Oracle® Communications Billing and Revenue Management

Suspending and Recycling Event Records

Release 15.2
G35847-01
January 2026

Oracle Communications Billing and Revenue Management Suspending and Recycling Event Records, Release 15.2

G35847-01

Contents

About This Content

This book describes how to use Oracle Communication Billing and Revenue Management (BRM) in conjunction with Oracle Communications Offline Mediation Controller to correct and recycle event records that fail in Offline Mediation Controller.

Audience

This book is for charging operations personnel and system administrators.

1

About Suspending and Recycling Event Records

Learn how to suspend failed records and recycle them by using Oracle Communications Billing and Revenue Management (BRM) and Oracle Communications Offline Mediation Controller.

Topics in this document:

About Suspending and Recycling Call Detail Records

You can suspend and recycle events to correct processing errors. Processing errors can occur when:

- There is an issue with a record, such as, missing or incorrect fields.
- There is a problem with an incoming CDR file due to a bad policy or configuration.
- There is an issue in your system configuration, such as, it contains the wrong pricing information or the account information is not loaded into the system.

You can suspend and recycle individual records or a batch of records.

Failed records and batches are found during processing in Offline Mediation Controller. They are sent to the BRM database where they are stored as **/suspended_usage** objects or **/suspended_batch** objects. You can then use two methods to process the records and return them to Offline Mediation Controller:

- Use Suspense Manager Center, a GUI application, to analyze and correct records, and then return them to Offline Mediation Controller.
- Use the **pin_recycle** utility to process suspended records and return them to Offline Mediation Controller.

Both components are part of Suspense Manager, an optional BRM manager.

You typically use the **pin_recycle** utility to recycle events after a rating interruption. In this usage, you configure the **pin_recycle** utility to run automatically at specified times to search for and recycle events. See "Setting Up pin_recycle to Run Periodically."

Suspending and Recycling Individual Event Records

The suspense handling workflow for individual event records is as follows:

1. In Offline Mediation Controller a node such as the ECE DC node detects an error.

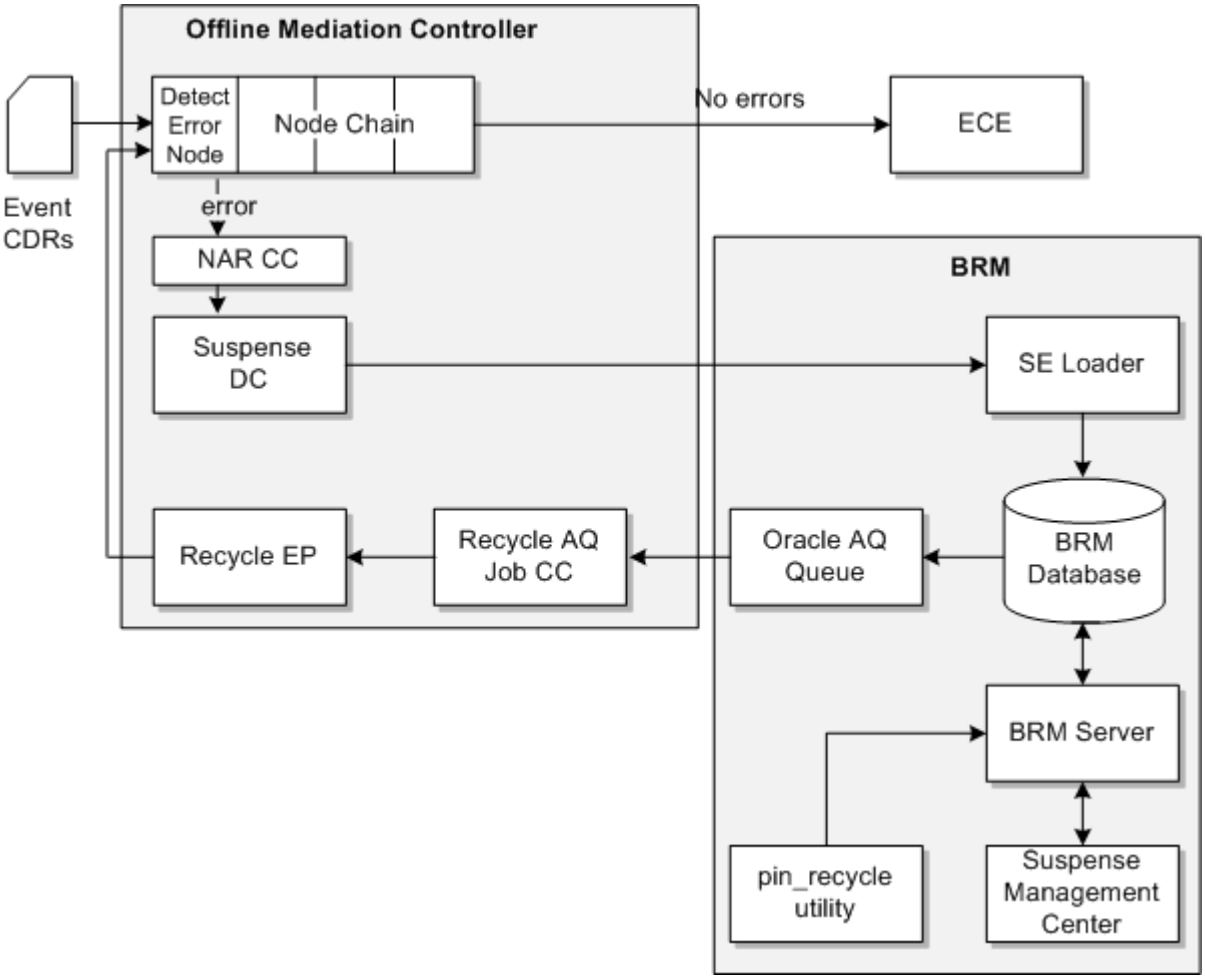
2. If an error is detected, the record is sent to the NAR CC node.
3. The NAR CC node collects failed records and sends them to the Suspense DC node.
4. The Suspense DC node generates **Create** files and **Update** files.
   - Create files contain records that have not been suspended before.
   - Update files contain records that have been suspended before.
5. The Suspense DC node sends the file to the BRM Suspended Event (SE) Loader.
6. The SE Loader loads the suspended events into **/suspended_usage** objects in the BRM database.
7. You use Suspense Manager Center or the **pin_recycle** utility to process the suspended records. In either case, the record is sent to the Oracle AQ queue.

   Instead of recycling the records and batches, you can delete, write off, or archive them.
8. The Recycle AQ Job CC node polls the Oracle AQ and receives the recycled records.
9. The Recycle AQ Job CC node sends the records to the Recycle EP node to continue processing.
10. The recycled record is processed by the node that detected the error.
    - If no more errors are detected, the record is sent to the next node in the node chain. Also, the recycled record is sent to the Suspense DC node, which changes the status to a succeeded state and generates an **Update** file for SE Loader.
    - If more errors are detected, the recycled CDR is sent back to the Suspense DC node, which changes the status to a suspended state and generates an **Update** file for SE Loader.

      The recycled CDR continues through the suspense handling flow until either it is removed (**Written off**) by Suspense Management Center, or it is successful (**Succeeded**).

Figure 1-1 shows the suspense handling flow of suspended and recycled event CDRs.

Figure 1-1 Record Suspense Handling Flow



Suspense Handling Workflow for a Batch of Records

The suspense handling workflow for processing a suspended batch file is as follows:

1. In Offline Mediation Controller, a node that processes records as files detects errors in the file. The node manager counts how many records have errors, and when a threshold percentage value is reached, the entire file is rejected. You set the threshold value in the node manager configuration file.
2. The file-level transaction threshold is configured in the Node Manager configuration file. The file-level transaction threshold determines what percentage of records in a file can be suspended before the entire input file is rejected.
3. If the file is rejected, the suspended batch file is put into a reject directory and information on the batch CDR file's suspension is written to a file.
4. The NAR CC node collects failed records and sends them to the Suspense DC node.
5. The Suspense DC node generates **Create** files and **Update** files.
   - Create files contain records that have not been suspended before.
   - Update files contain records that have been suspended before.

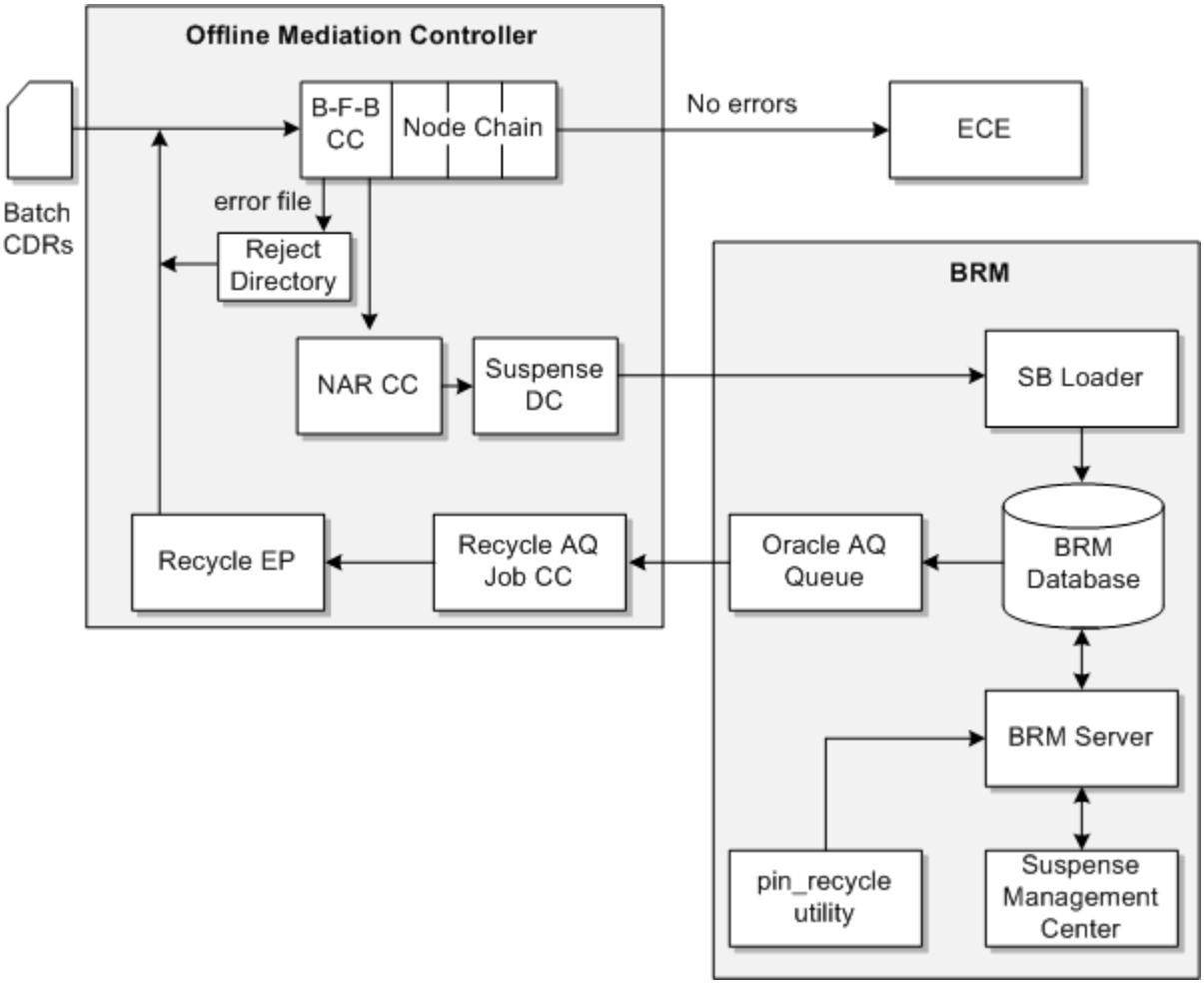
6. The Suspense DC node sends the file to the BRM Suspended Event (SE) Loader.
7. The SE Loader loads the suspended events into **/suspended_usage** objects in the BRM database.
8. You use Suspense Manager Center or the **pin_recycle** utility to process the suspended records. In either case, the record is sent to the Oracle AQ queue.

   Instead of recycling the batch, you can delete it or write it off (you cannot archive only batches).
9. The Recycle AQ Job CC node polls the Oracle AQ and receives the recycled records.
10. The Recycle AQ Job CC node sends the records to the Recycle EP node to continue processing.
11. The Recycle EP node uses the job ID message from the Recycle AQ Job CC node to retrieve the resubmitted batch and returns the suspended batch CDR file back to the input stream of the file-based CC node.
12. The file-based CC node continues processing the suspended batch CDR file:
    - If no more errors are detected, the records from the suspended batch are distributed to the target system, such as, ECE for rating. Also, the resubmitted batch goes back to the Suspense DC node, which changes the status to a succeeded state and generates an **Update** file for SB Loader.
    - If more errors are detected, the resubmitted batch is sent back to the Suspense DC node, which changes the status to a suspended state and generates an **Update** file for SB Loader.

      The resubmitted batch continues through the suspense handling flow until either it is removed (**Written off**) by Suspense Management Center, or it is successful (**Succeeded**).

Figure 1-2 shows the suspense handling flow of suspended and resubmitted batch records.

Figure 1-2 Batch Suspense Handling Flow



About Recycling Suspended Records after Rating Interruptions

Some BRM functionality temporarily interrupts and then restarts rating. These programs and features use **pin_recycle** to recycle records when the interruption is over. These features, such as account migration, temporarily stop rating by suspending calls that come in during the interruption. As these call records arrive, they are appended with a recycle key. When the interruption is over, you use **pin_recycle** to rate all the stored calls that contain that recycle key. You can further configure this feature by using any number of different recycle keys to control when suspended records get recycled.

The **-k** *recycle key* option directs **pin_recycle** to search for all records that contain a specific recycle key string and a status of **suspended**, and queues them for rating. The BRM feature that suspends records determines which records contain the same recycle key and need to be recycled together. This gives **pin_recycle** the flexibility to selectively restrict recycling to just the records with specific characteristics.

For example, the account migration feature moves groups of accounts across databases, and must temporarily stop rating for each group of accounts while they are being moved. Account migration uses internal job IDs to keep track of the accounts being moved, and it also uses these job IDs in the recycle keys for suspended records associated with those same accounts.

About Implementing Suspense and Recycle

To implement suspending and recycling records, you configure functions in Offline Mediation Controller, and on the BRM Server.

The configuration tasks on Offline Mediation Controller are:

- Configure nodes to send failed records and batches to the NAR CC node.
- Create the NAR CC and Suspense DC Node Chain. This node chain receives failed records and batch files and sends them to SE Loader and SB Loader.
- Create the Recycle AQ Job CC and Recycle EP Node Chain. This node chain receives recycled records and batches and sends them back into the original record processing node chain.

For information about configuring Offline Mediation Controller node chains, see the Offline Mediation Controller documentation and the Offline Mediation Controller Administration client Online Help.

The configuration tasks on the BRM server are:

- Configure how suspended records and batches are loaded into the BRM database.
- Configure how recycled records and batches are sent to Offline Mediation Controller
- Enable Suspense Manager to handle custom data.
- Configure debugging and performance options.

See "Configuring Suspense and Recycle on the BRM Server."

2

Installing Suspense Manager

Learn how to install the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager software.

Topics in this document:

Installing Suspense Management Center

To install the Suspense Management Center client software, see "Installing Individual BRM Clients" in *BRM Installation Guide*.

Starting and Using Suspense Management Center

To start and use the Suspense Management Center client software, see "Starting a BRM Client Application on Windows" in *BRM Installation Guide*.

Installing Suspense Manager

> **Note**
>
> If you already installed the product, you must uninstall its features before reinstalling them.

To install Suspense Manager, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Uninstalling Suspense Manager

To uninstall Suspense Manager, see "Uninstalling Optional Components" in *BRM Installation Guide*.

3

Configuring Suspense and Recycle on the BRM Server

Learn how to set up Oracle Communications Billing and Revenue Management (BRM) Suspense Manager.

Topics in this document:

Connecting Suspended Event (SE) Loader to the BRM Database

Suspended Event (SE) Loader moves suspended records from Offline Mediation Controller to **/suspended_usage** objects in the BRM database. To connect to the BRM database, SE Loader uses the **pin_rel** utility. Therefore, you need to add the suspense configuration entries to the **pin_rel** utility **Infranet.properties** file. To do so, run the following commands:

```
cd BRM_home/apps/pin_rel
cat suspense_Infranet.properties Infranet.properties
```

Configuring SE Loader for the pin_recycle Utility

The **pin_recycle** utility requires SE Loader configuration:

1. Add a separate instance of SE Loader to each instance of Offline Mediation Controller.
2. Create a new *BRM_home***/apps/pin_rel/suspense** directory by copying the contents of *BRM_home***/apps/pin_rel/gsm/tel** to *BRM_home***/apps/pin_rel/suspense**. *BRM_home* is the directory where you installed BRM components.
3. Confirm that these files are in the *BRM_home***/apps/pin_rel/suspense** directory:
   - **pin.conf**
   - **SampleRelHandler_config.values**
   - **SampleRelHandler.pl**
4. Add these entries to the *BRM_home***/apps/pin_rel/suspense/SampleRelHandler_config.values** file:

```
$FILETYPE = "*.out.bc";
$HANDLER_DIR = "BRM_home/apps/pin_rel/suspense";#
```

5. Edit the *BRM_home*/**apps/batch_controller/Infranet.properties** file, adding **SUSPENSE** and **RECYCLE_ROLLBACK** entries to **batch.random.events**:

```
batch.random.events     TEL,SMS,FAX,DATA,GPRS,SUSPENSE,RECYCLE_ROLLBACK
```

   Add these parameters to the new entries:

```
#for SUSPENSE events:
SUSPENSE.name                    SUSPENSE Usage
SUSPENSE.handlers                suspHandler
SUSPENSE.file.location           Offline Mediation Controller_output
SUSPENSE.file.pattern            suspense_ *.out

suspHandler.name                 suspHandler
suspHandler.max.at.highload.time 1
suspHandler.max.at.lowload.time  1
suspHandler.start.string           BRM_home/apps/pin_rel/suspense /
SampleRelHandler.pl

#For RECYCLE_ROLLBACK events:
RECYCLE_ROLLBACK.name            RECYCLE_ROLLBACK Usage
RECYCLE_ROLLBACK.handlers        recycleRollbackHandler
RECYCLE_ROLLBACK.file.location   Offline Mediation Controller_output
RECYCLE_ROLLBACK.file.pattern    testDB*.err

recycleRollbackHandler.name      recycleRollbackHandler
recycleRollbackHandler.max.at.highload.time  1
recycleRollbackHandler.max.at.lowload.time   1
recycleRollbackHandler.start.string      BRM_home/apps/pin_rel/recycle/
SampleRelHandler.pl
```

6. Confirm that these *BRM_home*/**apps/pin_rel/Infranet.properties** file entries are set to **false**:

```
infranet.rel.validate_dbnumber = false
infranet.rel.validate_indexes = false
```

> **Note**
>
> The SE Loader architecture makes obsolete the database consistency checks and number validation controlled by these entries.

7. Create a new *BRM_home*/**apps/pin_rel/recycle** directory by copying the contents of *BRM_home*/**apps/pin_rel/gsm/tel** to *BRM_home*/**apps/pin_rel/recycle**.

8. Add these entries to the *BRM_home*/**apps/pin_rel/recycle/SampleRelHandler_config.values** file:

```
$FILETYPE = "*.err.bc";
$HANDLER_DIR = "BRM_home/apps/pin_rel/recycle";#
```

Connecting Suspended Batch (SB) Loader to the BRM Database

Suspended Batch (SB) Loader moves suspended batch files from Offline Mediation Controller to **/suspended_batch** objects in the BRM database. To connect to the BRM database, SB Loader uses the **load_suspended_batch_info.pl** script This script is usually set up to run automatically, but can also be run manually as needed.

Configuring Event Notification for Suspense Manager

When suspended event records are recycled or written off, Suspense Manager uses event notification to call opcodes that perform the appropriate follow-up operations.

Although any subclass of the **/event** storable class can be used to trigger event notification (see "About Notification Events" in *BRM Developer's Guide*), Suspense Manager generates the following non-persistent events specifically to use for event notification:

- **/event/notification/suspense/recycle**: By default, when this event occurs, the EAI framework publishing opcode is called.
- **/event/notification/suspense/writeoff**: By default, when this event occurs, PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY is called.
- **/event/notification/suspense/delete**: By default, when this event occurs, *no* opcode is called. To enable this event to trigger an opcode call, see "Editing The Event Notification List" in *BRM Developer's Guide*.
- **/event/notification/suspense/edit**: By default, when this event occurs, *no* opcode is called. To enable this event to trigger an opcode call, see "Editing The Event Notification List" in *BRM Developer's Guide*.

Before you can use Suspense Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes entries for these events:
   - (For Revenue Assurance Manager only) From *BRM_home***/sys/data/config/pin_notify_ra**:

     /event/notification/suspense/writeoff
   - From *BRM_home***/sys/data/config/pin_notify_ifw_sync**:

     /event/notification/suspense/recycle
3. (Optional) Add entries for these events to your final event notification list:
   - **/event/notification/suspense/edit**
   - **/event/notification/suspense/delete**

   > **Note**
   >
   > These events are *not* in a default event notification configuration file. You must manually add them to your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
5. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.

6. Load your final event notification list into the BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Configuring the Oracle AQ Queue to Send Data to Offline Mediation Controller

Suspense Manager Center and the **pin_recycle** utility both use the Oracle AQ queue to send recycled records and batches to the Offline Mediation Controller Recycle AQ Job CC node.

The BRM installer automatically creates a default queue in a specified database schema. To configure the queue to send data to Offline Mediation Controller:

- If your system requires multiple queues, create additional queues by manually running the **pin_publish_aq_oracle.pl** utility.
- The default BRM database number for your Oracle DM is 0.0.0.1. If you change your database number, you must change the value of the DB attribute in the Oracle DM publisher definition.
- You configure which events the Oracle DM sends to each database queue by editing the **aq_event_map** file. This file must specify the names of all the queues in your system and which events to send to each queue.
- You must modify the CM **pin.conf** file to enable the EAI framework to notify the Oracle DM when specific events occur. You should also verify that the pointer to the Oracle DM specifies the correct database and port numbers.

> **Important**
>
> If you use a multischema system and set up more than one CM, you must edit the configuration file for each CM.

For more information, see "Configuring Your AQ Database Queues" in *BRM System Administrator's Guide*.

Configuring the Payload Definition File for Suspense Manager

If you use a different Synchronization Queue DM database number, you must change the value of the publisher database in the payload configuration file.

1. In a text editor, open the payload configuration file in the *BRM_home*/**sys/eai_js** directory (**payloadconfig_crm_sync.xml**, or the merged file if you merged payload configuration files), where *BRM_home* is the directory in which BRM is installed.
2. Search for the following entry:

   ```
   RecycleRequest
   ```

3. Add or modify the entries for suspended events.

   For example:

   ```
   <!-- For Suspended Events -->
   <RecycleRequest Source="EVENT"
   Tag="RecycleRequest"
   ```

```
StartEvent="/event/notification/suspense/recycle" >
<Attribute Tag="Version" Value="1.0" />
<Field PinFld="PIN_FLD_ACCOUNT_OBJ" Tag="AccountObj"/>
<SubElement Name="JobActions"
OnEvent="/event/notification/suspense/recycle" />
</RecycleRequest>
```

4. Search for the following entry:

   ```
   ResubmitBatchRequest
   ```

5. Add or modify the entries for suspended batch.

   For example:

   ```
   <!-- For Suspended Batch -->
   <ResubmitBatchRequest Source="EVENT"
   Tag="ResubmitBatchRequest"
   StartEvent="/event/notification/suspense/batch_resubmit" >
   <Attribute Tag="Version" Value="1.0" />
   <Field PinFld="PIN_FLD_ACCOUNT_OBJ" Tag="AccountObj"/>
   <SubElement Name="JobActions"
   OnEvent="/event/notification/suspense/batch_resubmit" />
   </ResubmitBatchRequest>
   ```

6. Add or modify the entries for the event sub-element.

   For example:

   ```
   <!-- Action Obj -->
   <JobActions Source="EVENT" PinFld="PIN_FLD_ACTIONS"
   DataFrom="PIN_FLD_ACTIONS" Tag="Actions" >
   <Field PinFld="PIN_FLD_ACTION_OBJ" Tag="ActionObj" />
   </JobActions>
   ```

7. Save and close the file.

Integrating Event Record Field Mapping

To process records in Offline Mediation Controller, you need to map record field names to numbers. To do so, you edit the **edr_field_mapping.xml** file and load it into the BRM database.

To define and load field mappings:

1. Open the **edr_field_mapping.xml** file in the *BRM_home***/sys/data/config/** directory.
2. Search for the following line:

   ```
   <edr_field_mapping name="Name">
   ```

   where *Name* is the version number of the mapping.
3. Add or update an **id** field entry, using the following syntax:
   - The **id** field must contain numbers and periods.

     The parent block field container must end with a **.b** value. The child **id** field must be sequentially numbered and prefixed with the parent **id** block field's value, without the **.b** value.
   - A **type** field, which contains the numbers **1, 2, 4, 8**, or **32**.

     where:

     – **1** is used for **string** data types.

- **2** is used for **integer** data types.
- **4** is used for **date** data types.
- **8** is used for **decimal** data types.
- **32** is used for **block** data types.

For example:

```
<f id="1.5.b" name="DETAIL.ASS_DUMMY_EXT" type="32"/>
<f id="1.5.0.1" name="DETAIL.ASS_DUMMY_EXT.RECORD_TYPE" type="1"/>
<f id="1.5.1.2" name=DETAIL.ASS_DUMMY_EXT.RECORD_NUMBER" type="2"/>
```

> **Note**
>
> Before new entries are loaded in the database, all existing entries that contain the same version number as the new entries are deleted. To prevent overwriting of existing entries when the **edr_field_mapping.xml** file is loaded into the database, configure the **name** attribute to increment the version number to the next number.

4. Save and close the file.
5. Go to the *BRM_home***/sys/data/config/** directory.
6. Run the following command, which loads the event record field mapping file into the BRM database:

   ```
   load_edr_field_mapping XML_file
   ```

   where *XML_file* is the name of the XML file that contains the configuration data.

See "load_edr_field_mapping" for more information.

Configuring Debugging

Suspense Management Center provides the following optional ways for capturing and displaying debugging information:

- The **SuspenseManagementCenter_opcodes.log** log file captures all opcode input and output flists used by Suspense Management Center.
- The **javapcm.log** file contains detailed debugging information. By default, the logging level is set to **0**, the lowest level. The highest level is **3**. You must set the error buffer to **true** to enable **javapcm** logging.
- The **Dloglevel** entry creates a console window for the Suspense Management Center that displays error messages and debugging information.

1. Set up Java PCM Logging by adding these entries to the **Infranet.properties** file (in the *BRM_home***/Program Files/Portal Software/SuspenseManagementCenter/lib** directory):

   ```
   infranet.log.level=3
   infranet.log.logallebuf=true
   infranet.log.opcodes.enabled=true
   infranet.log.opcodes.file=SuspenseManagementCenter_opcodes.log
   ```

2. Set up Suspense Management Center console logging by opening the **RunSM.bat** file (in the *BRM_home***/Program Files/Portal Software/SuspenseManagementCenter/lib** directory), and changing the **javaw** entry to **java,** and add this parameter:

```
java -Dloglevel="ALL".
```

Logging information is now:

- Displayed in the Suspense Management Center console window.
- Available in these files:
  - *BRM_home*/**Program Files/Portal Software/SuspenseManagementCenter/lib/javapcm.log**
  - *BRM_home*/**Program Files/Portal Software/SuspenseManagementCenter/lib/SuspenseManagementCenter_opcodes.log**

Setting Up pin_recycle to Run Periodically

Some BRM functionality temporarily interrupts and then restarts rating. These programs and features use **pin_recycle** to recycle records when the interruption is over. These features, such as account migration, temporarily stop rating by suspending calls that come in during the interruption. As these call records arrive, they are appended with a recycle key. When the interruption is over, you use **pin_recycle** to rate all the stored calls that contain that recycle key. You can further configure this feature by using any number of different recycle keys to control when suspended records get recycled.

The **-k** *recycle key* option directs **pin_recycle** to search for all records that contain a specific recycle key string and a status of **suspended**, and queues them for rating. The BRM feature that suspends records determines which records contain the same recycle key and need to be recycled together. This gives **pin_recycle** the flexibility to selectively restrict recycling to just the records with specific characteristics.

For example, the account migration feature moves groups of accounts across databases, and must temporarily stop rating for each group of accounts while they are being moved. Account migration uses internal job IDs to keep track of the accounts being moved, and it also uses these job IDs in the recycle keys for suspended records associated with those same accounts.

You need to run **pin_recycle** periodically both to queue the temporarily stored records for rating, and to delete them. The **cron** command is the typical way to do this, although you can run **pin_recycle** like any other BRM command-line utility. This section explains how to set up **cron** command to run **pin_recycle**.

You need to add two **pin_recycle** entries to the **cron** command. One to search for and recycle records, and the other to delete them after they are recycled. See "pin_recycle" for the syntax.

To run **pin_recycle** periodically, add entries like the following. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and queues records with a recycle key of **Trigger_Billing** for rating:

```
0 1 * * * BRM_home/bin/pin_recycle -k Trigger_Billing &
```

To remove records from the BRM database, add an entry like the following. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and deletes records with a recycle key of **Trigger_Billing**:

```
0 1 * * * BRM_home/bin/pin_recycle -k Trigger_Billing -d &
```

4

Changing the List of Suspense Reasons and Subreasons

Learn how to change the list of reasons for suspending events in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Changing the List of Suspense Reasons and Subreasons

Suspense Manager adds the reasons for failures to the records it stores. These reasons, called *suspense reasons,* can be divided into more specific *suspense subreasons*. These suspense reasons and subreasons are stored in the record. Because they are stored in the call records, you can search for them by using Suspense Management Center. For example, you can search for all the calls that could not be associated with a subscriber.

> **Note**
>
> You cannot define subreasons for suspended batches, only for suspended records.

The Offline Mediation Controller error messages that actually cause call failures are mapped to these suspense reasons and subreasons. A default error code-to-suspense reason mapping is provided in Suspense Manager. If your business requires different suspense reasons or subreasons, you can change them and their mappings. You can make these changes at any time, but because you may have to upgrade existing data, it is best to have this mapping in place before you go into production.

Table 4-1 shows the default mapping. If the default mappings do not meet your business needs, follow the steps below to change them. You first edit the text file with your new suspense reasons and subreasons, and then load the mapping into the database.

1. Edit the *BRM_home***/sys/msgs/suspense_reason_code.en_US** file, adding suspense reasons as strings and mapping them to integers.

   Sample entry for a suspense reason with the ID of 1:

   ```
   STR
         ID = 1 ;
         VERSION = 1 ;
         STRING - "Unable to identify customer" ;
   END
   ```

   Sample entry for a suspense subreason for a suspense reason with the ID of 2:

   ```
   DOMAIN = "suspense_subreason_1" ;
   STR
         ID = 2;
         VERSION = 1 ;
   ```

```
      STRING = "B number missing" ;
END
```

> **Note**
>
> - The default string has an ID of 0. This string appears by default in the case of an error that is not mapped to a suspense reason.
> - The reason code numbers 65535 and 65534 are reserved for use by BRM.

The format of the **suspense_reason_code.***locale* file is similar to that of the **reasons.locale** file.

2. Map your suspense reasons to Offline Mediation Controller error messages by editing the **pin_suspense_reason_code** (for records) or **pin_batch_suspense_reason_code** (for batch files) file in the *BRM_home***/sys/data/config** directory.
3. Load your localized strings into the database by using the **load_localized_strings** utility.

   Example syntax:

   ```
   load_localized_strings suspense_reason_code.en_US
   ```

4. Load your suspense reason code mapping into the database by using the **load_pin_suspense_reason_code** or **load_pin_batch_suspense_reason_code** utility (in the *BRM_home***/bin** directory). For details, see "load_pin_suspense_reason_code" or "load_pin_batch_suspense_reason_code".

   Example syntax for records:

   ```
   load_pin_suspense_reason_code pin_suspense_reason_code
   ```

   Example syntax for batch files:

   ```
   load_pin_batch_suspense_reason_code pin_batch_suspense_reason_code
   ```

5. Verify that the strings were loaded by displaying the **/strings** objects using the Object Browser or the **robj** command with the **testnap** utility. See *BRM Developer's Guide* for general instructions on using **testnap**.
6. Stop and restart the Connection Manager (CM).
7. Stop and restart Suspense Management Center.

Your suspense reason and subreason strings are now loaded into the BRM database to be displayed and used by Suspense Management Center.

Table 4-1 shows the default mapping between Offline Mediation Controller error messages and Suspense Manager reasons for suspending records.

Table 4-1 Suspense Reasons

Offline Mediation Controller Error Messages	Suspense Reason String	Suspense Subreason String
CANCEL_NOT_ALLOWED	ECE DC error	Request processing cannot be canceled.
CREDIT_CEILING_BREACH	ECE DC error	Request processing resulted in ceiling floor breached of an involved BalanceItem.

Table 4-1 (Cont.) Suspense Reasons

Offline Mediation Controller Error Messages	Suspense Reason String	Suspense Subreason String
CREDIT_FLOOR_BREACH	ECE DC error	Request processing resulted in credit floor breached of an involved BalanceItem.
CUSTOM_EXTENSION_ERROR	ECE DC error	Error has occurred while executing operator specific custom extension code.
CUSTOMER_IN_TRANSACTION	ECE DC error	Customer is in another transaction.
DUPLICATE_REQUEST	ECE DC error	It is a duplicate request.
FINAL_UNIT_INDICATOR	ECE DC error	Request processing raised a final unit indicator flag.
INCORRECT_IMPACTS_FOR_DEBIT	ECE DC error	The number of debit impacts does not match with the actual debit impacts.
INSUFFICIENT_RATED_QUANTITY	ECE DC error	Rated quantity was less than the minimum quantity for reservation.
LIFECYCLE_VALIDATION_FAILED	ECE DC error	After life cycle validation call was disallowed.
MISSING_SYSTEM_ALTERATION_FOR_DEBIT	ECE DC error	There are no system alteration offers for the debit request.
NO_QUALIFIED_CHARGE_OFFERS	ECE DC error	There are no qualified charge offers for this request.
NO_RATED_QUANTITY	ECE DC error	No quantity rated.
REFUND_NOT_ALLOWED	ECE DC error	Refund request is not allowed.
SYSTEM_ERR	ECE DC error	Undefined system error.
TX_FAILED	ECE DC error	Transaction commit failed.
ZERO_RUM_QUANTITY	ECE DC error	The RUM is zero.

5

Customizing Suspense and Recycle

Learn how to customize suspending and recycling events in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Overriding Suspense Handling Rules

During recycling, Suspense Management Center lets you process failed call records and batches.

This override feature allows you to capture and temporarily hold suspicious calls in a suspended state until you can inspect them. If they pass inspection, you can override your validation rules and recycle the calls to capture the revenue they represent. The reasons for suspended CDR file are separate from those or individual CDRs and must be handled separately.

You select the override reasons from the Suspense Management Center Recycle screen. The suspense reasons are then overridden for all of the calls in that recycle CDR files. This directs Suspense Manager to successfully process the individual CDRs or CDR files, even though they do not pass your validation rules.

Changing the List of Override Reasons

The list of override reasons offered to Customer Service Representatives (CSRs) during recycling is configurable. You can change the list at any time by editing the *BRM_home***/sys/data/config/pin_suspense_override_reason** file, and then loading it into your database by using the **load_pin_suspense_override_reason** utility in the *BRM_home***/bin** directory. *BRM_home* is the directory where you installed BRM components.

For example:

```
load_pin_suspense_override_reason pin_suspense_override_reason
```

For details on this utility, see "load_pin_batch_suspense_override_reason".

Changing the List of CDR File Override Reasons

The list of CDR file override reasons offered to Customer Service Representatives (CSRs) while resubmitting them is configurable and is separate from the override reasons for individual call records. You can change the list at any time by editing the *BRM_home***/sys/data/config/pin_batch_suspense_override_reason** file, and then loading it into your database by using the **load_pin_batch_suspense_override_reason** utility in the *BRM_home***/bin** directory.

For example:

```
load_pin_batch_suspense_override_reason pin_batch_suspense_override_reason
```

For details on this utility, see "load_pin_batch_suspense_override_reason".

Using Custom Data with Suspense Manager

When you use Suspense Management Center, you can search for records and edit records based on data in the Suspense Manager storable classes. You can extend those classes to add custom fields that you can display, search on, and edit in Suspense Management Center.

- The **/suspended_usage** class contains the data on which you can search for records, and edit records. For example, you can search for records that have a specific APN number.
- The **/suspended_batch** class contains the data on which you can search for batches. You cannot edit batches.

In addition to enabling custom fields to be searchable and/or editable, you can enable Suspense Management Center to display the custom fields.

> **Important**
>
> Before adding custom event types to the **pin_rel Infranet.properties** file, append the **suspense_Infranet.properties** file to the **pin_rel** utility **Infranet.properties** file. See "Connecting Suspended Event (SE) Loader to the BRM Database".

To use custom data in Suspense Manager Center:

1. Extend the Suspense Management storable classes:
   - To add fields for searching on and editing records, extend the **/suspended_usage** subclasses.
   - To add searchable fields for batches, extend the **/suspended_batch/cdr** subclass.

   To extend storable classes, use the Storable Class Editor. See *BRM Developer's Guide*. For a list of fields in the suspense classes, see *BRM Developer's Reference*.
2. Edit the *BRM_home***/apps/pin_rel/Infranet.properties** file to add new event types for each of your **/suspended_usage** or **/suspended_batch** subclasses and for temporary objects. Use the **/suspended_usage/telco** section of *BRM_home***/apps/pin_rel/Infranet.properties** as a guide.
3. To display custom fields in Suspense Management Center, add them to the **custom.properties** file in the **Program Files/Portal Software/SuspenseManagementCenter/lib**.

   This example defines a new field called **Record Type:** to display in Suspense Management Center:

   ```
   #1. Specify the display name:
   #
   #field.<dd_field_name>.name = <display name>

   field.PIN_FLD_RECORD_TYPE.name = Record Type:
   ```
4. To display custom fields in the Web version of Suspense Management Center, add them to the **SuspenseManager_en.jnlp** file (in the *Web_Start_home* directory), adding any custom fields from your **/suspended_usage** subclasses.

This example defines a new field called **Record Type:** to display in Suspense Management Center:

```
#1. Specify the display name:
#
#field.<dd_field_name>.name = <display name>

field.PIN_FLD_RECORD_TYPE.name = Record Type:
```

5. When you configure suspense and recycle on Offline Mediation Controller, you need to configure the Suspense DC node rule file and add the BRM table names for the searchable fields. See *Offline Mediation Controller User's Guide* and the Offline Mediation Controller Online Help.

Specifying Editable Fields in Suspense Manager Center

The fields that you can edit in Suspense Management Center are called *editable fields*. They are defined in the **pin_suspense_editable_flds** file.

To add or remove editable fields, edit and load the **pin_suspense_editable_flds** file. To improve performance, you should limit the number of editable fields to only those that are necessary.

These fields can be standard fields, or custom fields. such as custom /event fields. If they are custom fields, you also need to add them to the **/suspended_usage** subclasses to enable them to be part of the suspense and recycle process. For a list of fields in the **/suspended_usage** subclasses, see *BRM Developer's Reference*.

> **Important**
>
> If you need to edit custom fields, extend the **/suspended_usage** subclasses before defining editable fields.

To load your list of editable fields into the database for use by Suspense Management Center:

1. Edit the *BRM_home***/sys/data/config/pin_suspense_editable_flds** file.
2. Run the **load_pin_suspense_editable_flds** utility (located in *BRM_home***/bin)** to load the editable fields into the database:

   ```
   load_pin_suspense_editable_flds pin_suspense_editable_flds
   ```

   For details, see "load_pin_suspense_editable_flds".

6
Increasing Performance

Learn how to increase the performance for suspending and recycling events in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Increasing Heap Size to Avoid Performance Problems

If the searches you run in Suspense Management Center return particularly large results, your performance may slow noticeably, or you may get "Out of memory" error messages. The solution is to increase your maximum heap size. The exact amount varies greatly with your needs and system resources. If performance is very bad or you get "Out of memory" messages frequently, start by doubling the maximum heap size to 128 MB. Remember, however, that making the heap size too large will degrade the performance of other processes.

There are two ways to increase the maximum heap size, depending on whether you have standalone or WebStart BRM implementations.

Increasing Heap Size for Standalone Implementations

To increase the heap size for standalone implementations:

1. Edit the *BRM_home***/lib/runSMC.bat** file to increase the heap size (memory allocation pool) to solve "Out of memory" messages.

   By default, Suspense Management Center has a maximum heap size of 64 MB. This variable is controlled by the **-Xmx** *size* entry in the Suspense Manager Center startup line in **runSMC.bat**. No **-Xmx** *size* entry is present in the startup line by default. To increase the heap size, add this entry and a number (in megabytes) to the Suspense Management Center startup line.

   This example adds a 128 MB maximum heap size to Suspense Management Center:

   ```
   @start C:\PROGRA~1\COMMON~1\PORTAL~1\JRE\bin\javaw.exe -Xmx128m -cp
   ".;%SMCDIR%;%SMCDIR%\lib;%SMCDIR%\lib\suspensemgtmain.jar;%SMCDIR%
   \lib\pfc.jar;%SMCDIR%\3plibs\jh.jar;%SMCDIR%\lib\pcmext.jar;%SMCDIR%
   \lib\pcm.jar;%SMCDIR%\lib\Suspense_Management_Help_en.jar;%SMCDIR%
   \lib\Application_Center_Help_en.jar;" com.portal.appcenter.AppCenterMain
   suspensemgtsuite
   ```

   > **Note**
   >
   > Be sure to precede and follow the **-Xmx** *size* entry with a space.

2. Stop and restart Suspense Management Center to make the change take effect.

Increasing Heap Size for Web Start Implementations

To increase the heap size for Web Start implementations:

1. Open your **SuspenseManagement**_*locale*.**jnlp** file.
2. Change the **j2se** element to include a max-heap-size attribute.

   The default entry looks like this:

   ```
   <j2se version="1.4*"/>
   ```

   For example, this entry changes the maximum heap size to 128 MB:

   ```
   <j2se version="1.4*" max-heap-size="128m"/>
   ```

   > **Note**
   >
   > The max heap size specified in the JNLP file is used for all associated Suspense Management Center clients.

3. Stop and restart Suspense Management Center to make the change take effect.

Creating Indexes for Search Templates

By default, Suspense Manager does not include any database indexes for searches other than indexes based on POID IDs. You can improve database performance by creating indexes for your most common searches. The example below guides you through the process.

> **Note**
>
> If there are many indexes on the tables for **/suspended_usage** objects, you run the risk of degrading SE Loader performance during bulk loading of **/suspended_usage** objects. Experiment to find the right balance of indexes for your system.

Example search template:

```
#Suspense Management Template
#Fri Nov 14 09:16:53 PST 2003
PIN_FLD_CALL_DURATION.max=
PIN_FLD_SUSPENSE_REASON.value=<All>
PIN_FLD_CALL_DURATION.selected=false
PIN_FLD_EDITED.value=<All suspended>
PIN_FLD_TEST_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_RECORD_TYPE.selected=true
PIN_FLD_FILENAME.selected=true
PIN_FLD_TEST_ERROR_CODE.min=
PIN_FLD_STATUS.value=Suspended
PIN_FLD_RECORD_TYPE.text=
PIN_FLD_SUSPENSE_SUBREASON.selected=false
PIN_FLD_SERVICE_CODE.text=
PIN_FLD_NUM_RECYCLES.max=0
PIN_FLD_SUSPENSE_REASON.selected=true
PIN_FLD_TEST_SUSPENSE_REASON.value=<All>
PIN_FLD_START_TIME.selected=false
```

```
PIN_FLD_CALLING_FROM.text=
PIN_FLD_CALL_DURATION.min=
PIN_FLD_NUM_RECYCLES.selected=true
PIN_FLD_FILENAME.text=
PIN_FLD_EDITED.enabled=true
PIN_FLD_CALLED_TO.selected=true
PIN_FLD_STATUS.selected=false
PIN_FLD_TEST_SUSPENSE_REASON.selected=false
PIN_FLD_RECYCLE_T.selected=false
PIN_FLD_TEST_ERROR_CODE.selected=false
PIN_FLD_CALLING_FROM.selected=true
PIN_FLD_CALLED_TO.text=
PIN_FLD_TEST_ERROR_CODE.max=
PIN_FLD_BATCH_ID.selected=false
PIN_FLD_BATCH_ID.text=
PIN_FLD_SERVICE_CODE.selected=false
PIN_FLD_EDITED.selected=false
PIN_FLD_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_NUM_RECYCLES.min=0
```

The example search template translates into this SQL statement:

```
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
```

For Oracle databases, use the statements below to determine which indexes would improve performance.

To evaluate this SQL statement, turn on autotrace and run this statement.

This is the output:

```
SQL> set autotrace on;
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
...
13 rows selected.

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (FULL) OF 'SUSP_USAGE_TELCO_INFO_T'
   3    1    TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   4    3    INDEX (UNIQUE SCAN) OF 'I_SUSPENDED_USAGE__ID' (UNIQUE)

Statistics
----------------------------------------------------------
         176  recursive calls
           0  db block gets
          38  consistent gets
           0  physical reads
           0  redo size
        1218  bytes sent via SQL*Net to client
         430  bytes received via SQL*Net from client
           2  SQL*Net roundtrips to/from client
           4  sorts (memory)
           0  sorts (disk)
          13  rows processed
```

The Execution Plan shows a listing of TABLE ACCESS (FULL), indicating that search performance would be better if you had created the indexes. Based on the select statement, add appropriate indexes. In this example add them to both **num_recycles** and the status in **suspended_usage_t**. This sample statement creates those indexes:

```
SQL> create index i_susp_usage_test on suspended_usage_t (status,
SQL> num_recycles);
```

After creating the indexes, rerunning the select statement results in a more efficient Execution Plan:

```
Execution Plan
----------------------------------------------------------
   0        SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   3    2       INDEX (RANGE SCAN) OF 'I_SUSP_USAGE_TEST' (NONUNIQUE)
   4    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSP_USAGE_TELCO_INFO_T'
   5    4       INDEX (UNIQUE SCAN) OF 'I_SUSP_USAGE_TELCO__ID' (UNIQUE)

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          19  consistent gets
          0  physical reads
          0  redo size
          1218  bytes sent via SQL*Net to client
          430  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          13  rows processed
```

The table scan does not read FULL this time, and there are no **recursive calls** and fewer **consistent gets**. The result is a more efficient call.

Configuring the Number of Suspended Records to Process in a Transaction

To avoid a large database transaction during bulk operations, you can specify the number of records to process in each transaction in a bulk operation:

1. Edit the **pin_suspense_params** file in the *BRM_home*/**sys/data/config** directory to specify the maximum number of records to process in a transaction. The file includes examples and instructions.
2. Load the contents of the file into the **/config/suspense_params** object in the BRM database by using **load_pin_suspense_params**.

   See "load_pin_suspense_params".

Suspense Management Center and the **pin_recycle** utility read the **/config/suspense_params** file to get the number of records to process in each opcode call and determine the number of times to call the opcodes.

7

Suspense Management Utilities

This document provides reference information for Oracle Communications Billing and Revenue Management (BRM) Suspense Management utilities.

Topics in this document:

load_edr_field_mapping

Use this utility to load event record field mapping into the **/edr_field_mapping** object in the BRM database. See "Integrating Event Record Field Mapping".

Location

*BRM_home***/bin**

Syntax

```
load_edr_field_mapping [-d] [-v] [-t] [-h] XML_file
```

Parameters

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

-v
Displays detailed information as the utility runs.

-t
Validates the XML file. Your event record field mapping XML file must conform to the XML schema rules in the *BRM_home***/xsd/edr_field_mapping.xsd** file.

-h
Shows the help on the utility.

XML_file
The name and location of the event record field mapping configuration file, which maps the event record field name to an ID number. The default **edr_field_mapping.xml** file is in *BRM_home***/sys/data/config** directory.
If you copy the **edr_field_mapping.xml** file to the same directory from which you run the **load_edr_field_mapping** utility, you do not have to specify either the path or the file name. If you run the command in a different directory from where the **edr_field_mapping.xml** file is located, you must include the entire path for the file.

load_pin_suspense_editable_flds

Use this utility to load editable fields into the **/config/suspense_editable_flds** object in the BRM database. You define editable fields in the **pin_suspense_editable_flds** file in *BRM_home***/sys/data/config**.

For more information, see "Specifying Editable Fields in Suspense Manager Center".

> ⚠ **Caution**
>
> The **load_pin_suspense_editable_flds** utility overwrites existing **/config/suspense_editable_flds** objects. If you are updating editable fields, you cannot load new editable fields only. You must load complete sets of editable fields each time you run the utility.

> ⓘ **Note**
>
> To connect to the BRM database, the **load_pin_suspense_editable_flds** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_editable_flds [-d] [-v] [pin_suspense_editable_flds_file]
```

Parameters

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

-v
Displays detailed information as the utility runs.

pin_suspense_editable_flds_file
The name and location of the file that defines the list of editable fields used by Suspense Management Center. The default **pin_suspense_editable_flds** file is in *BRM_home***/sys/data/config**.

If you copy the **pin_suspense_editable_flds** file to the same directory from which you run the **load_pin_suspense_editable_flds** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_editable_flds** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_editable_flds** utility notifies you when it successfully creates the **/config/suspense_editable_flds** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_editable_flds** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

> **Note**
>
> You must restart Suspense Management Center to make new editable fields available.

load_pin_suspense_override_reason

Use this utility to load override reasons into the **/config/suspense_override_codes** object in the BRM database. You define override reasons in the **pin_suspense_override_reason** file in *BRM_home***/sys/data/config**. See "Overriding Suspense Handling Rules".

> **Caution**
>
> The **load_pin_suspense_override_reason** utility overwrites existing suspense override reasons. You must load complete sets of override reasons each time you run the utility.

> **Note**
>
> To connect to the BRM database, the **load_pin_suspense_override_reason** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_override_reason [-d] [-v] pin_suspense_override_reason_file
```

Parameters

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

-v
Displays detailed information as the utility runs.

pin_suspense_override_reason_file
The name and location of the file that defines the override reasons. The default **pin_suspense_override_reason** file is in *BRM_home***/sys/data/config**.
If you copy the **pin_suspense_override_reason** file to the same directory from which you run the **load_pin_suspense_override_reason** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_override_reason** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_override_reason** utility notifies you when it successfully creates the **/config/suspense_override_codes** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_override_codes** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

> **Note**
>
> You must restart Suspense Management Center to enable it to use the new suspense override reasons.

load_pin_suspense_params

Use this utility to load system-level configuration information for Suspense Manager into the **/config/suspense_params** object in the BRM database. You define the system parameters for Suspense Manager, such as the number of records to process in each opcode call, in the **pin_suspense_params** file in the *BRM_home***/sys/data/config** directory.

> **Caution**
>
> The **load_pin_suspense_params** utility overwrites existing Suspense Manager system parameters. You must load complete sets of parameters each time you run the utility.

> **Note**
>
> To connect to the BRM database, the **load_pin_suspense_params** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_params [-d] [-v] filename
```

Parameters

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

-v
Displays detailed information as the utility runs.

filename
The name of the text file containing the configuration parameters for suspense management. The default file name is **pin_suspense_params**.

Results

The **load_pin_suspense_params** utility notifies you when it successfully creates the **/config/suspense_params** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the data was loaded, display the **/config/suspense_params** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

load_pin_suspense_reason_code

Use this utility to load suspense reasons and subreasons into the /**config/suspense_reason_code** object in the BRM database. You define suspense reasons and subreasons in the **pin_suspense_reason_code** file in *BRM_home***/sys/data/config/suspense_reason_code**.

For more information, see "Changing the List of Suspense Reasons and Subreasons".

> **Caution**
>
> The **load_pin_suspense_reason_code** utility overwrites existing suspense reason and subreason codes. If you are updating suspense reason and subreason codes, you cannot load new codes only. You must load complete sets of codes each time you run the utility.

> **Note**
>
> To connect to the BRM database, the **load_pin_suspense_reason_code** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_reason_code [-d] [-v] pin_suspense_reason_code_file
```

Parameters

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

-v
Displays detailed information as the utility runs.

pin_suspense_reason_code_file
The name and location of the file that defines suspense reasons and subreasons. The default **pin_suspense_reason_code** file is in *BRM_home***/sys/data/config**.
If you copy the **pin_suspense_reason_code** file to the same directory from which you run the **load_pin_suspense_reason_code** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_reason_code** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_reason_code** utility notifies you when it successfully creates the **/config/suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

> **Note**
>
> If you are changing the suspense reason or subreason codes, you must also modify the **suspense_reason_code.en_US** file and run the **load_localized_strings** utility. See "Configuring Suspense and Recycle on the BRM Server".

load_pin_batch_suspense_override_reason

Use the **load_pin_batch_suspense_override_reason** utility to load batch suspense override-able reason codes into the **/config/batch_suspense_override_reason** object in the BRM

database. You define batch suspense override reason codes in the **pin_batch_suspense_override_reason** file in *BRM_home***/sys/data/config**. By default, no reason can be overridden, so the file is a placeholder.

> **⚠ Caution**
>
> The **load_pin_batch_suspense_override_reason** utility overwrites the existing **/config/batch_suspense_override_reason** object in the BRM database. If you are updating the **/config/batch_suspense_override_reason** object, you must load complete sets of batch suspense override-able reasons each time.

> **ⓘ Note**
>
> To connect to the BRM database, the utility needs the Connection Manager (CM) to be up and running.

Location

*BRM_home***/bin**

Syntax

```
load_pin_batch_suspense_override_reason [-d] [-v] pin_batch_suspense_override_reason_file
```

Parameters

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

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

> **ⓘ Note**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_batch_suspense_override_reason** *any_other_parameter* **-v >** *filename***.log**.

pin_batch_suspense_override_reason_file
The name and location of the file that defines batch suspense override-able reason codes. The default **pin_batch_suspense_override_reason** file is in *BRM_home***/sys/data/config**. If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip**
>
> If you copy the **pin_batch_suspense_override_reason** file to the directory from which you run the **load_pin_batch_suspense_override_reason** utility, you don't have to specify the path or file name. The file must be named **pin_batch_suspense_override_reason**.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the override reason codes were loaded, display the **/config/batch_suspense_override_reason** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

The following is an example of a **pin_batch_suspense_override_reason** file, which would be an input for this utility and could be compared to the output:

```
# Override Suspense Reason000010000200003
```

load_pin_batch_suspense_reason_code

Use the **load_pin_batch_suspense_reason_code** utility to load batch suspense reason codes into the **/config/batch_suspense_reason_code** object in the BRM database. You define batch suspense reasons in the **pin_batch_suspense_reason_code** file in *BRM_home***/sys/data/config**. BRM uses suspense reason codes to load suspense reasons into a batch suspense record when a call details record (CDR) file is suspended.

> **Caution**
>
> The **load_pin_batch_suspense_reason_code** utility overwrites the existing **/config/batch_suspense_reason_code** object in the BRM database. If you are updating load batch suspense reason codes, you cannot load new batch suspense reason codes only. Therefore, you must load a complete set of load batch suspense reason codes each time you run the utility.

> **Note**
>
> The **load_pin_batch_suspense_reason_code** utility must be connected to a running CM to load batch suspense reason codes into the database.

Location

*BRM_home***/bin**

Syntax

load_pin_batch_suspense_reason_code [**-d**] [**-v**] *pin_batch_suspense_reason_code_file*

Parameters

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

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

> **Note**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_batch_suspense_reason_code** *any_other_parameter* **-v >** *filename***.log**.

pin_batch_suspense_reason_code_file
The name and location of the file that defines the batch suspense reason codes. The default **pin_batch_suspense_reason_code** file is in *BRM_home***/sys/data/config**.
If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip**
>
> If you copy the **pin_batch_suspense_reason_code** file to the directory from which you run the **load_pin_batch_suspense_reason_code** utility, you do not have to specify the path or file name. The file must be named **pin_batch_suspense_reason_code**.

Results

The **load_pin_batch_suspense_reason_code** utility notifies you when it successfully creates the **/config/batch_suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the elements were loaded, display the **/config/batch_suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects" and "Using testnap to Modify /config Objects" in *BRM Developer's Guide*.

The following example shows sample entries from the **/config/batch_suspense_reason_code** object:

```
PIN_FLD_POID                    POID [0] 0.0.0.1 /config/batch_suspense_reason_code
93712 00 PIN_FLD_CREATED_T             TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000
PM0 PIN_FLD_MOD_T                TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000 PM0
PIN_FLD_READ_ACCESS               STR [0] "G"0 PIN_FLD_WRITE_ACCESS              STR
[0] "S"0 PIN_FLD_ACCOUNT_OBJ             POID [0] 0.0.0.1 /account 1 00
PIN_FLD_DESCR                     STR [0] ""0 PIN_FLD_HOSTNAME                  STR
[0] "-"0 PIN_FLD_NAME                     STR [0] "batch_suspense_reason_code"0
PIN_FLD_OP_CORRELATION_ID         STR [0] "2:CT1255:UnknownProgramName:0:AWT-
EventQueue-0:3:1153836497:0:root.0.0.0.1::user1:123456789"0
PIN_FLD_PROGRAM_NAME              STR [0] "load_pin_batch_suspense_reason_code"0
```

```
PIN_FLD_VALUE                       STR [0] ""0 PIN_FLD_VERSION                    STR
[0] "1"0 PIN_FLD_SUSPENSE_REASONS         ARRAY [0] allocated 1, used 11
PIN_FLD_SUSPENSE_REASON       ENUM [0] 00 PIN_FLD_SUSPENSE_REASONS         ARRAY [471]
allocated 1, used 11     PIN_FLD_SUSPENSE_REASON       ENUM [0] 10
PIN_FLD_SUSPENSE_REASONS         ARRAY [119] allocated 1, used 11
PIN_FLD_SUSPENSE_REASON       ENUM [0] 20 PIN_FLD_SUSPENSE_REASONS         ARRAY [120]
allocated 1, used 11
   PIN_FLD_SUSPENSE_REASON       ENUM [0] 20 PIN_FLD_SUSPENSE_REASONS         ARRAY
[126] allocated 1, used 11     PIN_FLD_SUSPENSE_REASON       ENUM [0] 20
PIN_FLD_SUSPENSE_REASONS         ARRAY [127] allocated 1, used 11
PIN_FLD_SUSPENSE_REASON       ENUM [0] 20 PIN_FLD_SUSPENSE_REASONS         ARRAY [147]
allocated 1, used 11     PIN_FLD_SUSPENSE_REASON       ENUM [0] 20
PIN_FLD_SUSPENSE_REASONS         ARRAY [148] allocated 1, used 11
PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
```

pin_recycle

Use this utility to search for failed records in the BRM database, and either queue the records for recycling or delete them. See "About Suspending and Recycling Event Records."

Location

BRM_home/**bin**

Syntax

pin_recycle [**-f** *CDR_file*] [**-k** *recycle_key*] **[-d|-D|-r** *reason_code*|**-t**]

Parameters

-f *CDR_file*
Queues all the failed records that arrived in a single file.

-k *recycle_key*
Searches for and queues records for rating that contain:

- The *recycle_key*, an application-specific string that is added to each record as it is suspended.
- A status of **suspended**.

-d
Searches for and deletes all records with a status of **succeeded** or **written off**.

-D
Searches for and deletes all records with a status of **succeeded**, **written off**, or **suspended**.

-r *reason_code*
Searches for and recycles all records that have the specified reason code.

-t
Specifies a test recycle. In test mode, **pin_recycle** creates a report about the processing, but does not make any changes to the database.

Results

This utility logs messages to **stdout**.