the new parent bill unit is not a child of the existing collections sharing group.
- Validates that the new parent bill unit is not already a parent of another collections sharing group.
- Calls the PCM_OP_GROUP_SET_PARENT opcode to assign the new parent bill unit to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, this opcode returns the POID of the **/group/collections_targets** object.

Configuring Promise-to-Pay Agreements

Collections Center calls the following opcodes to configure the promise-to-pay agreements. To add this functionality to a custom client application, customize it to accept the information required by the target opcode's input flist and pass it to the appropriate opcode.

- PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY. See "Creating a Promise-to-Pay Action".
- PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS. See "Updating Amount and Payment Details".
- PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY. See "Revoking a Promise-to-Pay Agreement".
- PCM_OP_COLLECTIONS_RESCHEDULE_ACTION. See "Rescheduling a Promise-to-Pay Action".
- PCM_OP_COLLECTIONS_SET_ACTION_STATUS. See "Changing the Status of a Promise-to-Pay Action".

Creating a Promise-to-Pay Action

For a bill unit (**/billinfo** object) that is already in collections, use the PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY opcode to create the promise-to-pay action.

This opcode calls the PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY policy opcode. By default, this policy opcode is an empty hook.

Collections Center calls this opcode when a customer requests a promise-to-pay agreement for a bill unit that is in collections.

This opcode takes as input:

- The number of payment milestones (PIN_FLD_NUM_MILESTONES) or the amount per milestone (PIN_FLD_MILESTONE_AMOUNT).
- The total number of days during which the full outstanding amount will be paid off (PIN_FLD_DAYS) or the interval in days between each payment milestone (PIN_FLD_MILESTONE_INTERVAL).
- (Optional) The date for the first payment milestone. If this is not specified, the current date is used.

This opcode performs the following actions:

- Creates **/collections_action/promise_to_pay** actions for the payment milestones and updates the action due dates and milestone amounts. This **/promise_to_pay** action is attached to the existing scenario.
- Reschedules the pending collections actions by one day after the last milestone due date.
- Generates an **/event/audit/collections/actions** audit event to record the promise-to-pay details.

You can add additional milestones and **collect_payment** actions to an existing collections scenario using Collections Center.

Updating Amount and Payment Details

Use the PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS opcode to update the amount and payment details for a collections action.

Collections Center calls this opcode to update the amount and payment details for the **Promise to Pay** and **Collect Payment** actions.

If the input action type is **Promise to Pay** and the amount details are updated, the opcode updates the total outstanding amount in the **/collections_scenario** object with the difference amount.

If the input action type is **Promise to Pay** and the payment details are updated, depending on the option whether the updated details is applicable only to the current installment or includes the next installments, the opcode updates the details.

When calling this opcode, send only the data that needs to be updated. For example, in a scenario where only the amount details should be updated, if the payment details are also sent in the input flist with NULL values, the opcode updates the payment details of the current action with the NULL values.

Revoking a Promise-to-Pay Agreement

Use the PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY opcode to revoke the promise-to-pay agreement.

Collections Center calls this opcode when the customer cancels the promise-to-pay agreement.

As input, this opcode takes the POID of the **/event/activity/collections/promise_to_pay** object and performs the following actions:

- Cancels the outstanding payment milestones and deletes the corresponding schedule objects.
- Reschedules the scenario actions to start from the next day and updates the corresponding schedule objects.
- Removes the reference of **/event/activity/collections/promise_to_pay** object from the collections scenario object.

Rescheduling a Promise-to-Pay Action

Use the PCM_OP_COLLECTIONS_RESCHEDULE_ACTION opcode to reschedule a **Promise to Pay** action that was scheduled by a bill unit's collections scenario.

See "Rescheduling an Action Scheduled for a Bill Unit" for more information about rescheduling an action.

Changing the Status of a Promise-to-Pay Action

Use the PCM_OP_COLLECTIONS_SET_ACTION_STATUS opcode to change the status of a **Promise to Pay** action.

See "Changing the Status of a Collections Action" for more information about changing the status of a promise-to-pay action.

Creating CollectionsInfoChange Business Events

If your custom application connects to BRM through Oracle Application Integration Architecture (Oracle AIA), the Oracle Advance Queue (AQ) needs to be notified with a **CollectionsInfoChange** business event when a collections job has run.

To create a **CollectionInfoChange** business event:

1. Open the *BRM_home*/**apps/pin_collections/pin.conf** file in a text editor.
2. Add a **publish_run details** entry and set it to **1**:

   ```
   - pin_collections_process publish_run_details 1
   ```

3. Save and close the file.

Customizing Collections Manager

Use the following policy opcodes to customize Collections Manager:

- PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA. See "Customizing Dunning Letters".
- PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES. See "Applying Finance Charges".
- PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES. See "Applying Late Fees".
- PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT. See "Assigning Bill Units Automatically".
- PCM_OP_COLLECTIONS_POL_ASSIGN_DCA. See "Assigning Bill Units to a Debt Collections Agency".

- PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS. See "Customizing to Which Collections Group Members to Apply Collections Actions".
- PCM_OP_COLLECTIONS_POL_SELECT_PROFILE. See "Mapping Bill Units to Collections Profiles".
- PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT. See "Adding Information that Is Passed to Custom Client Applications".
- PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION. See "Performing Custom Collections Actions".
- PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO. See "Performing Custom Actions when a Bill Unit Leaves Collections".

Customizing Dunning Letters

By default, the PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA policy opcode gathers the following data for your dunning letters:

- Current overdue amount
- Currency type
- Current bill unit overdue date
- Account POID

However, you can modify the type of data gathered by customizing the policy opcode. For example, you can enrich the standard data with additional information, such as the date on which the account will be inactivated.

Important: By default, this policy opcode gathers the data required for typical dunning letter templates, such as for the samples in *BRM_home*/**sys/data/config/stylesheets**. If you use custom templates with placeholders for non-default data, you must customize this policy opcode.

This policy opcode is called by the PCM_OP_COLLECTIONS_SET_DUNNING_LETTER opcode.

Applying Finance Charges

Use the PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES policy opcode to apply finance charges to overdue amounts.

This policy opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode to calculate the finance charge when the scenario associated with the bill unit calls for finance charges.

By default, its calculation is based on the finance charge action definition entered in Collections Configuration. The finance charge percentage from Collections Configuration is stored in the **/collections_action/finance_charge** object, which is created when the account bill unit enters a scenario that includes a finance charge action.

Required inputs include the POID of the **/collections_action/finance_charge** object for this finance charge as well as the POID of the **/billinfo** object, POID of the account with which this **/billinfo** object is associated, overdue amount, and overdue date.

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

You can also customize this policy opcode to find a specific balance group where finance charges apply. By default, BRM applies finance charges to the bill unit's default balance group.

By default, errors in the syntax of the input flist cause this policy opcode to fail. Customizing this policy opcode may introduce additional error conditions.

By default, this policy opcode returns the POID of the **/collections_action/finance_charge** object, the POID of the **/billinfo** object, the action status, and the PIN_FLD_EVENT substruct. After customization, the return value depends on the features implemented.

If this policy opcode fails, it returns an error in the error buffer. The transaction is rolled back.

Applying Late Fees

Use the PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES policy opcode to apply a late fee to overdue charges.

This policy opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode when the scenario associated with a bill unit calls for late fees. By default, it uses the late fee action definition entered in Collections Configuration. The late fee amount or percentage is stored in the **/collections_action/late_fee** object, which is created when the account bill unit enters a scenario that includes a late fee action.

Required inputs include the POID of the **/collections_action/late_fee** object for this late fee, the POID of the **/billinfo** object or the POID of the account to which the bill unit belongs, the overdue amount, and the overdue date.

This policy opcode performs the following tasks by default:

- Gathers the late fee definition information and, if it is based on a percentage, calculates the amount.
- Converts currency, if necessary.

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

You can also customize this policy opcode to find specific balance groups where late fees apply. By default, BRM applies late fees to the default balance group of the bill unit.

By default, errors in the input flist cause this policy opcode to fail. Customizing this policy opcode may introduce additional error conditions.

By default, this policy opcode returns the POID of the **/collections_action/late_fee** object and the status of the action. After customization, the return value depends on the features implemented.

If this policy opcode fails, it returns an error in the error buffer. The transaction is rolled back.

Assigning Bill Units Automatically

Use the PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT policy opcode to automatically assign bill units to collections agents. By default, this policy opcode is an empty hook.

This policy opcode is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Assigning Bill Units to a Debt Collections Agency

Use the PCM_OP_COLLECTIONS_POL_ASSIGN_DCA policy opcode to automate the logic of selecting a debt collections agent (DCA) when multiple DCAs are configured. By default, this policy opcode is an empty hook.

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

Customizing to Which Collections Group Members to Apply Collections Actions

Use the PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS policy opcode to override the action target setting in the **/config/collections_actions** object. By default, this policy opcode is an empty hook.

This policy opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode.

When you add a collections action to a scenario in Collections Configuration, you specify whether the collections action applies to:

- The individual bill unit
- The parent and all child bill units in a collections sharing group
- All child bill units in a collections sharing group

You can customize this policy opcode to use additional attributes for deciding to which bill units to apply the collections action.

This policy opcode must return in the output flist the **/account** POID and one of the following:

- The PIN_FLD_BILLINFO array with a list of **/billinfo** objects to which to apply the collections action.
- The PIN_FLD_TARGET field set to:
  - **0** to apply the collections action to the **/billinfo** object passed in the input flist.
  - **1** to apply the collections action to the parent and all child **/billinfo** objects in the collections sharing group.
  - **2** to apply the collections action to all child **/billinfo** objects in the collections sharing group.

Mapping Bill Units to Collections Profiles

Use the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode to map a bill unit to a collections profile. By default, this policy opcode maps all bill units to the default collections profile, but you can customize it to map bill units to custom profiles based on specific criteria.

For example, you can assign bill units to profiles based on credit score. You can map bill units with low credit scores to profiles with more aggressive collections scenarios

and bill units with high credit scores to profiles with more lenient collections scenarios.

Your custom code must return the POID of the **/config/collections/profile** object or the collections profile to which the bill units map. BRM creates profile objects when you define one in Collections Configuration.

This policy opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode.

Adding Information that Is Passed to Custom Client Applications

Use the PCM_OP_COLLECTIONS_POL_PUBLISH_EVENT policy opcode to append additional fields to the **/event/audit/collections/action** notification event before it is passed to your custom client application. The entire **/event/audit/collections/action** notification event is passed in the input flist to this policy opcode.

This policy opcode is called by the PCM_OP_COLLECTIONS_PUBLISH_EVENT opcode.

By default, this policy opcode adds the following information to the notification event:

- **/billinfo** POID
- **/collections_action** POID
- Collections action name
- Collections action description
- Collections action type
- Collections action status
- Collections action due date
- Collections action completed date
- Opcode number
- Overdue amount
- Currency
- Total number of days the account has been in collections
- Bill details
- Accounting type

Performing Custom Collections Actions

Use the PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION policy opcode to perform custom collections actions, such as sending SMS text messages to a customer's wireless phone. By default, this policy opcode sets the collections action status to **Pending** and then returns to the calling opcode.

By default, this policy opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode.

The names and descriptions of custom actions are created in Collections Configuration and are stored in **/collections_action** objects. This information is then passed in an input flist to the PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION policy opcode. When you customize this policy opcode, you use this bill unit information to find any other information required for a particular custom action.

Performing Custom Actions when a Bill Unit Leaves Collections

Use the PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO policy opcode to perform custom tasks, such as cleaning up files or modifying a customer's credit score, when a bill unit exits collections. By default, this policy opcode is an empty hook.

This policy opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode when a bill unit exits the collections process.

4

Managing Bill Units in Collections

This chapter provides an overview of managing account bill units in collections by using the Oracle Communications Billing and Revenue Management (BRM) Collections Center application.

For general information about collections, see "Understanding Collections Manager".

For information about installing Collections Center, see "Installing Collections Manager" and "Installing BRM Client Applications on Windows" in *BRM Installation Guide*.

For general information about managing customer bill units, see "About Managing Customers" in *BRM Concepts*.

For detailed instructions about using Collections Center, see the Collections Center Help system.

About Collections Center

Collections Center is an application used by collections personnel to perform day-to-day collections tasks. It supports the work of both collections agents and collections managers.

You assign collections roles (agent and manager) in the Collections Configuration application. All collections personnel must have customer service representative (CSR) accounts. See "Assigning Collections Personnel Roles".

Collections Agent Tasks

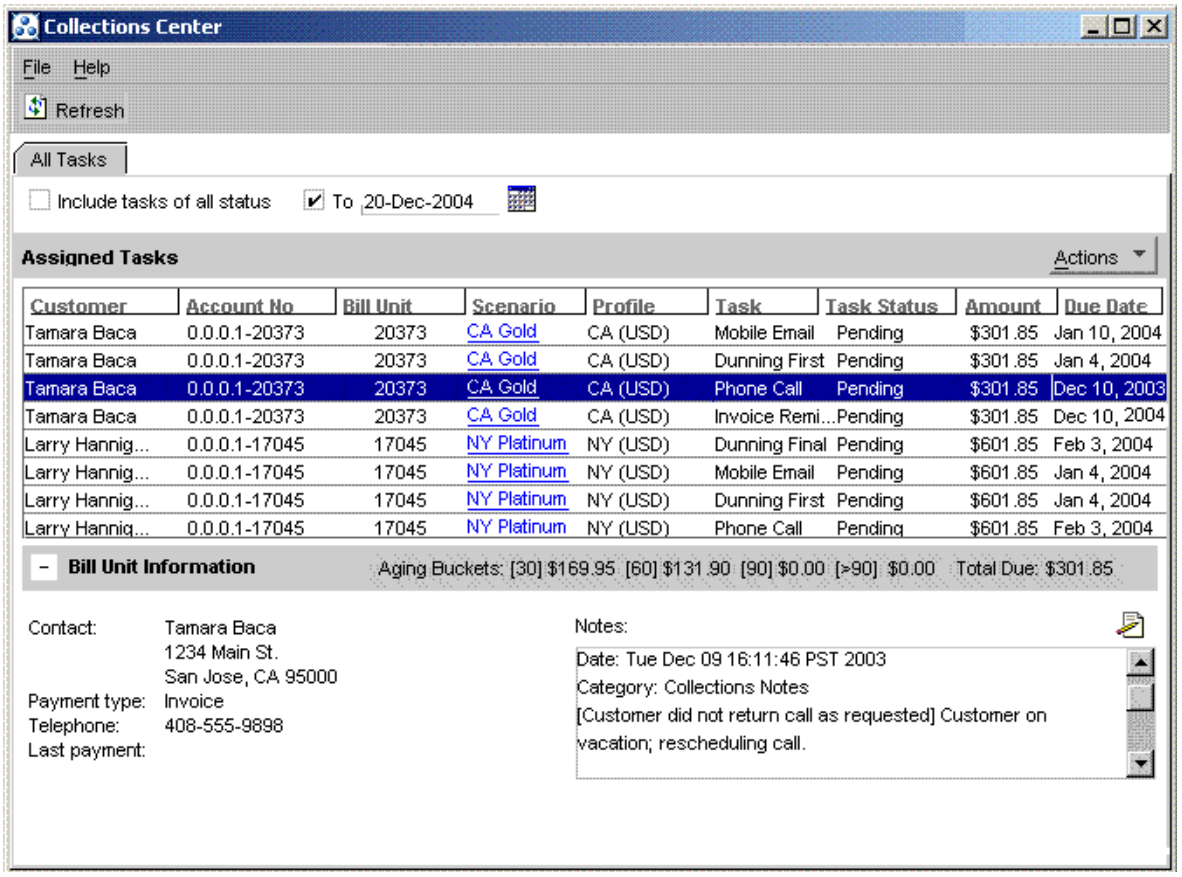
Collections agents monitor and execute collections activities for the bill units assigned to them. When agents log into Collections Center, they see the **All Tasks** tab, which includes a list of tasks (actions) that must be performed that day.

Using Collections Center, collections agents can:

- View collections tasks for bill units assigned to them.
- Monitor the status of both automatic and manual actions defined by the collections scenario.
- Insert new actions for a particular bill unit.
- Reschedule actions.
- Receive credit card payments.

Figure 4–1 shows the **All Tasks** tab for a collections agent:

Figure 4–1 All Tasks Tab



For step-by-step instructions about using the features for collections agents, see Collections Center Help.

Collections Manager Tasks

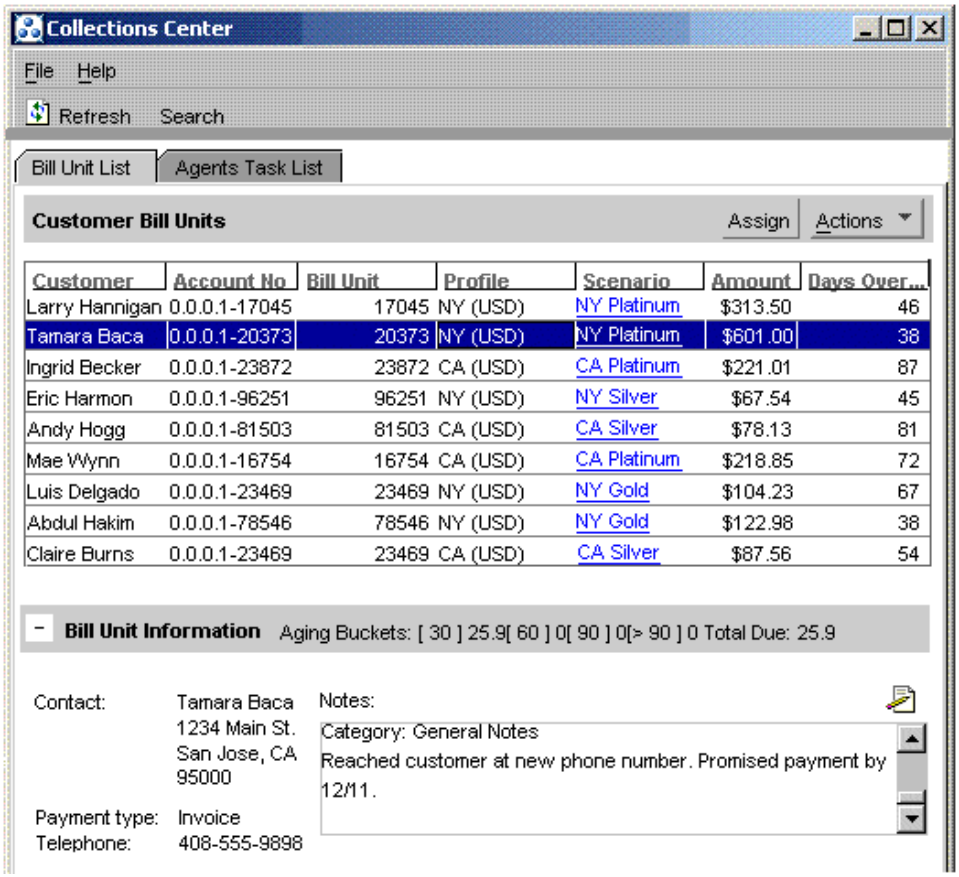
Collections managers supervise the work of collections agents. When they log into Collections Center, they see the **Manager View.**

Using Collections Center, managers can:

- Find specific bill units in collections.
- View bill units.
- View all collections tasks scheduled for a bill unit.
- Assign bill units to collections agents.
- View collections tasks assigned to particular agents.
- Exempt bill units from collections.
- Perform all collections agent tasks.

Figure 4–2 shows the **Bill Unit List** tab that lists bill units based on the specified search criteria:

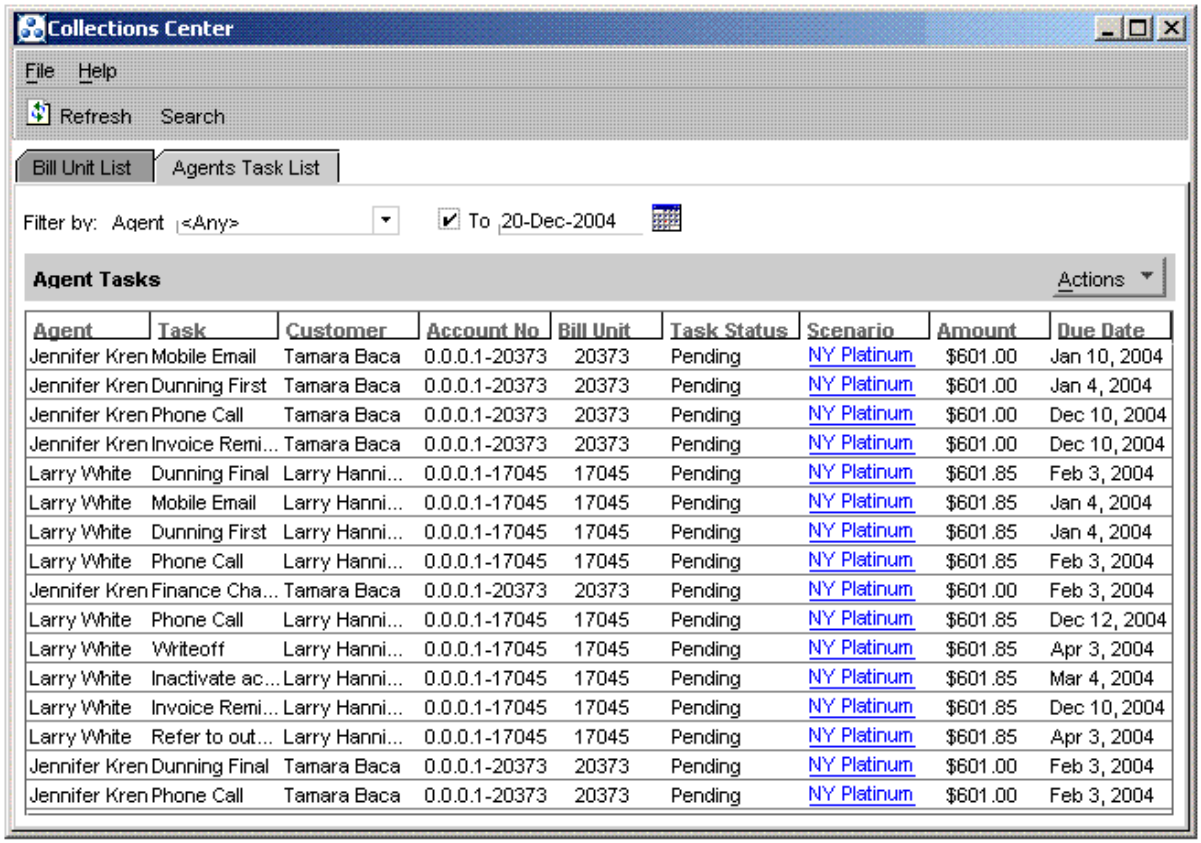
Figure 4–2 Bill Unit List Tab



Collections managers also have access to the **Agents Task List** tab. It is similar to the **All Tasks** tab for collections agents except that it displays tasks for all collections agents that a manager supervises.

Figure 4–3 shows the Agent Task List tab.

Figure 4–3 Agent Task List Tab



For step-by-step instructions about using the Collections Center features for managers, see Collections Center Help.

5

Collections Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Collections Manager utilities.

pin_collections_process

Use this utility to find bill units that are in collections and to trigger collections actions defined by the bill unit's collections scenario.

You must set a minimum value that this utility uses to determine whether bill units should be considered for collections. See "Setting the Minimum Overdue Balance to Process".

This utility is included in the **pin_bill_day** script. It should be run daily before **pin_collections_send_dunning**. See "About Billing Your Customers" in *BRM Configuring and Running Billing* for more information about the billing scripts.

> **Note:** To connect to the BRM database, **pin_collections_process** requires a configuration (**pin.conf**) file in the directory from which you run the utility. A configuration file is installed automatically in *BRM_Home*/**apps/pin_collections** when you install Collections Manager.

Location

BRM_Home/**bin**

Syntax

```
pin_collections_process [-billinfo BillInfoPOID] [-debug] [-verbose] [-report]
[-help]
```

Parameters

-billinfo *BillInfoPOID*
Triggers collections actions against the specified bill unit. Use this parameter to perform collections against a single bill unit that belongs to a collections sharing group.

-debug
Sets the log level to debug and outputs debug information into the log file for this process. If not set, only error-level information is output.

-verbose
Displays progress information.

-report
Shows the summary of this billing run that includes information about how many account bill units are in collections, how many exited collections, and which system actions were performed.

-help
Displays the syntax and parameters for this utility.

Results

The collections status of all bill units with overdue balances is evaluated and, if necessary, updated. For bill units that enter or remain in collections, all actions defined by the relevant collections scenario are performed.

pin_collections_send_dunning

Use this utility to email or print dunning letters. This application is called by **pin_collections_process**, but it can also be run independently.

The delivery method for bill units that use the invoice payment type is specified in the Payment Details panel of Customer Center:

- If the delivery method is set to **Email**, **pin_collections_send_dunning** emails the dunning letter.
- If the delivery method is set to **Postal**, **pin_collections_send_dunning** prints the dunning letter.

For non-invoice bill units, you specify the delivery option in the collections configuration (**pin.conf**) file. See "Setting Dunning Letter Delivery Preferences for Non-Invoice Bill Units".

Important: The Email Data Manager must be running when you run this utility, even for printing. For information, see "Sending Email to Customers Automatically" in *BRM Managing Customers* and "Configuring the Email Data Manager for Printing" in *BRM Designing and Generating Invoices*.

The data used to create the dunning letter is generated by **pin_collections_process** and stored in the database. This data is combined with a dunning letter template to create the final letter.

Note: To connect to the BRM database, **pin_collections_send_dunning** requires a configuration (**pin.conf**) file in the directory from which you run the utility. A configuration file is installed automatically in *BRM_Home*/**apps/pin_collections** when you install Collections Manager.

Location

BRM_Home/**bin**

Syntax

```
pin_collections_send_dunning [-verbose] [-export Path]
```

Parameters

-verbose
Displays progress information.

-export *Path*
Exports the generated dunning letters to the directory you specify.

Results

The dunning letters whose data is collected by **pin_collections_process** are prepared and sent using the delivery method for each bill unit.

pin_load_template

Use this utility to load brand-specific dunning letter templates into the BRM database as **/config/invoice_templates/dunning** objects. These objects contain the XSL templates that you use to format dunning letters and invoice reminders.

For additional information, see "Loading Dunning Letter Templates".

> **Note:** This utility is brand-aware. You can load separate **/config/invoice_templates/dunning** objects for each brand.

> **Important:** To connect to the BRM database, **pin_collections_process** requires a configuration (**pin.conf**) file in the directory from which you run the utility. A configuration file is installed automatically in *BRM_Home*/**apps/pin_collections** when you install Collections Manager.

Location

BRM_Home/**bin**

Syntax

```
pin_load_template [-brand brand_POID] -name template_name [-type template_type]
-format template_output_format -locale locale_name -template template_file
[-usexsl] [-debug] [-logfile log_file]
```

Parameters

-brand *brand_POID*
Specifies the brand account POID for which you are loading the template: **0.0.0.1 /account 104 0** for example.

If your system is not brand enabled, the POID of the root account (**0.0.0.1 /account 1**) must be used.

> **Important:** If the brand account POID contains spaces, use quotation marks.

-name *template_name*
Specifies the template name. The name must be unique for the brand and type. The name is displayed in the Collections Configuration application.

> **Important:** If the name contains spaces, use quotation marks.

-type *template_type*
Specifies the template category. Use **dunning** for dunning letter templates.

-format *template_output_format*
Specifies the output format for the dunning letter templates.

-locale *locale_name*
Specifies the locale name. The default is **en_US**.

-template *template_file*
Specifies the template file name and path.

-usexsl
Specifies the use of XSL to format dunning letters. Must be used when loading an XSL dunning letter template.

-debug
Logs the flist and detailed messages in the log file.

-logfile *log_file*
Specifies the full path and name of the log file.

Results

Loads the specified file into the BRM database as a **/config/invoice_templates/dunning** object.

The utility fails if the specified file or path is invalid.