Oracle® Communications Billing and Revenue Management
Loading Rated Events

ORACLE®

Contents

Preface

1   About Loading Rated Events into the BRM Database

2   Installing Rated Event Loader

8 Customizing Rated Event Loader

9 Troubleshooting Rated Event Loading

10 Improving Rated Event Loader Performance

ORACLE

11   Rated Event Loader pin_rel Utility

12   Rated Event Loader Manager Utility

Preface

This guide describes how to load rated events from Elastic Charging Engine (ECE) into the Oracle Communications Billing and Revenue Management (BRM) database by using Rated Event Loader (RE Loader).

Audience

This guide is intended for system administrators.

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

1
About Loading Rated Events into the BRM Database

Learn about loading rated events into the Oracle Communications Billing and Revenue Management (BRM) database from Elastic Charging Engine (ECE) using Rated Event Loader (RE Loader).

Topics in this chapter:

- How Rated Events are Loaded Into the BRM Database
- About Backing Up RE Loader Files
- About Using RE Loader in a Multischema System
- About Configuring Rated Event Loading

See also:

- Installing Rated Event Loader
- Configuring the Rated Event Loader Infranet.properties File

How Rated Events are Loaded Into the BRM Database

The RE Loader can be used by both the Pipeline Manager and ECE to load rated events into the BRM database. For information about using RE Loader with Pipeline manager, see "Loading Rated Events" in *Configuring Pipeline Rating and Discounting*. There are different methods of loading the records, depending on the event types, whether the records are coming from Pipeline Manager or ECE, and your business needs.

Methods of Transferring Rated Events to BRM

There are several methods of transferring rated events to BRM. The original method is the most flexible, supporting all events from both ECE and Pipeline Manager. There is also the Rated Event Manager (RE Manager), which has higher performance than the original method and includes five different modes of operation, equating to five different methods of transferring rated events to BRM. The RE Manager has a plug-in for the Rated Event Formatter (RE Formatter) in ECE and a daemon in BRM. The RE Manager can be used only for ECE (not for Pipeline Manager), and it can be used for all types of events from ECE except for suspense management events.

The following methods of transferring rated events are available:

- Original Method
- DIRECT Mode
- ZIP_DB Mode
- ZIP_FILE Mode
- CDR Mode
- Event Streaming Mode

Original Method

The original method can be used for all types of rated events, and for files from both ECE and Pipeline Manager. It transfers files to an intermediate file system where they are picked up by BRM. For information about using RE Loader with Pipeline manager, see "Loading Rated Events" in *Configuring Pipeline Rating and Discounting*.

The process when this method is used with ECE is:

1. In the ECE system, the Elastic Charging Server (ECS) publishes rated events to an Oracle NoSQL database or Oracle Database.

2. RE Formatter uses the BRM call detail record (CDR) plug-in to write the rated events to a CDR file and send the file to a specified directory.

3. Rated Event Loader (RE Loader) retrieves the CDR file from the directory and loads the rated events into the BRM database. RE Formatter and RE Loader can run on the same system or different systems, as long as they can both access the CDR file directory.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader

- Configuring Rated Event Output from ECE

- Configuring BRM Server Components for Rated Event Loading

- Configuring the Rated Event Loader Infranet.properties File

- Configuring Rated Event Loader to Run Automatically

- Customizing Rated Event Loader

- Troubleshooting Rated Event Loading

- Improving Rated Event Loader Performance

- Rated Event Loader pin_rel Utility

DIRECT Mode

This mode loads the data directly from ECE into the regular tables in the BRM database. This is the most direct method, but it uses more bandwidth than the ZIP_DB option. The process when this mode is used is:

1. In the ECE system, ECS publishes rated events to an Oracle NoSQL database or Oracle Database.

2. RE Formatter uses the ECE RE Manager Plug-in to write the rated events to temporary CDR files. The RE Manager Loader inserts the CDRs directly into the BRM database.

3. The RE Manager Loader passes information to the RE Manager Updater, which updates the balances in the BRM database.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader

- Configuring Rated Event Output from ECE

- Configuring BRM Server Components for Rated Event Loading

- Configuring Rated Event Manager

- Rated Event Loader Manager Utility

ZIP_DB Mode

This mode creates a ZIP file and stores it in the /batch/rel object in the database, where a BRM process loads it into the regular tables. The process when this mode is used is:

1. In the ECE system, ECS publishes rated events to an Oracle NoSQL database or Oracle Database.

2. RE Formatter uses the ECE RE Manager Plug-in to write the rated events to temporary CDR files. The RE Manager compresses the CDRs and inserts the ZIP file directly into a new table in the BRM database.

3. The RE Manager Zip Transfer Processor in BRM reads the ZIP file and updates the RE Manager Loader and the RE Manager Updater.

4. The RE Manager Loader inserts the records into the BRM Database and the RE Manager Updater updates the balances.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader
- Configuring Rated Event Output from ECE
- Configuring BRM Server Components for Rated Event Loading
- Configuring Rated Event Manager
- Troubleshooting Rated Event Loading
- Rated Event Loader Manager Utility

ZIP_FILE Mode

This mode compresses the CDRs in a transaction into a single file before transferring it to an intermediate file system where it is picked up by BRM. The process when this mode is used is:

1. In the ECE system, ECS publishes rated events to an Oracle NoSQL database or Oracle Database.

2. RE Formatter uses the ECE RE Manager Plug-in to write the rated events to temporary CDR files. The RE Manager compresses the CDRs and creates the ZIP file on the specified file system.

3. The RE Manager Directory Processor in BRM reads the ZIP file and updates the RE Manager Loader and the RE Manager Updater.

4. The RE Manager Loader inserts the records into the BRM Database and the RE Manager Updater updates the balances.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader
- Configuring Rated Event Output from ECE
- Configuring BRM Server Components for Rated Event Loading
- Configuring Rated Event Manager
- Troubleshooting Rated Event Loading
- Rated Event Loader Manager Utility

ORACLE

CDR Mode

This mode is similar to the original method. It transfers individual files to an intermediate file system, where they are picked up by BRM. You might choose this method if you wanted to read or perform some action on the files before they were picked up by BRM. The process when this mode is used is:

1. In the ECE system, ECS publishes rated events to an Oracle NoSQL database or Oracle Database.

2. RE Formatter uses the ECE RE Manager Plug-in to write the rated events to temporary CDR files. RE Manager in CDR Mode writes the rated events to CDR files and creates the files in a specified directory.

3. The RE Manager Directory Processor in BRM reads the CDR files and updates the RE Manager Loader and the RE Manager Updater.

4. The RE Manager Loader inserts the records into the BRM Database and the RE Manager Updater updates the balances.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader

- Configuring Rated Event Output from ECE

- Configuring BRM Server Components for Rated Event Loading

- Configuring Rated Event Manager

- Rated Event Loader Manager Utility

Event Streaming Mode

This mode streams events to and from a Kafka topic. The process when this mode is used is:

1. In the ECE system, RE Formatter publishes rated events to one or more Kafka topics. These topics can be subscribed to from external systems, such as analytics and reporting systems.

2. The RE Manager Stream Processor in BRM reads the messages from the Kafka stream and updates the RE Manager Loader and the RE Manager Updater.

3. The RE Manager Loader inserts the records into the BRM Database and the RE Manager Updater updates the balances.

The following chapters contain information that is relevant to using this processing method:

- Installing Rated Event Loader

- Configuring Rated Event Output from ECE

- Configuring BRM Server Components for Rated Event Loading

- Configuring Rated Event Manager

- Rated Event Loader Manager Utility

Rated Event Loading

RE Manager Loader and RE Loader load rated events directly into the BRM database, bypassing the Connection Manager (CM) and Data Manager (DM). RE Loader then updates account balances, billing items, and journals in the BRM database.

RE Loader uses a partitioned database and inserts pre-rated events into separate partitions allocated for delayed events. The events are called "delayed" because they are rated before they are loaded, and there is a delay between the two actions.

> **✎ Note:**
>
> You *must* partition your database when loading pre-rated events. See "Partitioning Database Tables" in *BRM System Administrator's Guide*.

By default, RE Loader loads the events shown in Table 1-1. However, you can configure RE Loader to load custom events. See "Adding New Types of Events for RE Loader to Load".

Table 1-1   Services and Events Loaded by Default

Service	Event
GPRS (General Packet Radio Service)	**/event/delayed/session/gprs**
GSM (Global System for Mobile Communication)	**/event/delayed/session/telco/gsm**

About Running RE Loader

You can run RE Loader in the following ways:

- By using Batch Controller (see "Configuring RE Loader to Run Automatically by Using Batch Controller"). Batch Controller detects when a file is present in the RE Formatter output directory and starts the RE Loader batch handler. RE Loader batch handler starts the RE Loader utility (**pin_rel**) to load the events. RE Loader batch handler moves the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded.

- By using the RE Loader daemon (see "Configuring RE Loader to Run Automatically by Using the RE Loader Daemon"). The RE Loader daemon detects when CDR files arrive in the input directory, and then processes them. The RE Loader daemon moves the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded.

If your CDR files are small, Oracle recommends loading events by using the RE Loader daemon. You should compare the two methods by running them both in a test system.

You can also run RE Loader manually, by running the **pin_rel** utility. See "Rated Event Loader pin_rel Utility".

If any errors occur during event loading, all events loaded in that session are deleted from the database. After events are successfully loaded, if any errors occur during the update procedure, you can correct the errors and then update the relevant events by rerunning the RE Loader utility. The utility detects that the events are loaded correctly and performs only the update procedure. See "Troubleshooting Rated Event Loading".

About Backing Up RE Loader Files

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

To prevent data loss when your system shuts down:

- Make full backups of the BRM database on a regular basis.

- Archive all successfully loaded files until you make a full database backup.

You can re-enable redo generation, at the cost of loading performance, by modifying the RE Loader control files. See "Configuring Whether to Perform Redo Generation".

About Using RE Loader in a Multischema System

If you use a multischema system, you must set up the following for each BRM database schema in your system:

- RE Loader instance. Each instance of RE Loader must also have its own set of RE Loader processing directories.

- (Running RE Loader automatically only) An instance of Batch Controller and the RE Loader batch handler.

About Configuring Rated Event Loading

To retrieve and load rated events:

- Configure the ECE Rated Event Formatter component. See "Configuring Rated Event Output from ECE".

- Configure BRM system components to enable proper loading of rated events. For example, to use RE Loader, you must disable invoice event caching. See "Configuring BRM Server Components for Rated Event Loading".

- Configure the RE Formatter BRM CDR plug-in to transfer events from ECE to RE Loader. See "Configuring RE Formatter Output".

- Configure the RE Loader **Infranet.properties** file. See "Configuring the Rated Event Loader Infranet.properties File".

- Configure RE Loader to load events automatically. See "Configuring Rated Event Loader to Run Automatically".

2

Installing Rated Event Loader

Learn how to install the Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader) software, which includes the Rated Event Manager (RE Manager) software.

Topics in this chapter:

- Installing RE Loader
- Preventing POID Errors in Multischema Systems
- Uninstalling RE Loader

See also:

- About Loading Rated Events into the BRM Database
- Configuring the Rated Event Loader Infranet.properties File

Installing RE Loader

To install RE Loader and RE Manager, perform the procedures in these sections:

1. Granting Execute Permission for dbms_lock
2. Granting Write Permission to the Data Manager
3. Installing the RE Loader Package
4. Creating RE Loader Database Partitions
5. Returning Data Manager Permissions to their Original Values

Granting Execute Permission for dbms_lock

Before you install RE Loader, you must grant execute permission to *pin_user* for **dbms_lock**:

1. Log in to your database as the SYS user:

   ```
   % sqlplus sys@databaseAlias
   Enter password: password
   ```

2. Grant execute privileges to *pin_user*:

   ```
   SQL> grant execute on dbms_lock to pin_user
   ```

Granting Write Permission to the Data Manager

When you install RE Loader on a system where BRM is not installed, you must grant the Data Manager (DM) write permission before installing RE Loader.

Perform the following on all machines containing a DM:

1. In a text editor, open your DM configuration file:

   *BRM_home***/sys/dm_oracle/pin.conf**

   BRM_home is the directory in which you installed BRM components.

2. Write down the values of your **dd_write_enable_fields**, **dd_write_enable_objects**, **dd_write_enable_portal_objects**, and **dd_mark_as_portal** entries.

3. Set the values of the following entries to **1**:

   ```
   - dm dd_write_enable_fields 1
   - dm dd_write_enable_objects 1
   - dm dd_write_enable_portal_objects 1
   - dm dd_mark_as_portal 1
   ```

   > **Note:**
   >
   > If any entry is not in the file, add it.

   For more information, see the information in the DM **pin.conf** file.

4. Save and close the file.

5. Stop and restart the DM.

   You can now install RE Loader.

Installing the RE Loader Package

To install RE Loader, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Creating RE Loader Database Partitions

> **Note:**
>
> You must perform this step to ensure that the new event tables have the same partitioning layout as your existing event tables. If you install several optional components, perform this step only after installing the last component.

To create partitions for RE Loader events:

1. On the system where BRM is installed, go to the *BRM_home***/apps/partition_utils** directory.

2. Run the **partition_utils** utility to enable delayed-event partitions:

   ```
   perl partition_utils.pl -o enable -t delayed -c storable_class
   ```

   where *storable_class* specifies the event classes for which you want partitioning.

> **Note:**
>
> You must create partitions for all subclasses of a specific event that you want to load.

For example, this command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

```
perl partition_utils.pl -o enable -t delayed -c /event/session/telco/gsm
```

For more information, see:

- "Enabling Delayed-Event Partitioning" in *BRM System Administrator's Guide*
- "partition_utils" in *BRM System Administrator's Guide*.

Your RE Loader installation is now complete.

Returning Data Manager Permissions to their Original Values

To return your DM permissions to their original values:

1. In a text editor, open your DM configuration file:

   *BRM_home***/sys/dm_oracle/pin.conf**

2. Restore the following entries to their original values (the values they had before you modified them). The default value for each entry is **0**:

   ```
   - dm dd_write_enable_fields
   - dm dd_write_enable_objects
   - dm dd_write_enable_portal_objects
   - dm dd_mark_as_portal
   ```

3. Save and close the file.

4. Stop and restart the DM.

Preventing POID Errors in Multischema Systems

BRM multischema systems ensure that all POIDs are unique across all database schemas by using a POID-generation algorithm. This BRM algorithm sets each schema's starting sequence number to a unique value and then increments each sequence number by a set value. By default, BRM sets the increment value equal to the number of schemas in your system.

For example, if your system contains three schemas:

- Schema 1 uses a starting sequence number of 10000
- Schema 2 uses a starting sequence number of 10001
- Schema 3 uses a starting sequence number of 10002

The incremental value is 3.

This example results in the following POID numbers shown in Table 2-1:

Table 2-1   Example Schema POID Numbers

Time	POID for Schema 1	POID for Schema 2	POID for Schema 3
1	10000	10001	10002
2	10003	10004	10005
3	10006	10007	10008

When RE Loader loads a batch of objects into the BRM database, it reserves a group of POIDs as follows:

1. Changes the increment value by using the following equation:

   ```
   (Number of objects to load) x (Current increment value)
   ```

   For example, if RE Loader must load 2,000 objects into the database and the current increment value is 3, it changes the increment value to 2,000 x 3 = 6,000.

2. Allocates POIDs to objects.

3. Returns the increment value to its original value.

However, if a major error occurs during the allocation process, the increment value can remain at the incorrect high value. To catch these situations, you can configure RE Loader to check the database increment value against a specified maximum before it reserves a group of POIDs. When the increment value exceeds the specified maximum, RE Loader exits and logs an error message, notifying your database administrator to manually reset the increment value.

To configure RE Loader to compare the increment value against a specified maximum:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.

2. Set the **infranet.rel.max_increment_by** entry to the number of database schemas in your system:

   ```
   infranet.rel.max_increment_by = 20
   ```

   The default is **20**.

3. Save and close the file.

Uninstalling RE Loader

To uninstall RE Loader, see "Uninstalling Optional Components" in *BRM Installation Guide*.

3

Configuring Rated Event Output from ECE

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) Elastic Charging Engine (ECE) for the Rated Event Loader (RE Loader) and Rated Event Formatter (RE Formatter) to process events.

Topics in this chapter:

- Adding an RE Formatter Instance
- Configuring RE Formatter
- Adding a BRM Plug-in Instance for RE Formatter
- Configuring RE Formatter Output
- Troubleshooting Rated Event Formatter Processing

See also:

- About Loading Rated Events into the BRM Database
- Configuring BRM Server Components for Rated Event Loading
- Configuring the Rated Event Loader Infranet.properties File
- Improving Rated Event Loader Performance
- Troubleshooting Rated Event Loading

Adding an RE Formatter Instance

To add an RE Formatter instance:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

2. Expand the **ECE Configuration** node.

3. Expand **charging.ratedEventFormatters**.

4. Expand **Operations**.

5. Select **addRatedEventFormatterConfiguration**.

6. Enter a value for the **name** parameter.

7. Click **addRatedEventFormatterConfiguration**.

8. Use Elastic Charging Controller (ECC) to start the RE Formatter instance.

Configuring RE Formatter

You can use this procedure to configure and tune the RE Formatter. You can use this procedure to configure which plug-in the RE Formatter will use to process events, either the default plug-in for the original processing method or the ECE Rated Event Manager plug-in. See the pluginType entry in Table 3-1 for information about this configuration. See "Methods of

Transferring Rated Events to BRM" for more information about the different processing methods.

To configure RE Formatter, do the following:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

2. Expand **charging.ratedEventFormatters.**_instance_name_, where _instance_name_ is the name of the instance you want to configure.

3. Expand **Attributes**.

4. Specify values for **name**, **partition**, and **primaryInstanceName** and for any other attributes as described in Table 3-1 .

5. Use Elastic Charging Controller (ECC) to stop and restart any RE Formatter instances that you configured.

   Each instance reads its configuration information by name.

   For information about stopping and starting RE Formatter instances, see "Starting and Stopping ECE" in _BRM System Administrator's Guide_.

Table 3-1 describes the parameters you must set to configure a RE Formatter instance.

Table 3-1   RE Formatter Instance Configuration Attributes and Values

Attribute Name	Default Value	Description
name	formatter	The name of a Rated Event Formatter instance. Name Rated Event Formatter instances consistently and uniquely (for example, **formatter1**, **formatter2**, and so on).
		The value of **name** must match the name of the Rated Event Formatter node instance in the _ECE_home_/**oceceserver/config/eceTopology.conf** file.
partition	1	The partition for the rated events to be processed by a Rated Event Formatter instance. The value of **partition** must match the target BRM database schema number for the schema to which the ECE rated event is to be exported.
		For example, for a BRM multischema environment, BRM schema **0.0.0.1**, **partition** must be set to **1**; for BRM schema **0.0.0.2**, **partition** must be set to **2**; and so on. See "Configuring ECE for a Multischema BRM Environment" in _BRM System Administrator's Guide_ for more information.
		You may want to name your formatter instances to correlate with the partition number. For example, one instance of Rated Event Formatter named **formatter1** can process rated events to be exported to the BRM schema **0.0.0.1** (**partition** value **1**), and another instance of Rated Event Formatter named **formatter2** can process rated events to be exported to the BRM schema **0.0.0.2** (**partition** value **2**), and so on.
primaryInstanceName	ratedEventFormatter1	The name of the primary Rated Event Formatter instance.
connectionName	OraclePersistence1	The connection name of the persistence database. This attribute is applicable only if you are using Oracle Database for storing rated events.
dataStoreName	kvstore	The data store name to be used to access an Oracle NoSQL system.
		This attribute is applicable only if you are using Oracle NoSQL database for storing rated events.

Table 3-1   (Cont.) RE Formatter Instance Configuration Attributes and Values

Attribute Name	Default Value	Description
dataStoreConnection	localhost:5000	The connection information to the Oracle NoSQL database.
		The connection string consists of host name and port number for connecting to an Oracle NoSQL system.
		This attribute is applicable only if you are using Oracle NoSQL database for storing rated events.
threadPoolSize	4	The number of threads used by the RE Formatter instance to process a set of rated events for each time range defined by **checkPointInterval**.
		Valid values are greater than zero and up to any number the system resources allow. Tune this value to the expected workload in the deployed environment.
retainDuration	0	The duration in seconds that rated events must be retained in the Oracle NoSQL database after they have been processed before they can be purged.
		Set the value to the seconds you want to retain rated events in the Oracle NoSQL database after Rated Event Formatter has published the rated events as RE Loader records (CDR records).
		The default is **0**, which means that as soon as rated events are processed, they are purged immediately.
		This attribute is applicable only if you are using Oracle NoSQL database for storing rated events.
ripeDuration	60	The duration in seconds that rated events have existed before they can be processed.
		This setting must be greater than the time it takes for ECE charging servers to fully recover after failure. Delaying the processing of rated events up to the **ripeDuration** time allows time for resolving any duplicate rated events that may have been persisted in the Oracle NoSQL database.
		The **ripeDuration** value is the minimum number of seconds rated event information must be stored in the Oracle NoSQL Database before the Rated Event Formatter can read it.
checkPointInterval	4	The time range in seconds used by the Rated Event Formatter instance to read a set of rated events at a repeated time interval.
		Valid values must be the following:
		• Less than or equal to the value of **ripeDuration**
		• Evenly divisible by the number of threads configured for **threadPoolSize**
		This is the number of seconds Rated Event Formatter waits before reading a batch of rated event information. If rated event information in a batch has not yet met the value of **ripeDuration**, the Rated Event Formatter does not read it.
pluginPath	n/a	The JAR library path that contains the reader plug-in implementation. This value is only needed if the library is not in the class path for ECE.
pluginName	n/a	This attribute must be set to the value configured in the **charging.brmCdrPlugins.**_name_ MBean. See "Adding a BRM Plug-in Instance for RE Formatter" for more information.

Table 3-1 (Cont.) RE Formatter Instance Configuration Attributes and Values

Attribute Name	Default Value	Description
pluginType	oracle.communication.brm.charging.ratedevent.formatterplugin.internal.BrmCdrPluginDirect	Set this to **oracle.communication.brm.charging.ratedevent.formatterplugin.internal.BrmCdrPluginDirect** if you are using the original RE Formatter plug-in, or set it to **com.oracle.brm.ref_brm_plugin.RatedEventManagerCdrPlugin** to use the new Rated Event Manager functionality.
logFormatterWorker	false	Enables or disables logging for the worker thread pool.
maxPersistenceCatchupTime	0	The maximum number of seconds that Rated Event Formatter waits after a database outage to resume processing events.
siteName	n/a	The name of the site that the Rated Event Formatter instance formats events for. In an active-active deployment where data persistence is enabled, **siteName** for a primary Rated Event Formatter instance matches the site where the instance runs. For secondary Rated Event Formatter instances, **siteName** is different from the site where the instance runs, and specifies the site that the instance is a backup for. This allows the secondary instance to process the rated events from a remote site when the remote site is unavailable. See "About Rated Event Formatter in a Persistence-Enabled Active-Active System" in *BRM System Administrator's Guide*.

Changing the Event Transfer Mode in Runtime

You can use this procedure to change the rated event transfer mode without restarting the RE Formatter. You must be using the ECE Rated Event Manager plug-in for RE Formatter event processing for this procedure to work. See the pluginType entry in Table 3-1 for information about this configuration. See "Methods of Transferring Rated Events to BRM" for more information about the different processing methods.

> **Note:**
>
> Using this method to change the processing mode will not update the **REM.properties** file, and the change will not persist if the RE Formatter is restarted. You must update the **REM.properties** file **ref.mode** parameter separately if you want the change to persist. See General ECE RE Manager Plug-in Configuration for more information about configuring this value.

To configure RE Formatter to use a different processing mode on a running REF server, do the following:

1. Connect to the REF node enabled for JMX management.

   You do this by connecting to the **JMX port** set for the REF node in the *ECE_home***/oceceserver/config/eceTopology.conf** file, where *ECE_home* is the directory in which ECE is installed.

2. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.

3. In the editor's MBean hierarchy, find the **REFBRMPlugin** MBeans.

4. Expand **REFBRMPlugin.Configuration**.

5. Expand **Attributes**.

6. Specify one of the following values for **operationModeString**:

    • DIRECT

    • ZIP_DB

    • ZIP_FILE

    • CDR

    The processing mode will be changed.

Adding a BRM Plug-in Instance for RE Formatter

To add a BRM Plug-in instance for RE Formatter:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

2. Expand the **ECE Configuration** node.

3. Expand **charging.brmCdrPlugins**.

4. Expand **Operations**.

5. Select **addBrmCdrPluginConfiguration**.

6. Add configuration parameters, including a name that matches what is configured in the **pluginName** parameter in the appropriate **charging.ratedEventFormatters.***instance_name* MBean. See "Configuring RE Formatter" for more information.

7. Click **addBrmCdrPluginConfiguration**.

8. Use Elastic Charging Controller (ECC) to start the instance of Rated Event Formatter that the BRM CDR plug-in is associated with. This is defined in the Rated Event Formatter MBean (**charging.ratedEventFormatters**) **pluginName** attribute.

Configuring RE Formatter Output

To configure RE Formatter output, you configure the plug-in run by RE Formatter:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

2. Expand the **ECE Configuration** node.

3. Expand **charging.brmCdrPlugins.***instance_name*.

4. Expand **Attributes**.

5. Specify values for the following attributes as needed:

Attribute Name	Default	Description
tempDirectoryPath	/tmp/tmp	The directory path for the BRM CDR Plug-in to store temporary files while processing the rated events.

Attribute Name	Default	Description
doneDirectoryPath	/home/charging/ rel_input/	The directory path for the BRM CDR Plug-in to store completed CDR files from processing the rated events. The values for **doneDirectoryPath**, **headerFileDirectoryPath**, and **dataFileDirectoryPath** must be the same for an instance.
doneFileExtension	.done	The file extension for the completed CDR files created by the BRM CDR Plug-in.
headerFileDirectoryPath	/home/charging/ rel_input/	The directory path for the BRM CDR Plug-in to store the files that define the format of the CDRs. The values for **doneDirectoryPath**, **headerFileDirectoryPath**, and **dataFileDirectoryPath** must be the same for an instance.
headerFileExtension	.hdr	The file extension for the CDR header files from the BRM CDR Plug-in.
dataDirectoryPath	/home/charging/ rel_input/	The directory path for the BRM CDR Plug-in to store the rated CDR files for processing. The values for **doneDirectoryPath**, **headerFileDirectoryPath**, and **dataFileDirectoryPath** must be the same for an instance.
dataFileExtension	.data	The file extension for the rated CDR files for processing by the BRM CDR Plug-in.
enableInvalidRatedEvents	false	Specifies whether RE Formatter moves invalid events to **invalidRatedEventDirectoryPath**. **Note**: Rated events can be invalidated for a variety of reasons, such as a missing balance impact. You can drop or reprocess invalid events from NoSQL by restarting RE Formatter.
invalidRatedEventDirectoryPath	/tmp/dc2/secondary/ invalid	RE Formatter moves invalid events to this directory if **enableInvalidRatedEvents** is set to **true**.

6. Stop and start the RE Formatter instance associated with the BRM Plug-in instance.

Troubleshooting Rated Event Formatter Processing

If you suspect a problem with how Rated Event Formatter processes rated events, look in the *ECE_home*/**oceceserver/logs/rated-event-formatter.log** file for errors. The log file contains information about how many rated events are processed, how many are purged, and any errors during rated-event processing.

Accessing ECE Configuration MBeans

For all configurations, start by accessing the ECE configuration MBeans:

1. Log on to the driver machine.

2. Start the ECE charging servers (if they are not started).

3. Connect to the ECE charging server node enabled for JMX management.

This is the charging server node set to **start CohMgt = true** in the *ECE_home*/
oceceserver/config/eceTopology.conf file, where *ECE_home* is the directory in which
ECE is installed.

4. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.

5. In the editor's MBean hierarchy, find the ECE configuration MBeans.

4

Configuring BRM Server Components for Rated Event Loading

Learn how to set up Oracle Communications Billing and Revenue Management (BRM) server components to use Rated Event Loader (RE Loader) to load events.

Topics in this chapter:

- Enabling a Billing Delay for CDRs
- Disabling Invoice Event Caching
- Setting Up RE Loader for Virtual Column-Enabled Systems
- Configuring Whether to Perform Redo Generation
- Configuring Field Lengths for Input Data Files

See also:

- About Loading Rated Events into the BRM Database
- Configuring Rated Event Output from ECE
- Configuring the Rated Event Loader Infranet.properties File
- Improving Rated Event Loader Performance
- Troubleshooting Rated Event Loading

Enabling a Billing Delay for CDRs

Offline charging often results in events arriving to be rated after the billing date that they belong to. For example, an event that occurred on December 31 might arrive on January 2nd, past the January 1st billing date. To handle this delay, you must configure delayed billing.

To enable delayed billing, see "Setting Up Delayed Billing" in *BRM Configuring and Running Billing*.

Disabling Invoice Event Caching

If your system uses both RE Loader and invoicing, you must disable invoice event caching to ensure that invoices contain event details.

To disable invoice event caching, disable the **event_cache** entry in the CM pin.conf file. See "Improving Performance by Customizing Event Caching" in *Designing and Generating Invoices* for more information about this setting.

Setting Up RE Loader for Virtual Column-Enabled Systems

If you generate virtual columns on event tables in your BRM installation, you must run the **pin_gen_classid_values.pl** script. Running the script ensures that the proper mapping of BRM object types and their corresponding object IDs is created for your extended event

objects in a virtual column-enabled system. For information about enabling virtual columns in the BRM database, see the discussion on virtual columns in *BRM System Administrator's Guide*.

Configuring Whether to Perform Redo Generation

This information applies only to the original processing method.

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

You can re-enable redo generation by removing the UNRECOVERABLE option from each RE Loader control file.

To enable redo generation, do the following for each table's control file:

1. Open the *BRM_home***/apps/pin_rel/***control_file* file in a text editor, where *control_file* can be one of the files shown in Table 4-1.

Table 4-1   RE Loader Control Files

File Name	Description
event_bal_impacts_t.ctl	Control file for the EVENT_BAL_IMPACTS_T table.
event_delayed_act_wap_inter_t.ctl	Control file for the EVENT_DELAYED_ACT_WAP_INTER_T table.
event_delayed_session_gprs_t.ctl	Control file for the EVENT_DELAYED_SESSION_GPRS_T table.
event_sub_bals_t.ctl	Control file for the EVENT_SUB_BALS_T table.
event_sub_bal_imp_t.ctl	Control file for the EVENT_SUB_BAL_IMP_T table.
event_dlay_sess_tlcs_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_T table.
event_dlay_sess_tlcs_svc_cds_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_SVC_CDS_T table.
event_t.ctl	Control file for the EVENT_T table.
event_total_t.ctl	Control file for the EVENT_TOTAL_T table.
event_dlyd_session_tlco_gsm_t.ctl	Control file for the EVENT_DLYD_SESSION_TLCO_GSM_T table.

2. Comment out the UNRECOVERABLE option:

    ```
    # UNRECOVERABLE
    ```

    > **✎ Note:**
    >
    > Removing the UNRECOVERABLE option significantly decreases loading performance.

3. Save and close the file.

Configuring Field Lengths for Input Data Files

Any value in the input data file longer than 255 characters must include its maximum size. If the maximum size is not specified, the value is truncated to 255 characters when it is loaded into the database.

Fields in the input data file that should not be loaded into the database are specified with the label FILLER in the SQL Loader control file. If the input data file contains a FILLER field with a value longer than 255 characters, SQL Loader aborts with an error specifying the field at fault. If this happens, add the maximum field size to the field entry in the SQL Loader control file. Use this syntax:

Field_name   FILLER CHAR(*max_size*)

For example:

DISCOUNT_INFO   FILLER CHAR(2000)

5

Configuring the Rated Event Loader Infranet.properties File

Learn how to configure the Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader) **Infranet.properties** file.

This information applies only to the original processing method.

Topics in this chapter:

See also:

Configuring the RE Loader Infranet.properties File

To configure RE Loader, you edit the RE Loader **Infranet.properties** file.

The **Infranet.properties** file contains configuration information for processing CDR files, such as the location of the RE Loader processing directory, how to connect to the BRM database, and how to process specific events.

> ✎ **Note:**
>
> The entry descriptions might include one or both of the following values:
>
> - **Initial value:** The value to which the RE Loader installation program sets the entry.
> - **Default:** The value that RE Loader uses when the entry is missing or has no value.

To configure your RE Loader **Infranet.properties** file:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.
2. Edit the file.
3. Save and close the file.

BRM Database Connection

Configure the following entries in the RE Loader Infranet.properties file to connect to the BRM database.

Table 5-1   Database Connection Entries

Entry	Description
infranet.rel.dbtype	Specifies the BRM database type. The initial value is **oracle**. The default is **ORACLE**.
infranet.rel.dbname	Specifies the BRM database name. **Note**: Your database name is the TNSNAMES alias in the *Oracle_home*/**network/admin/tnsnames.ora** file. Ensure that this value is correct for your system.
infranet.rel.userid	Specifies the user ID for connecting to the BRM database. Ensure that this value is correct for your system.
infranet.rel.password	Specifies the password for connecting to the BRM database. Ensure that this value is correct for your system.
infranet.rel.dbhost	Specifies the database machine's host name.
infranet.rel.dbport	Specifies the database port number. The initial value and the default are **1433**.
infranet.rel.partition_set_number	Specifies the partition set number, from 1 through 7. This entry applies only to BRM databases with multiple delayed partition sets. The initial value and the default are **1**. • **1** uses delayed partition set P_1D to P_12D. • **2** uses delayed partition set P_1D to P_12D2. • **3** uses delayed partition set P_1D to P_12D3. • **4** uses delayed partition set P_1D to P_12D4. • **5** uses delayed partition set P_1D to P_12D5. • **6** uses delayed partition set P_1D to P_12D6. • **7** uses delayed partition set P_1D to P_12D7.

BRM Server Connection

Configure the following entries in the RE Loader **Infranet.properties** file to configure how log files are created.

Table 5-2   BRM Server Connection Entries

Entry	Description

infranet.connection	Specifies the user login name. For example:
	
	`infranet.connection=pcp://root.0.0.0.1:`*`password`*`@localhost:11960/`
	`service/pcm_client`
	
	RE Loader uses this Connection Manager (CM) connection to log audit information.
	
	Important: RE Loader writes audit information to the database specified in this entry. If you use a multischema system, you might want to modify this entry to write audit information to the schema where the records are loaded.

infranet.login.type	Specifies whether RE Loader requires a login name and password to log in to BRM. • **0** specifies that a login name and password *are not* required. • **1** specifies that a login name and password *are* required. The initial value is **1**. No default exists.

infranet.failover	In high-availability systems, specifies the secondary CM connection. For example:
	
	`infranet.failover.1 = pcp://`
	`root.0.0.0.`*`db_no`*`:`*`password`*`@`*`failover_host`*`:`*`failover_port`*`/service/`
	`pcm_client`

infranet.rel.polling_interval	Specifies the interval, in milliseconds, that RE Loader checks the database to see whether another process is loading. The initial value and the default are **1000**. The polling interval depends on the number and size of your input files. If you have very large files, make the polling interval longer. If you have many small files, make the interval shorter.
infranet.rel.polling_time_out	Specifies the time, in milliseconds, that RE Loader waits to load events before exiting. The initial value and the default are **600000**. The time-out period depends on the number and size of your input files and how many parallel RE Loader processes are running. If you have very large files or many processes, make the time-out period longer.
infranet.rel.updater_threads	Specifies the number of threads dedicated to the update and preupdate stored procedures. You can specify a fixed number of threads or configure RE Loader to adjust the number of threads based on the number of database objects to update. To specify a fixed number of threads, set the entry equal to the desired number of threads. To configure RE Loader to automatically adjust the number of threads, set the entry to **0**. RE Loader spawns the number of threads shown below: Less than 1,000 objects: 2 threads Between 1,000 and 200,000 objects: 4 threads More than 200,000 objects: 8 threads The initial value is **4**. The default is **1**. **Note:** Specifying a number of threads that exceeds the number of CPUs in your system may cause deadlock due to a lack of system resources. If you set the **infranet.rel.updater_threads** entry to a value greater than 8, RE Loader returns a warning message and continues processing.

ORACLE

Table 5-2   (Cont.) BRM Server Connection Entries

Entry	Description
infranet.rel.validate_dbnumber	Specifies whether RE Loader performs an extra validation step to ensure that it is loading a call detail record (CDR) file into the correct database schema. The initial value and the default are **true**. **Important:** Use this option only for debugging. In a production environment, set this to **False**. Setting it to **True** degrades performance while loading data into the database. See "Turning Off Database Verification to Improve Processing Performance".
infranet.rel.validate_indexes	Specifies whether RE Loader verifies that the database indexes are correct before loading data into the database. The initial value is **false**. The default is **true**. **Important:** Use this option only for debugging. In a production environment, set this to **false**. Setting it to **true** degrades performance while loading data into the database. See "Turning Off Index Verification to Improve Database Loading Performance".
infranet.rel.max_increment_by	Specifies the number of database schemas in your system. This value is used by the POID generation algorithm to ensure that POIDs are unique across all database schemas in your system. The initial value and the default are **20**. See "Preventing POID Errors in Multischema Systems".
infranet.rel.sort.limit	Defines the maximum number of CDRs that the preprocessing script can sort by account ID. This improves performance later during the balance updating process. If the number of CDRs in the input file is greater than the **infranet.rel.sort.limit** value, the preprocessing script does not sort the CDRs. The initial value is **100000**. The default is **500000**.

RE Loader Daemon

To run the RE Loader daemon, add or modify the RE Loader daemon **Infranet.properties** file entries shown in Table 5-3.

Table 5-3   RE Loader Daemon Entries

Entry	Description
batch.check.interval	Specifies the time interval, in seconds, to monitor files from the output directory. The default is **5**.
batch.file.rename.extension	Specifies the file name extension that the RE Loader daemon uses to rename the interim files before processing them. The default is **.bc**.
batch.start.highload.time	Specifies the start time of your system's busiest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.end.highload.time	Specifies the start time of your system's slowest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.lock.socket.addr	Specifies the port address of the process.

Table 5-3   (Cont.) RE Loader Daemon Entries

Entry	Description
batch.rel.archiveDir	Specifies the full path to the directory where a successfully processed file is archived. This is the default archive directory for all the event handlers.
batch.rel.rejectDir	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the default reject directory for all the event handlers.
batch.random.events	Specifies the name of the event to process. If you have two or more types of events, separate each with a comma, but no blank space. For example, TEL,SMS,GPRS.
event.**max.at.highload.time**	Specifies the highest number of the RE Loader threads permitted to run simultaneously for this event during the high-load time (that is, from **batch.start.highload.time** to **batch.end.highload.time**). For example, if *event*.**max.at.highload.time** is **2**, two threads are permitted to run simultaneously for this event during the high-load time.
event.**max.at.lowload.time**	Specifies the highest number of the RE Loader threads permitted to run simultaneously for this event during the low-load time (that is, from **batch.end.highload.time** to **batch.start.highload.time**). For example, if *event*.**max.at.lowload.time** is **2**, two threads are permitted to run simultaneously for this event during the low-load time.
event.**file.location**	Specifies the full path name of the directory to monitor for the arrival of new files that match the pattern in *event*.**file.pattern**.
event.**file.pattern**	Specifies the file name pattern to look for. You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
event.**archiveDir**	(Optional) Specifies the full path to the directory where a successfully processed file is archived for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that archives files from a particular event handler.
event.**rejectDir**	(Optional) Specifies the full path to the directory where an unsuccessfully processed file is stored for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that stores files from a particular event handler.
event.**file.type**	Specifies the type of input CDR file. • **ECE_PRE_SPLIT** specifies that RE Loader uses ECE generated preprocessed control files and data files. • **STANDARD** specifies that RE Loader uses pipeline generated input files. The default is **STANDARD**. **Note**: You cannot specify both **ECE_PRE_SPLIT** and **STANDARD** in the same **Infranet.properties** file.

RE Loader Processing

Configure the following entries in the RE Loader **Infranet.properties** file to configure how RE Loader retrieves CDR files and loads events. Set the default configuration entries shown in Table 5-4. The configuration information in this section applies to all events except for those defined in the storable class-specific section.

Table 5-4   Default Configuration Entries

Entry	Description
infranet.rel.default.interim_directory	Specifies the RE Loader processing directory. This is the location where RE Formatter loads the CDR files. The initial value is *BRM_home***/apps/pin_rel**. The default is **./**.
infranet.rel.default.supported_creation _processes	Specifies which creation processes are supported. As initially configured, RE Loader supports all creation processes: • **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. • **SUSPENSE_CREATE** specifies that the RE Loader process creates new suspense records in the suspended usage table. • **SUSPENSE_UPDATE** specifies that the RE Loader process updates existing suspense records in the suspended usage table. By default, RE Loader supports only the **PIN_REL_TRANSFORM_CDR** process.
infranet.rel.default.failure_script	Specifies the script called when RE Loader attempts to reload events that previously failed to load into the database. The initial value and the default are **pin_rel_handle_interim_files.pl**.
infranet.rel.default.failure_flags	Specifies the flag passed to the failure script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** failure script: • **0** to do nothing. • **1** to rename the interim files by appending **.saved.timestamp** to the file name. • **2** to delete the temporary files. • **3** to move the unsuccessfully processed data files generated by ECE to the reject subdirectory. The initial value and the default are **1**.
infranet.rel.default.preprocess_script	Specifies the name of the preprocessing script. The initial value and the default are **pin_rel_preprocess_cdr.pl**.
infranet.rel.default.preprocess_flags	Specifies the flag passed to the preprocessing script. The initial value and the default are **0**.
infranet.rel.default.load_util	Specifies the name of the load utility. **Important:** If you use Oracle SQL Loader, you must use conventional-path loading. BRM does not support direct-path loading. You must also use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option. Specify the conventional-path loading mode as follows: **sqlldr direct=false** The initial value and the default are **sqlldr direct=false streamsize=5000000 readsize=10000000**.
infranet.rel.default.preupdater_sproc	Specifies the name of the preupdate stored procedure. The initial value and the default are **pin_rel.pin_rel_pre_updater_sp**.
infranet.rel.default.preupdater_batch_ size	Specifies the size of the preupdate batch. The initial value and the default are **5**.
infranet.rel.default.preupdater_flags	Specifies the flag passed to the preupdate stored procedure. The initial value and the default are **1**.

Table 5-4   (Cont.) Default Configuration Entries

Entry	Description
infranet.rel.default.updater_sproc	Specifies the name of the update stored procedure. The initial value and the default are **pin_rel.pin_rel_updater_sp**.
infranet.rel.default.updater_batch_size	Specifies the size of the update batch. The initial value and the default are **5**.
infranet.rel.default.updater_flags	Specifies the flag passed to the update stored procedure. The initial value and the default are **1**.
infranet.rel.default.success_script	Specifies the script called when RE Loader successfully loads a batch of events into the BRM database. The initial value and the default are **pin_rel_handle_interim_files.pl**.
infranet.rel.default.success_flags	Specifies the flag passed to the success script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** script: • **0** to do nothing. • **1** to rename the interim files by appending **.saved.timestamp** to the file name. • **2** to delete the temporary files. The initial value and the default are **2**.
infranet.rel.default.storable_class	Specifies the storable class you are loading. The initial value and the default are **/event/delayed/session/gprs**. **Important**: If you use Oracle SQL Loader, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option.
infranet.rel.default.creation_process	Specifies whether the file contains prerated, rerated, or discount events: • **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. **Important:** RE Loader can dynamically source the creation process from the event record header file. Uncomment this entry only if all of your event record files come from the same creation process.
infranet.rel.default.ece_control_file_directory	Specifies the location of the control files generated by BRM Elastic Charging Engine (ECE).The default is *BRM_home*/**apps/pin_rel**.
infranet.rel.default.ece_data_file_directory	Specifies the location of the data files generated by ECE.The default is *BRM_home*/**apps/pin_rel**.
infranet.rel.ece_preprocessed	Specifies whether RE Loader uses ECE generated preprocessed control files and data files: • **TRUE** specifies that RE Loader uses ECE generated preprocessed control files and data files. • **FALSE** specifies that RE Loader uses pipeline generated input files. The default is **FALSE**.

Loading Event Records

Configure the following entries in the RE Loader **Infranet.properties** file to configure how ER Loader handles event records.

Table 5-5   RE Loader Event Handling Configuration Entries

Entry	Description
infranet.rel.use_end_time	Specifies whether RE Loader uses the start time or end time of the rated event for deciding the billing cycle. • **1** specifies that RE Loader uses the end time of the rated event for deciding the billing cycle. The initial value and the default are **1**. • **0** specifies that RE Loader uses the start time of the rated event for deciding the billing cycle.
infranet.rel.default.header.record_type	Specifies the header record type. The initial value and the default are **010**.
infranet.rel.default.detail.record_type	Specifies the detail record type. The initial value and the default are **020**.
infranet.rel.default.trailer.record_type	Specifies the trailer record type. The initial value and the default are **090**.
infranet.rel.field.delimiter	Specifies the delimiter symbol. The initial value and the default are **\t** for tabs.
infranet.rel.header.position.storable_class	Specifies which field in the event record file contains the storable class name. The initial value and the default are **20**. **Note:** When you set this field to **0**, RE Loader uses the default storable class specified in **infranet.rel.default.storable_class**.
infranet.rel.header.position.creation_process	Specifies which field in the event record file contains the name of the creation process (for example, whether the file contains prerated, rerated, or discount events). The initial value and the default are **18**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.sender	Specifies which field in the event record file contains the sender. The initial value and the default are **3**.
infranet.rel.header.position.recipient	Specifies which field in the event record file contains the recipient. The initial value and the default are **4**.
infranet.rel.header.position.file_sequence	Specifies which field in the event record file contains the file sequence number. The initial value and the default are **5**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.creation_timestamp	Specifies which field in the event record file contains the creation timestamp. The initial value and the default are **7**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.trailer.position.record_count	Specifies the field position of the field that contains the total number of detail records in the output file. The initial value and the default are **7**. The field position starts with 1.
infranet.rel.file_extension.disc.transform_script	By default, this entry is commented out. The initial value is **pin_rel_transform_cdr.pl**. No default exists.
infranet.rel.file_extension.disc.transform_flags	By default, this entry is commented out. The initial value is **0**. No default exists.

Storable Class Configuration

To override the default settings for the specified storable class, set the storable class-specific **Infranet.properties** entries shown in Table 5-6.

For each storable class, only the **infranet.rel.storable_class.***classname***.number_of_tables** and **infranet.rel.storable_class.***classname***.table.***N***.name** entries are mandatory. RE Loader uses the default settings for any undefined storable class-specific entries.

When editing these entries:

- Create a set of entries for each event you want to load.

- Replace *classname* with the appropriate storable class name. For example, use **event_delayed_session_gprs** for the **/event/delayed/session/gprs** storable class.

- Create a set of ***.table.***N***.*** entries for each table. For example, if the storable class contains three tables, create a set of ***.table.1.*** entries, a set of ***.table.2.*** entries, and a set of ***.table.3.*** entries.

Table 5-6   Storable Class-Specific Configuration Entries

Entry	Description
infranet.rel.storable_class.*classname***.interim_directory**	RE Loader processing directory. This is the location where preprocessed events are temporarily stored before they are loaded.
infranet.rel.storable_class.*classname***.supported_creation_processes**	Specifies whether the file contains prerated, rerated, or discount events.
infranet.rel.storable_class.*classname***.failure_script**	Specifies the script to call when RE Loader attempts to load events that previously failed to load into the database.
infranet.rel.storable_class.*classname***.failure_flags**	Specifies the flag to pass to the failure script.
infranet.rel.storable_class.*classname***.preprocess_script**	Specifies the name of the preprocessing script.
infranet.rel.storable_class.*classname***.preprocess_flags**	Specifies the flag to pass to the preprocessing script.
infranet.rel.storable_class.*classname***.number_of_tables**	Specifies the number of tables in the storable class. **Important**: This entry is mandatory for all types of events.
infranet.rel.storable_class.*classname***.table.***N***.name**	Specifies the name of a storable class table. **Important**: This entry is mandatory for all types of events.
infranet.rel.storable_class.*classname***.table.***N***.load_util**	Specifies the name of the load utility. **Important:** If you use Oracle SQL Loader, you must use conventional-path loading. BRM does not support direct-path loading. You must also use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option. Specify the conventional-path loading mode as follows: **sqlldr direct=false**
infranet.rel.storable_class.*classname***.table.***N***.control_file**	Specifies the control file to use when loading the data file into the database.
infranet.rel.storable_class.*classname***.preupdater_sproc**	Specifies the name of the preupdater stored procedure.
infranet.rel.storable_class.*classname***.preupdater_batch_size**	Specifies the preupdater batch size.
infranet.rel.storable_class.*classname***.preupdater_flags**	Specifies the flag to pass to the preupdater stored procedure.
infranet.rel.storable_class.*classname***.updater_sproc**	Specifies the name of the updater stored procedure.
infranet.rel.storable_class.*classname***.updater_batch_size**	Specifies the updater batch size.

Table 5-6   (Cont.) Storable Class-Specific Configuration Entries

Entry	Description
infranet.rel.storable_class.classname**.updater_flags**	Specifies the flag to pass to the updater stored procedure.
infranet.rel.storable_class.classname**.success_script**	Specifies the script to call when RE Loader successfully loads a data file into the BRM database.
infranet.rel.storable_class.classname**.success_flags**	Specifies the flag to pass to the success script when RE Loader successfully loads a data file into the BRM database.

RE Loader Log Files

Configure the following entries in the RE Loader **Infranet.properties** file to configure how log files are created.

Table 5-7   RE Loader Log File Configuration Entries

Entry	Description
infranet.log.file	Specifies the name of the RE Loader log file. The initial value is **rel.pinlog**. No default exists.
infranet.log.name	Specifies the name of the application. The initial value is **REL** for RE Loader. No default exists.
infranet.log.level	Specifies the log reporting level: • **1** specifies error-level reporting. • **2** specifies warning-level reporting. • **3** specifies debug-level reporting. The initial value is **1**. The default is **String.valueOf(ErrorLog.Error)** (equivalent to **1**). See "Setting the Reporting Level for Logging Messages" in *BRM System Administrator's Guide*.
infranet.log.logallebuf	Specifies whether RE Loader automatically logs all EbufExceptions. The initial value is **true**. No default exists.

infranet.rel.custom_error_codes	Specifies the name of the custom error code file.
	The initial value is **CustomErrorCodes.properties**.
	No default exists.
	To move this file from its default location, you must create a symbolic link between the name of the file and its new location. To create this link, go to the *BRM_home*/**apps/pin_rel** directory and enter the following at the command prompt:
	`$ ln -s path_to_where_file_was_moved /`
	`CustomErrorCodes.properties ./CustomErrorCodes.properties`

6

Configuring Rated Event Loader to Run Automatically

Learn how to set up Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader) to run automatically using either Batch Controller or the RE Loader daemon.

This information applies only to the original processing method.

Topics in this chapter:

- Configuring RE Loader to Run Automatically by Using Batch Controller
- Configuring RE Loader to Run Automatically by Using the RE Loader Daemon

See also:

- About Loading Rated Events into the BRM Database
- Configuring the Rated Event Loader Infranet.properties File
- Improving Rated Event Loader Performance
- Troubleshooting Rated Event Loading

Configuring RE Loader to Run Automatically by Using Batch Controller

To run RE Loader automatically, configure Batch Controller and the RE Loader batch handler. When a pre-rated event file is available, Batch Controller automatically starts the RE Loader batch handler, which runs the RE Loader utility (**pin_rel**).

To configure RE Loader to run automatically, do the following for *each instance of RE Loader*:

- Configuring the RE Loader Batch Handler
- Configuring Batch Controller
- Specify each RE Loader batch handler and handler settings in the Batch Controller configuration file. See "Handler Identification" in *BRM System Administrator's Guide*.

Configuring the RE Loader Batch Handler

To configure the RE Loader batch handler:

1. Give the **SampleRelHandler.pl** file a unique name. You will configure Batch Controller to call the handler using this name.

2. Create the following subdirectories: An **archive** subdirectory where successfully processed files can be stored and a **reject** subdirectory where unsuccessfully processed files can be stored.

3. Open the **SampleRelHandler_config.values** file and modify the entries shown in Table 6-1.

Table 6-1   Mandatory RE Loader Batch Handler Configuration Entries

Entry	Description
$FILETYPE	Specifies the event record file-name pattern to look for. To load only specific files, change the value of this entry. The default is ***.dat.bc** (any data file processed by Batch Controller). Batch Controller runs the RE Loader batch handler for each file with a name that matches this pattern. **Tip:** You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
$HANDLER_DIR	Specifies the full path to the directory containing the RE Loader batch handler, which is this processing directory.
$pinRELDir	Specifies the full path to the directory containing the RE Loader application, which is this processing directory.
$pinREL	Specifies the full path to the batch application executable.
$STAGING	Specifies the full path to the event output directory. If you specify a directory other than the event output directory, use the Linux command that links the event output directory to the input staging directory.
$PROCESSING	Specifies the full path to the directory from which event record files are processed. The default is defined in the **$pinRELDir** environment variable. This must be the same directory specified in the following RE Loader **Infranet.properties** entries: • **infranet.rel.default.interim_directory** • **infranet.rel.storable_class.***classname***.interim_directory**
$ARCHIVE	Specifies the full path to the directory where a successfully processed file is archived. This is the **archive** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/archive**.
$REJECT	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the **reject** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/reject**.

4. Save and close the file.

Configuring Batch Controller

The RE Loader package includes Batch Controller. If your system already has Batch Controller installed, the RE Loader installer does not install another. Use the sample Batch Controller properties file (*BRM_home***/apps/pin_rel/SampleBatchControllerInfranet.properties**) to configure Batch Controller.

The default configuration for the sample Batch Controller runs RE Loader batch handler whenever a rated event record file appears in the event output directory. You can change this setting to trigger the RE Loader batch handler at specified times or based on other kinds of occurrences by editing the event entries that trigger the RE Loader batch handler. See "Setting Activity Times and Triggers" in *BRM System Administrator's Guide*.

The optimal number of RE Loader processes to configure is at least three. Depending on the size of your files and the time it takes to load events, configuring four or five processes might save time.

To configure Batch Controller:

1. Copy the *BRM_home***/apps/pin_rel/SampleBatchControllerInfranet.properties** file to your *BRM_home***/apps/batch_controller** directory and change its name to **Infranet.properties**.

2. Open the *BRM_home***/apps/batch_controller/Infranet.properties** file.

3. Edit the BRM connection parameters.

   See "Using Configuration Files" in *BRM System Administrator's Guide*.

4. Set the **relHandler.start.string** parameter to the path of the RE Loader batch handler and to the name of the handler script that you gave it when configuring the RE Loader batch handler. See step 1 in "Configuring the RE Loader Batch Handler".

   For example:

   ```
   relHandler.start.string    BRM_home/apps/pin_rel/REL_handler_name.pl
   ```

5. (Optional) To change the number of RE Loader batch handler processes you want to run, set the maximum batch handler entries.

   > **Note:**
   >
   > The number of RE Loader batch handler processes called depends on the number of event record files in the event output directory. If you change these entries, test RE Loader's performance.

   ```
   relHandler.max.at.highload.time 3
   relHandler.max.at.lowload.time 3
   ```

6. Set the **cdrFileEvent.file.location** parameter to specify the location of the event output directory:

   ```
   cdrFileEvent.file.location /export/Portal/integRate
   ```

7. Set the **cdrFileEvent.file.pattern** parameter to which files should be processed by Batch Controller:

   ```
   cdrFileEvent.file.pattern cdr*.dat
   ```

   > **Note:**
   >
   > You can use an asterisk (*) as a wildcard character to represent zero or more characters in the file name.

8. Save and close the file.

9. Stop and restart Batch Controller.

For more information about configuring Batch Controller, see "Controlling Batch Operations" in *BRM System Administrator's Guide*.

Configuring RE Loader to Run Automatically by Using the RE Loader Daemon

The RE Loader daemon detects when CDR files arrive in the input directory, and then processes them. RE Loader daemon moves the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded.

To run RE Loader automatically by using the RE Loader daemon:

1. Configure the RE Loader **Infranet.properties** file. See "Configuring the Rated Event Loader Infranet.properties File".

2. Use the **pin_ctl** utility to run the *BRM_home***/bin/start_rel_daemon** script. See "Running the pin_ctl Utility" in *BRM System Administrator's Guide*.

To stop the RE Loader daemon, run the *BRM_home***/bin/stop_rel_daemon** script.

You can also run these scripts manually.

7

Configuring Rated Event Manager

Learn how to configure the Oracle Communications Billing and Revenue Management (BRM) Rated Event Manager (RE Manager), including the Elastic Charging Engine (ECE) RE Manager plug-in for the Rated Event Formatter (RE Formatter) and the Rated Event Loader Manager (REL Manager) utility.

Topics in this chapter:

- Configuring the RE Manager and ECE RE Manager Plug-in REM.properties Files
- Configuring REL Manager
- Configuring RE Manager and REL Manager Logging

Configuring the RE Manager and ECE RE Manager Plug-in REM.properties Files

You configure the RE Manager and the ECE RE Manager Plug-in using **REM.properties** files.

The following topics are in this section:

- BRM RE Manager Configuration
- ECE RE Manager Plug-in Configuration

BRM RE Manager Configuration

You configure the RE Manager for BRM using the **REM.properties** file. This file is located in the *$BRM_HOME*/**apps/rated_event_manager** directory. It contains several types of configuration, which are described in the following sections:

- General RE Manager Configuration
- RE Manager JDBC Pool Configuration
- RE Manager Directory Processor Configuration
- RE Manager ZIP Processor Configuration
- RE Manager Retry Processor Configuration
- RE Manager Stream Processor Configuration

There is also a sample configuration file provided: Sample RE Manager REM.properties File.

General RE Manager Configuration

You configure the parameters in this section to set general processing behavior for the RE Manager.

The parameters in Table 7-1 are all preceded by **rated_event_manager.** (including the period at the end).

Table 7-1   General RE Manager Configuration Parameters

Name	Example	Description
control_file_dir	${HOME}/data/cdr_files	The directory in which the control files for the directory processors are contained.
stop_semaphore_filename	${PIN_LOG_DIR}/ rated_event_manager/ rem.stop	The file used to indicate that the RE Manager should stop. Generally, this should not be changed.
load_thread_capacity	30	This tuning parameter sets the number of parallel threads used for loading BRM tables. For example, if each event type uses 5 tables and you set this value to 15, 3 files could be loaded simultaneously for each event type. Use this to set an upper limit on the loading load.
update_thread_capacity	20	This tuning parameter sets the number of parallel threads used for running updater procedures. For example, if each file requires 2 updater threads and you set this value to 6, 3 files could be processed in parallel. Use this to set an upper limit on the updater load.
direct_path_loading	false	This tuning parameter sets whether direct-path loading is used for database inserts. The Oracle JDBC batch API permits inserts to be hinted with the /*+ APPEND_VALUES */ hint to ensure inserts are above the high-water mark for better load performance. The insert hint will be added if this parameter is set to true. When used this value applies to all tables.

RE Manager JDBC Pool Configuration

You configure the parameters in this section to set how the RE Manager connects to the BRM database.

The parameters in Table 7-2 are all preceded by **rated_event_manager.jdbc_pool.** (including the period at the end).

Table 7-2   RE Manager JDBC Pool Configuration Parameters

Name	Example	Description
connectionURL	jdbc:oracle:thin:@//$ {HOSTNAME}:1521/PINDB	The JDBC connection string for all database connections. Note that in a multischema BRM system, multiple independent RE Formatter or RE Manager processes must be running (one per schema).
user	pin	The database user name.
wallet_location	${BRM_CONF_WALLET}	The Oracle wallet containing the database password. If this parameter is present, the **password** parameter will not be considered.
wallet_entry_name	infranet.rel.password	The entry in the Oracle wallet corresponding to the database password.
password	&aes|09|...	The database password, in BRM AES/OZT encryption (for BRM) or ECE encryption (for ECE). Do not include this parameter if you are using Oracle Wallet.
validateSchema	true	Sets whether the BRM schema should be validated when the pool is created. It checks the account_t root account entry to ensure the schema matches with the expected configuration.
initialPoolSize	1	The number of database connections to establish when the application starts.

Table 7-2   (Cont.) RE Manager JDBC Pool Configuration Parameters

Name	Example	Description
minPoolSize	1	The minimum number of database connections that the pool will attempt to maintain at all times.
maxPoolSize	50	This tuning parameter sets the maximum number of database connections that the pool will create. This number should be considered in relation to the number of application loading and updating threads. If this number is too low, performance may be compromised. This parameter provides an effective limit on the concurrent work that can be submitted to the database.
connectionWaitTimeout	10	The amount of time, in seconds, that a thread will wait for a connection. If maxPoolSize is configured to handle all of the application threads concurrently, this parameter will usually not be significant. However, if you are using the maxPoolSize parameter to limit the database workload, you may need to set this parameter to a larger number to avoid transient processing failures.
validateConnectionOn Borrow	false	Determines whether a connection will be validated when it is acquired (by sending a SQL statement).
sqlForValidateConnecti on	SELECT SYSDATE FROM DUAL	The SQL statement used to validate a connection.
fastConnectionFailover Enabled	false	Defines whether the Oracle Fast Connection Failover feature is enabled. See the Oracle Database documentation for more information.
onsConfiguration	propertiesfile=/usr/ons/ ons.properties or nodes=racnode1:4200,racno de2:4200	If fastConnectionFailoverEnabled is set to true, set this to define the Oracle Notification Services configuration for Fast Connection Failover. You can either use a properties file, in which case this entry should contain the entire path to the file, or you can define the configuration directly in the contents of this parameter. See the Oracle Database documentation for more information.
maxStatements	25	The size of the statement cache for each connection.
deferredStart	false	Defines whether the pool should be instantiated without being started. This should usually be set to false, since RE Manager needs connections immediately when it starts.
jmxEnabled	true	Determines whether Java Management Extensions (JMX) is enabled for the Oracle Universal Connection Pool (UCP).
metricUpdateInterval	10	The interval (in seconds) between pool metric updates by UCP.
poolLogLevel	INFO	The JDBC logging level. Possible values include INTERNAL_ERROR, SEVERE, WARNING, INFO, CONFIG, and FINE. See the Oracle Database documentation for more information.
nonTransientErrorCode s	1,1450,12899,1722	A list of Oracle Database error codes that will be considered *non-transient* errors. A non-transient error is an error that will persistently cause loading to fail. Errors of this type will not be retried and will result in the file in which they are contained being marked as bad. In ZIP_DB mode, the status of the row will be set to 4. In ZIP_FILE and CDR mode, the suffix **.bad** will be appended to the file name. In DIRECT mode, the transaction will fail.

RE Manager Directory Processor Configuration

You configure the parameters in this section to set how the RE Manager loads CDR files (both individual files and ZIP files). Ordinarily, you need only one set of Directory Processor parameters to handle all event types. This type of parameter is indicated by **default.** in the

parameter name. You also have the option of processing files differently by event type. If you want to do this, you put the event type in the parameter name.

The parameters in Table 7-3 are all preceded by **rated_event_manager.directory_processor.default.** or **rated_event_manager.directory_processor.**_event_type_**.** (including the period at the end).

Table 7-3   RE Manager Directory Processor Configuration Parameters

Name	Example	Description
header_dir	${HOME}/data/cdr_files	The directory to monitor for header files to process.
header_expr	^BRMCDR_GPRS.*.out$	The regular expression used to select file names to process.
data_dir	${HOME}/data/cdr_files	The directory to monitor for data files to load.
frequency	1	The frequency (in seconds) to scan the header directory for files to process.
concurrent_files	2	This tuning parameter indicates the number of files that can be processed concurrently. You use it to balance database and CPU load against throughput. Start with small values and increase as required if throughput is insufficient and there is idle database capacity which may be used.
creation_process	RATING_PIPELINE	Do not change this value.
object_cache_type	2	Do not change this value.
updater_proc_name	pin_rel.pin_rel_updater_sp	The stored procedure for balance update processing. Include this parameter if you want the rated events to update balances as well as being loaded. Ordinarily, this parameter will be included.
updater_threads	0 : 1, 10000 : 2, 50000 : 3, 100000 : 4	Use this to set the thread counts per file being processed. You can use a single numeric value, or you can set values to determine the number of threads based on the size of the file being processed. To do the latter, you enter a list of comma-separated _threshold:threads_ value pairs. The example value would result in one thread for files with 1 through 9,999 events, two threads for files with 10,000 through 49,999 events, three threads for files with 50,000 through 99,999 events, and four threads for files with 100,000 or more events.
updater_flags	1	This parameter determines whether the updater is allowed to create bill items if they are not precreated. If this is set to **0**, the update of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database update transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
updater_timeout_seconds	60	The time, in seconds, that the update procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
pre_updater_proc_name	pin_rel.pin_rel_pre_updater_sp	The name of the stored procedure which is run prior to balance updates as part of the loading stage.
pre_updater_threads	1	Use this to set the thread counts per file being processed by the preupdater. The format of this parameter is the same as the format for the **updater_threads** parameter.

Table 7-3   (Cont.) RE Manager Directory Processor Configuration Parameters

Name	Example	Description
pre_updater_flags	1	This parameter determines whether the preupdater is allowed to create bill items if they are not precreated. If this is set to **0**, the preupdate of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
pre_updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database preupdate transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
pre_updater_timeout_seconds	60	The time, in seconds, that the preupdate procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
insert_batch_size	250	This tuning parameter is used to set the number of events processed in each database insert transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
insert_timeout_seconds	30	The time, in seconds, that the insert procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
commit_batch_size	250	This tuning parameter sets the number of rows in an event table insert batch. Larger values will tend to reduce the number of database round trips and improve throughput, especially if direct-loading is being used
max_rows_per_load_thread	50000	The maximum number of rows that will be inserted by a single processing thread. This permits very large files to be processed with parallel inserts which may improve throughput.
use_end_time	true	Indicates whether an event's start_time or end_time is used to determine whether to load events into delayed event tables. Use **true** for end_time and **false** for start_time.
enable_serveroutput	0	Enable procedure output for the updater to be collected in the diagnostic log. The following values are permitted: **0**: Enables diagnostic collection with the Oracle Server output buffer set to unlimited. **-1**: Disables diagnostic collection for procedures (no output will be generated even if the diagnostic logger is set to DEBUG) **1** or higher: Set an explicit size for the Oracle Server output buffer. If this is set to a value which is too small for the data logged by a given update step, the updater may fail. The output will be enabled and collected only when DEBUG level logging is enabled for the RE Manager diagnostic logger (configured as **com.oracle.brm.rated_event_manager.diagnostic** in the **log4j2.xml** file).

Table 7-3   (Cont.) RE Manager Directory Processor Configuration Parameters

Name	Example	Description
success.mode	delete	Determines what to do with files upon successful loading. The following options are available: **move**: Move all files to the value specified in **success.target_directory**. **rename**: Rename all files by appending the value of **success.postfix** to each file name. **delete**: Delete all files. **noop**: Do nothing with the files. Leave them in the processing directory.
success.target_directory	${HOME}/data/cdr_files_archives	Directory to which the successfully loaded files will be moved if **success.mode** is set to **move**.
success.postfix	.backup	The text appended to the names of successfully loaded files if **success.mode** is set to **rename**.
failure.mode	move	Determines what to do with files upon failure to load. See the description of **success.mode** for the available values.
failure.target_directory	${HOME}/data/cdr_files_reject	Directory to which the unsuccessfully loaded files will be moved if **failure.mode** is set to **move**.
failure.postfix	.error	The text appended to the names of unsuccessfully loaded files if **failure.mode** is set to **rename**.

RE Manager ZIP Processor Configuration

You configure the parameters in this section to set how the RE Manager processes ZIP files.

The parameters in Table 7-4 are all preceded by **rated_event_manager.zip_processor.zip.** (including the period at the end).

Table 7-4   RE Manager ZIP Processor Configuration Parameters

Name	Example	Description
frequency	1	The frequency (in seconds) to scan the header directory for files to process.
concurrent_files	2	This tuning parameter indicates the number of files that can be processed concurrently. You use it to balance database and CPU load against throughput. Start with small values and increase as required if throughput is insufficient and there is idle database capacity which may be used.
delete_zip_after_load	1	Determines whether ZIP files are deleted from the staging table in the BRM database after being processed. If this is set to **0**, you must purge the data manually. You can do this using the Rated Event Loader Manager **purge** command. Over time, ZIP data, if not purged, may consume a large amount of space in the database.
num_threads	1	The number of threads used for reading data to process. Generally one thread is sufficient, but you can set the value higher if there is a bottleneck reading the data from the database.
sessions_per_fetch	5	The number of records to select on each fetch for processing. The selected data will be submitted for processing and will be processed based on the value of **concurrent_files**. Once data is processed, the next fetch will be performed.

Table 7-4   (Cont.) RE Manager ZIP Processor Configuration Parameters

Name	Example	Description
creation_process	RATING_PIPELINE	Do not change this value.
object_cache_type	2	Do not change this value.
updater_proc_name	pin_rel.pin_rel_updater_sp	The stored procedure for balance update processing. Include this parameter if you want the rated events to update balances as well as being loaded. Ordinarily, this parameter will be included.
updater_threads	0 : 1, 10000 : 2, 50000 : 3, 100000 : 4	Use this to set the thread counts per file being processed. You can use a single numeric value, or you can set values to determine the number of threads based on the size of the file being processed. To do the latter, you enter a list of comma-separated *threshold:threads* value pairs. The example value would result in one thread for files with 1 through 9,999 events, two threads for files with 10,000 through 49,999 events, three threads for files with 50,000 through 99,999 events, and four threads for files with 100,000 or more events.
updater_flags	1	This parameter determines whether the updater is allowed to create bill items if they are not precreated. If this is set to **0**, the update of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database update transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
updater_timeout_seconds	60	The time, in seconds, that the update procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
pre_updater_proc_name	pin_rel.pin_rel_pre_updater_sp	The name of the stored procedure which is run prior to balance updates as part of the loading stage.
pre_updater_threads	1	Use this to set the thread counts per file being processed by the preupdater. The format of this parameter is the same as the format for the **updater_threads** parameter.
pre_updater_flags	1	This parameter determines whether the preupdater is allowed to create bill items if they are not precreated. If this is set to **0**, the preupdate of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
pre_updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database preupdate transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
pre_updater_timeout_seconds	60	The time, in seconds, that the preupdate procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
insert_batch_size	250	This tuning parameter is used to set the number of events processed in each database insert transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.

ORACLE®

Table 7-4   (Cont.) RE Manager ZIP Processor Configuration Parameters

Name	Example	Description
insert_timeout_seconds	30	The time, in seconds, that the insert procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
commit_batch_size	250	This tuning parameter sets the number of rows in an event table insert batch. Larger values will tend to reduce the number of database round trips and improve throughput, especially if direct-loading is being used
max_rows_per_load_thread	50000	The maximum number of rows that will be inserted by a single processing thread. This permits very large files to be processed with parallel inserts which may improve throughput.
use_end_time	true	Indicates whether an event's start_time or end_time is used to determine whether to load events into delayed event tables. Use **true** for end_time and **false** for start_time.
enable_serveroutput	0	Enable procedure output for the updater to be collected in the diagnostic log. The following values are permitted: **0**: Enables diagnostic collection with the Oracle Server output buffer set to unlimited. **-1**: Disables diagnostic collection for procedures (no output will be generated even if the diagnostic logger is set to DEBUG) **1** or higher: Set an explicit size for the Oracle Server output buffer. If this is set to a value which is too small for the data logged by a given update step, the updater may fail. The output will be enabled and collected only when DEBUG level logging is enabled for the RE Manager diagnostic logger (configured as **com.oracle.brm.rated_event_manager.diagnostic** in the **log4j2.xml** file).

RE Manager Retry Processor Configuration

You configure the parameters in this section to set how the RE Manager processes retries.

The parameters in Table 7-5 are all preceded by **rated_event_manager.retry_processor.retry.** (including the period at the end).

Table 7-5   RE Manager Retry Processor Configuration Parameters

Name	Example	Description
header_dir	${HOME}/data/cdr_files	The directory to monitor for header files to process.
header_expr	^BRMCDR_GPRS.*.out$	The regular expression used to select file names to process.
frequency	1	The frequency (in seconds) to scan the header directory for files to process.
concurrent_files	2	This tuning parameter indicates the number of files that can be processed concurrently. You use it to balance database and CPU load against throughput. Start with small values and increase as required if throughput is insufficient and there is idle database capacity which may be used.
num_threads	1	The number of threads used for reading data to process. Generally one thread is sufficient, but you can set the value higher if there is a bottleneck reading the data from the database.

Table 7-5   (Cont.) RE Manager Retry Processor Configuration Parameters

Name	Example	Description
sessions_per_fetch	5	The number of records to select on each fetch for processing. The selected data will be submitted for processing and will be processed based on the value of **concurrent_files**. Once data is processed, the next fetch will be performed.
creation_process	RATING_PIPELINE	Do not change this value.
object_cache_type	2	Do not change this value.
updater_proc_name	pin_rel.pin_rel_updater_sp	The stored procedure for balance update processing. Include this parameter if you want the rated events to update balances as well as being loaded. Ordinarily, this parameter will be included.
updater_threads	0 : 1, 10000 : 2, 50000 : 3, 100000 : 4	Use this to set the thread counts per file being processed. You can use a single numeric value, or you can set values to determine the number of threads based on the size of the file being processed. To do the latter, you enter a list of comma-separated *threshold:threads* value pairs. The example value would result in one thread for files with 1 through 9,999 events, two threads for files with 10,000 through 49,999 events, three threads for files with 50,000 through 99,999 events, and four threads for files with 100,000 or more events.
updater_flags	1	This parameter determines whether the updater is allowed to create bill items if they are not precreated. If this is set to **0**, the update of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database update transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
updater_timeout_seconds	60	The time, in seconds, that the update procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
pre_updater_proc_name	pin_rel.pin_rel_pre_updater_sp	The name of the stored procedure which is run prior to balance updates as part of the loading stage.
pre_updater_threads	1	Use this to set the thread counts per file being processed by the preupdater. The format of this parameter is the same as the format for the **updater_threads** parameter.
pre_updater_flags	1	This parameter determines whether the preupdater is allowed to create bill items if they are not precreated. If this is set to **0**, the preupdate of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
pre_updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database preupdate transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
pre_updater_timeout_seconds	60	The time, in seconds, that the preupdate procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.

Table 7-5   (Cont.) RE Manager Retry Processor Configuration Parameters

Name	Example	Description
insert_batch_size	250	This tuning parameter is used to set the number of events processed in each database insert transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
insert_timeout_seconds	30	The time, in seconds, that the insert procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
commit_batch_size	250	This tuning parameter sets the number of rows in an event table insert batch. Larger values will tend to reduce the number of database round trips and improve throughput, especially if direct-loading is being used
max_rows_per_load_thread	50000	The maximum number of rows that will be inserted by a single processing thread. This permits very large files to be processed with parallel inserts which may improve throughput.
use_end_time	true	Indicates whether an event's start_time or end_time is used to determine whether to load events into delayed event tables. Use **true** for end_time and **false** for start_time.
enable_serveroutput	0	Enable procedure output for the updater to be collected in the diagnostic log. The following values are permitted: **0**: Enables diagnostic collection with the Oracle Server output buffer set to unlimited. **-1**: Disables diagnostic collection for procedures (no output will be generated even if the diagnostic logger is set to DEBUG) **1** or higher: Set an explicit size for the Oracle Server output buffer. If this is set to a value which is too small for the data logged by a given update step, the updater may fail. The output will be enabled and collected only when DEBUG level logging is enabled for the RE Manager diagnostic logger (configured as **com.oracle.brm.rated_event_manager.diagnostic** in the **log4j2.xml** file).
success.mode	delete	Determines what to do with files upon successful loading. The following options are available: **move**: Move all files to the value specified in **success.target_directory**. **rename**: Rename all files by appending the value of **success.postfix** to each file name. **delete**: Delete all files. **noop**: Do nothing with the files. Leave them in the processing directory.
success.target_directory	${HOME}/data/cdr_files_archives	Directory to which the successfully loaded files will be moved if **success.mode** is set to **move**.
success.postfix	.backup	The text appended to the names of successfully loaded files if **success.mode** is set to **rename**.
failure.mode	move	Determines what to do with files upon failure to load. See the description of **success.mode** for the available values.
failure.target_directory	${HOME}/data/cdr_files_reject	Directory to which the unsuccessfully loaded files will be moved if **failure.mode** is set to **move**.

Table 7-5   (Cont.) RE Manager Retry Processor Configuration Parameters

Name	Example	Description
failure.postfix	.error	The text appended to the names of unsuccessfully loaded files if **failure.mode** is set to **rename**.

RE Manager Stream Processor Configuration

You configure the parameters in this section to set how the RE Manager processes event streaming.

When using REM REF Plug-in in CDR streaming mode, a running Apache Kafka installation is required. The Kafka topic needs to be created before using this feature. By default, the topic name is configured as **RatedEvents**.

The parameters in Table 7-6 are all preceded by **rated_event_manager.stream_processor.stream1.** (including the period at the end). For more information about the parameters in the table that are preceded by **stream_config.consumers**, see the Kafka documentation for "Consumer Configs."

Table 7-6   RE Manager Stream Processor Loading Configuration Parameters

Name	Example	Description
load_config.creation_process	RATING_PIPELINE	Do not change this value.
load_config.object_cache_type	2	Do not change this value.
load_config.updater_threads	0 : 1, 10000 : 2, 50000 : 3, 100000 : 4	Use this to set the thread counts per file being processed. You can use a single numeric value, or you can set values to determine the number of threads based on the size of the file being processed. To do the latter, you enter a list of comma-separated *threshold:threads* value pairs. The example value would result in one thread for files with 1 through 9,999 events, two threads for files with 10,000 through 49,999 events, three threads for files with 50,000 through 99,999 events, and four threads for files with 100,000 or more events.
load_config.delete_zip_after_load	1	Determines whether ZIP files are deleted from the staging table in the BRM database after being processed. If this is set to **0**, you must purge the data manually. You can do this using the Rated Event Loader Manager **purge** command. Over time, ZIP data, if not purged, may consume a large amount of space in the database.
load_config.updater_proc_name	pin_rel.pin_rel_updater_sp	The stored procedure for balance update processing. Include this parameter if you want the rated events to update balances as well as being loaded. Ordinarily, this parameter will be included.
load_config.updater_flags	1	This parameter determines whether the updater is allowed to create bill items if they are not precreated. If this is set to **0**, the update of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.

Table 7-6   (Cont.) RE Manager Stream Processor Loading Configuration Parameters

Name	Example	Description
load_config.updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database update transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
load_config.updater_timeout_seconds	60	The time, in seconds, that the update procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
load_config.pre_updater_proc_name	pin_rel.pin_rel_pre_updater_sp	The name of the stored procedure which is run prior to balance updates as part of the loading stage.
load_config.pre_updater_threads	1	Use this to set the thread counts per file being processed by the preupdater. The format of this parameter is the same as the format for the **updater_threads** parameter.
load_config.pre_updater_flags	1	This parameter determines whether the preupdater is allowed to create bill items if they are not precreated. If this is set to **0**, the preupdate of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
load_config.pre_updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database preupdate transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
load_config.pre_updater_timeout_seconds	60	The time, in seconds, that the preupdate procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
load_config.insert_batch_size	250	This tuning parameter is used to set the number of events processed in each database insert transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
load_config.insert_timeout_seconds	30	The time, in seconds, that the insert procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
load_config.commit_batch_size	250	This tuning parameter sets the number of rows in an event table insert batch. Larger values will tend to reduce the number of database round trips and improve throughput, especially if direct-loading is being used.
load_config.max_rows_per_load_thread	50000	The maximum number of rows that will be inserted by a single processing thread. This permits very large files to be processed with parallel inserts which may improve throughput.
load_config.use_end_time	true	Indicates whether an event's start_time or end_time is used to determine whether to load events into delayed event tables. Use **true** for end_time and **false** for start_time.

ORACLE

Table 7-6   (Cont.) RE Manager Stream Processor Loading Configuration Parameters

Name	Example	Description
load_config.enable_serveroutput	0	Enable procedure output for the updater to be collected in the diagnostic log. The following values are permitted: **0**: Enables diagnostic collection with the Oracle Server output buffer set to unlimited. **-1**: Disables diagnostic collection for procedures (no output will be generated even if the diagnostic logger is set to DEBUG) **1** or higher: Set an explicit size for the Oracle Server output buffer. If this is set to a value which is too small for the data logged by a given update step, the updater may fail. The output will be enabled and collected only when DEBUG level logging is enabled for the RE Manager diagnostic logger (configured as **com.oracle.brm.rated_event_manager.diagnostic** in the **log4j2.xml** file).
stream_config.poll_timeout	5000	The time, in milliseconds, spent waiting for data to read from the topic. If it is set to 0, the poll will return immediately with any data in the buffer, or will return empty if there is no data waiting in the buffer. This must not be set to a negative number.
stream_config.topic	RatedEvents	You can configure one topic for all messages using this parameter, or you can configure a topic and partition per schema using the **topic.1**, **topic.2**... **topic.**n parameters below.
stream_config.topic.1 (stream_config.topic.2...stream_config.topic.n)	RatedEvents	Topic configuration parameters used instead of the **stream_config.topic** parameter if you are configuring different topics and partitions per schema.
stream_config.consumer.bootstrap.servers	${HOSTNAME}:9092	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. This list should be in the form: `host1:port1,host2:port2,...`
stream_config.consumer.max.poll.records	1000	The maximum number of records returned in a single poll.
stream_config.consumer.client.id	REM-Stream1	A logical ID string to pass to the server, to identify the application making the request in the logs.
stream_config.consumer.group.id	REM	The unique string that identifies the consumer group this consumer belongs to. This property is required if the consumer uses either the group management functionality or the Kafka-based offset management strategy.
stream_config.consumer.enable.auto.commit	true	Specifies whether the consumer's offset will be periodically committed in the background.
stream_config.consumer.isolation_level	read_committed	Controls how to read messages written transactionally. Valid values are: • read_uncommitted: (default) All messages will be returned, even transactional messages which have been aborted. • read_committed: Only transactional messages which have been committed will be returned. Non-transactional messages will be returned unconditionally in either mode.

Table 7-6   (Cont.) RE Manager Stream Processor Loading Configuration Parameters

Name	Example	Description
stream_config.consumer.auto.commit.interval.ms	2000	The frequency in milliseconds that the consumer offsets are auto-committed to Kafka if **stream_config.consumer.enable.auto.commit** is set to **true**.
stream_config.consumer.key.deserializer	org.apache.kafka.common.serialization.StringDeserializer	The deserializer class for keys for the Kafka interface.
stream_config.consumer.value.deserializer	org.apache.kafka.common.serialization.ByteArrayDeserializer	The deserializer class for values for the Kafka interface.
stream_config.consumer.reconnect.backoff.ms	2000	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.
stream_config.consumer.reconnect.backoff.max.ms	10000	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.
stream_config.consumer.fetch.max.bytes	1048576	The maximum amount of data the server should return for a fetch request.

Sample RE Manager REM.properties File

Following is a sample of the **REM.properties** file for RE Manager on BRM.

```
#-------------------------------------------------------------------------
-#
# Rated Event Manager Configuration
#-------------------------------------------------------------------------
-#

# RE Manager General

rated_event_manager.control_file_dir = ${HOME}/data/cdr_files
rated_event_manager.stop_semaphore_filename = ${PIN_LOG_DIR}/
rated_event_manager/rem.stop
rated_event_manager.load_thread_capacity = 30
rated_event_manager.update_thread_capacity = 20
rated_event_manager.direct_path_loading = false

# RE Manager JDBC

rated_event_manager.jdbc_pool.connectionURL = jdbc:oracle:thin:@//$
{HOSTNAME}:1521/PINDB
rated_event_manager.jdbc_pool.user = pin
rated_event_manager.jdbc_pool.wallet_location = ${BRM_WALLET}
rated_event_manager.jdbc_pool.wallet_entry_name = infranet.rel.password
# rated_event_manager.jdbc_pool.password =
WzBZBgkqhkiG9w0BBQ0wTDArBgkqhkiG9w0BBQwwHgQIyLXajS0B/
bACAWQCARAwDAYIKoZIhvcNAgcFADAdBglghkgBZQMEAQIEECP6WTLyMj4F2kMyRWl2GSGqc3wiMuR
```

```
dOKQ1dhvObext
rated_event_manager.jdbc_pool.validateSchema = true
rated_event_manager.jdbc_pool.initialPoolSize = 1
rated_event_manager.jdbc_pool.minPoolSize = 1
rated_event_manager.jdbc_pool.maxPoolSize = 50
rated_event_manager.jdbc_pool.connectionWaitTimeout = 10
rated_event_manager.jdbc_pool.validateConnectionOnBorrow = false
rated_event_manager.jdbc_pool.sqlForValidateConnection = SELECT SYSDATE FROM
DUAL
rated_event_manager.jdbc_pool.fastConnectionFailoverEnabled = false
rated_event_manager.jdbc_pool.onsConfiguration =
rated_event_manager.jdbc_pool.maxStatements = 50
rated_event_manager.jdbc_pool.deferredStart = false
rated_event_manager.jdbc_pool.jmxEnabled = true
rated_event_manager.jdbc_pool.metricUpdateInterval = 10
rated_event_manager.jdbc_pool.poolLogLevel = INFO
rated_event_manager.jdbc_pool.nonTransientErrorCodes = 1,1450,12899,1722

# RE Manager Directory Processor

rated_event_manager.directory_processor.default.header_dir = ${HOME}/data/
cdr_files
rated_event_manager.directory_processor.default.header_expr = ^BRMCDR_.*.(out|
zip)$
rated_event_manager.directory_processor.default.data_dir = ${HOME}/data/
cdr_files
rated_event_manager.directory_processor.default.frequency = 5
rated_event_manager.directory_processor.default.concurrent_files = 1
rated_event_manager.directory_processor.default.creation_process =
RATING_PIPELINE
rated_event_manager.directory_processor.default.object_cache_type = 2
rated_event_manager.directory_processor.default.updater_proc_name =
pin_rel.pin_rel_updater_sp
rated_event_manager.directory_processor.default.updater_threads = 0 : 1,
10000 : 2, 50000 : 3, 100000 : 4
rated_event_manager.directory_processor.default.updater_flags = 1
rated_event_manager.directory_processor.default.updater_batch_size = 25
rated_event_manager.directory_processor.default.updater_timeout_seconds = 60
rated_event_manager.directory_processor.default.pre_updater_proc_name =
pin_rel.pin_rel_pre_updater_sp
rated_event_manager.directory_processor.default.pre_updater_threads = 0 : 1,
10000 : 2, 50000 : 3, 100000 : 4
rated_event_manager.directory_processor.default.pre_updater_flags = 1
rated_event_manager.directory_processor.default.pre_updater_batch_size = 25
rated_event_manager.directory_processor.default.pre_updater_timeout_seconds =
60
rated_event_manager.directory_processor.default.insert_batch_size = 250
rated_event_manager.directory_processor.default.insert_timeout_seconds = 30
rated_event_manager.directory_processor.default.commit_batch_size = 250
rated_event_manager.directory_processor.default.max_rows_per_load_thread =
50000
rated_event_manager.directory_processor.default.use_end_time = true
rated_event_manager.directory_processor.default.enable_serveroutput = 0
rated_event_manager.directory_processor.default.success.mode = move
rated_event_manager.directory_processor.default.success.target_directory = $
{HOME}/data/cdr_files_archives
```

```
rated_event_manager.directory_processor.default.failure.mode = rename
rated_event_manager.directory_processor.default.failure.postfix = .error

# RE Manager ZIP Processor

rated_event_manager.zip_processor.zip.frequency = 5
rated_event_manager.zip_processor.zip.concurrent_files = 2
rated_event_manager.zip_processor.zip.delete_zip_after_load = false
rated_event_manager.zip_processor.zip.num_threads = 1
rated_event_manager.zip_processor.zip.sessions_per_fetch = 5
rated_event_manager.zip_processor.zip.creation_process = RATING_PIPELINE
rated_event_manager.zip_processor.zip.object_cache_type = 2
rated_event_manager.zip_processor.zip.updater_proc_name =
pin_rel.pin_rel_updater_sp
rated_event_manager.zip_processor.zip.updater_threads = 0 : 1, 10000 : 2,
50000 : 3, 100000 : 4
rated_event_manager.zip_processor.zip.updater_flags = 1
rated_event_manager.zip_processor.zip.updater_batch_size = 25
rated_event_manager.zip_processor.zip.updater_timeout_seconds = 60
rated_event_manager.zip_processor.zip.pre_updater_proc_name =
pin_rel.pin_rel_pre_updater_sp
rated_event_manager.zip_processor.zip.pre_updater_threads = 0 : 1, 10000 : 2,
50000 : 3, 100000 : 4
rated_event_manager.zip_processor.zip.pre_updater_flags = 1
rated_event_manager.zip_processor.zip.pre_updater_batch_size = 25
rated_event_manager.zip_processor.zip.pre_updater_timeout_seconds = 60
rated_event_manager.zip_processor.zip.insert_batch_size = 250
rated_event_manager.zip_processor.zip.insert_timeout_seconds = 30
rated_event_manager.zip_processor.zip.commit_batch_size = 250
rated_event_manager.zip_processor.zip.max_rows_per_load_thread = 50000
rated_event_manager.zip_processor.zip.use_end_time = true
rated_event_manager.zip_processor.zip.enable_serveroutput = 0

# RE Manager Retry Processor

rated_event_manager.retry_processor.retry.header_dir = ${HOME}/data/cdr_files
rated_event_manager.retry_processor.retry.data_dir = ${HOME}/data/cdr_files
rated_event_manager.retry_processor.retry.frequency = 5
rated_event_manager.retry_processor.retry.concurrent_files = 1
rated_event_manager.retry_processor.retry.num_threads = 1
rated_event_manager.retry_processor.retry.sessions_per_fetch = 5
rated_event_manager.retry_processor.retry.creation_process = RATING_PIPELINE
rated_event_manager.retry_processor.retry.object_cache_type = 2
rated_event_manager.retry_processor.retry.updater_proc_name =
pin_rel.pin_rel_updater_sp
rated_event_manager.retry_processor.retry.updater_threads = 0 : 1, 10000 : 2,
50000 : 3, 100000 : 4
rated_event_manager.retry_processor.retry.updater_flags = 1
rated_event_manager.retry_processor.retry.updater_batch_size = 25
rated_event_manager.retry_processor.retry.updater_timeout_seconds = 60
rated_event_manager.retry_processor.retry.pre_updater_proc_name =
pin_rel.pin_rel_pre_updater_sp
rated_event_manager.retry_processor.retry.pre_updater_threads = 0 : 1,
10000 : 2, 50000 : 3, 100000 : 4
rated_event_manager.retry_processor.retry.pre_updater_flags = 1
rated_event_manager.retry_processor.retry.pre_updater_batch_size = 25
```

ORACLE

```
rated_event_manager.retry_processor.retry.pre_updater_timeout_seconds = 60
rated_event_manager.retry_processor.retry.insert_batch_size = 250
rated_event_manager.retry_processor.retry.insert_timeout_seconds = 30
rated_event_manager.retry_processor.retry.commit_batch_size = 250
rated_event_manager.retry_processor.retry.max_rows_per_load_thread = 50000
rated_event_manager.retry_processor.retry.use_end_time = true
rated_event_manager.retry_processor.retry.enable_serveroutput = 0
rated_event_manager.retry_processor.retry.success.mode = move
rated_event_manager.retry_processor.retry.success.target_directory = ${HOME}/
data/cdr_files_archives
rated_event_manager.retry_processor.retry.failure.mode = rename
rated_event_manager.retry_processor.retry.failure.postfix = .error


# Stream processor

#rated_event_manager.stream_processor.stream1.load_config.creation_process =
RATING_PIPELINE
#rated_event_manager.stream_processor.stream1.load_config.object_cache_type =
2
#rated_event_manager.stream_processor.stream1.load_config.updater_threads =
0 : 1, 10000 : 2, 50000 : 3, 100000 : 4
#rated_event_manager.stream_processor.stream1.load_config.delete_zip_after_loa
d = false
#rated_event_manager.stream_processor.stream1.load_config.updater_proc_name =
pin_rel.pin_rel_updater_sp
#rated_event_manager.stream_processor.stream1.load_config.updater_flags = 1
#rated_event_manager.stream_processor.stream1.load_config.updater_batch_size
= 25
#rated_event_manager.stream_processor.stream1.load_config.updater_timeout_seco
nds = 60
#rated_event_manager.stream_processor.stream1.load_config.pre_updater_proc_nam
e = pin_rel.pin_rel_pre_updater_sp
#rated_event_manager.stream_processor.stream1.load_config.pre_updater_threads
= 0 : 1, 10000 : 2, 50000 : 3, 100000 : 4
#rated_event_manager.stream_processor.stream1.load_config.pre_updater_flags =
1
#rated_event_manager.stream_processor.stream1.load_config.pre_updater_batch_si
ze = 25
#rated_event_manager.stream_processor.stream1.load_config.pre_updater_timeout_
seconds = 60
#rated_event_manager.stream_processor.stream1.load_config.insert_batch_size =
250
#rated_event_manager.stream_processor.stream1.load_config.insert_timeout_secon
ds = 30
#rated_event_manager.stream_processor.stream1.load_config.commit_batch_size =
250
#rated_event_manager.stream_processor.stream1.load_config.max_rows_per_load_th
read = 50000
#rated_event_manager.stream_processor.stream1.load_config.use_end_time = true
#rated_event_manager.stream_processor.stream1.load_config.enable_serveroutput
= 0
#rated_event_manager.stream_processor.stream1.stream_config.poll_timeout =
5000
#rated_event_manager.stream_processor.stream1.stream_config.topic =
RatedEvents
#rated_event_manager.stream_processor.stream1.stream_config.consumer.bootstrap
```

```
.servers = ${HOSTNAME}:9092
#rated_event_manager.stream_processor.stream1.stream_config.consumer.max.poll.
records = 1000
#rated_event_manager.stream_processor.stream1.stream_config.consumer.client.id
 = REM-Stream1
#rated_event_manager.stream_processor.stream1.stream_config.consumer.group.id
= REM
#rated_event_manager.stream_processor.stream1.stream_config.consumer.enable.au
to.commit = true
#rated_event_manager.stream_processor.stream1.stream_config.consumer.isolation
_level = read_committed
#rated_event_manager.stream_processor.stream1.stream_config.consumer.auto.comm
it.interval.ms = 2000
#rated_event_manager.stream_processor.stream1.stream_config.consumer.key.deser
ializer = org.apache.kafka.common.serialization.StringDeserializer
#rated_event_manager.stream_processor.stream1.stream_config.consumer.value.des
erializer = org.apache.kafka.common.serialization.ByteArrayDeserializer
#rated_event_manager.stream_processor.stream1.stream_config.consumer.reconnect
.backoff.ms = 2000
#rated_event_manager.stream_processor.stream1.stream_config.consumer.reconnect
.backoff.max.ms = 10000
#rated_event_manager.stream_processor.stream1.stream_config.consumer.fetch.max
.bytes = 1048576
```

ECE RE Manager Plug-in Configuration

You configure ECE RE Manager Plug-in using the **REM.properties** file. This file is located in the *$ECE_HOME*/**oceceserver/config** directory. It contains several types of configuration, which are described in the following sections:

- General ECE RE Manager Plug-in Configuration

- ECE RE Manager Plug-in JDBC Pool Configuration

- ECE RE Manager Plug-in Loading Configuration

- ECE RE Manager Plug-in Retry Configuration

- ECE RE Manager Plug-in Stream Processor Configuration

There is also a sample configuration file provided: Sample ECE RE Manager Plug-in REM.properties File.

General ECE RE Manager Plug-in Configuration

You configure the parameters in this section to set general processing behavior for the ECE RE Manager Plug-in.

The parameters in Table 7-7 are all preceded by **ref.** (including the period at the end).

Table 7-7   General ECE RE Manager Plug-in Configuration Parameters

Name	Example	Description
load_thread_capacity	30	This tuning parameter sets the number of parallel threads used for loading BRM tables. For example, if each event type uses 5 tables and you set this value to 15, 3 files could be loaded simultaneously for each event type. Use this to set an upper limit on the loading load.
update_thread_capacity	20	This tuning parameter sets the number of parallel threads used for running updater procedures. For example, if each file requires 2 updater threads and you set this value to 6, 3 files could be processed in parallel. Use this to set an upper limit on the updater load.
concurrent_updaters	2	This tuning parameter sets the number of threads used for asynchronous update processing. If this is set to 0, the ECE RE Manager Plug-in will not do update processing.
updater_backlog_limit	50	This tuning parameter sets the maximum number of files waiting for update that can be accumulated. Once this limit is reached, the ECE RE Manager Plug-in will discard further update requests. You will need to recover those updates by running the updater when the performance problem has been resolved.
mode	DIRECT	The processing mode to use. Available values are: **CDR**: CDR files will be generated. **DIRECT**: CDRs will be loaded directly into the BRM database. **ZIP_DB**: Data will be compressed and loaded into the BRM database. **ZIP_DB**: Data will be compressed and stored in files. **STREAM**: Data will be streamed to Apache Kafka.
direct_path_loading	false	This tuning parameter sets whether direct-path loading is used for database inserts. The Oracle JDBC batch API permits inserts to be hinted with the /*+ APPEND_VALUES */ hint to ensure inserts are above the high-water mark for better load performance. The insert hint will be added if this parameter is set to true. When used this value applies to all tables.
blob_timeout_seconds	30	The time limit for inserting zipped BLOB data into the **batch_rel_zip_file_t** table.

ECE RE Manager Plug-in JDBC Pool Configuration

You configure the parameters in this section to set how the ECE RE Manager Plug-in connects to the BRM database.

The parameters in Table 7-8 are all preceded by **ref.jdbc_pool.** (including the period at the end).

Table 7-8   ECE RE Manager Plug-in JDBC Pool Configuration Parameters

Name	Example	Description
connectionURL	jdbc:oracle:thin:@//$ {HOSTNAME}:1521/PINDB	The JDBC connection string for all database connections. Note that in a multischema BRM system, multiple independent RE Formatter or RE Manager processes must be running (one per schema).
user	pin	The database user name.

ORACLE

Table 7-8   (Cont.) ECE RE Manager Plug-in JDBC Pool Configuration Parameters

Name	Example	Description
wallet_location	${ECE_HOME}/wallet	The Oracle wallet containing the database password. If this parameter is present, the **password** parameter will not be considered.
wallet_entry_name	brm.db.pwd	The entry in the Oracle wallet corresponding to the database password.
password	&aes|09|...	The database password, in BRM AES/OZT encryption (for BRM) or ECE encryption (for ECE). Do not include this parameter if you are using Oracle Wallet.
validateSchema	true	Sets whether the BRM schema should be validated when the pool is created. It checks the account_t root account entry to ensure the schema matches with the expected configuration.
initialPoolSize	1	The number of database connections to establish when the application starts.
minPoolSize	1	The minimum number of database connections that the pool will attempt to maintain at all times.
maxPoolSize	50	This tuning parameter sets the maximum number of database connections that the pool will create. This number should be considered in relation to the number of application loading and updating threads. If this number is too low, performance may be compromised. This parameter provides an effective limit on the concurrent work that can be submitted to the database.
connectionWaitTimeout	10	The amount of time, in seconds, that a thread will wait for a connection. If maxPoolSize is configured to handle all of the application threads concurrently, this parameter will usually not be significant. However, if you are using the maxPoolSize parameter to limit the database workload, you may need to set this parameter to a larger number to avoid transient processing failures.
validateConnectionOnBorrow	false	Determines whether a connection will be validated when it is acquired (by sending a SQL statement).
sqlForValidateConnection	SELECT SYSDATE FROM DUAL	The SQL statement used to validate a connection.
fastConnectionFailoverEnabled	false	Defines whether the Oracle Fast Connection Failover feature is enabled. See the Oracle Database documentation for more information.
onsConfiguration	propertiesfile=/usr/ons/ons.properties or nodes=racnode1:4200,racnode2:4200	If fastConnectionFailoverEnabled is set to true, set this to define the Oracle Notification Services configuration for Fast Connection Failover. You can either use a properties file, in which case this entry should contain the entire path to the file, or you can define the configuration directly in the contents of this parameter. See the Oracle Database documentation for more information.
maxStatements	25	The size of the statement cache for each connection.
deferredStart	false	Defines whether the pool should be instantiated without being started. This should usually be set to false, since RE Manager needs connections immediately when it starts.
jmxEnabled	true	Determines whether Java Management Extensions (JMX) is enabled for the Oracle Universal Connection Pool (UCP).
metricUpdateInterval	10	The interval (in seconds) between pool metric updates by UCP.
poolLogLevel	INFO	The JDBC logging level. Possible values include INTERNAL_ERROR, SEVERE, WARNING, INFO, CONFIG, and FINE. See the Oracle Database documentation for more information.

ORACLE

Table 7-8   (Cont.) ECE RE Manager Plug-in JDBC Pool Configuration Parameters

Name	Example	Description
nonTransientErrorCodes	1,1450,12899,1722	A list of Oracle Database error codes that will be considered *non-transient* errors. A non-transient error is an error that will persistently cause loading to fail. Errors of this type will not be retried and will result in the file in which they are contained being marked as bad. In ZIP_DB mode, the status of the row will be set to 4. In ZIP_FILE and CDR mode, the suffix **.bad** will be appended to the file name. In DIRECT mode, the transaction will fail.

ECE RE Manager Plug-in Loading Configuration

You configure the parameters in this section to set how the ECE RE Manager Plug-in loads data. Ordinarily, you need only one set of ECE RE Manager Plug-in loading parameters to handle all event types. This type of parameter is indicated by **default.** in the parameter name. You also have the option of processing files differently by event type. If you want to do this, you put the event type in the parameter name.

The parameters in Table 7-9 are all preceded by **ref.default.** or **ref.***event_type***.** (including the period at the end).

Table 7-9   ECE RE Manager Plug-in Loading Configuration Parameters

Name	Example	Description
creation_process	RATING_PIPELINE	Do not change this value.
object_cache_type	2	Do not change this value.
updater_proc_name	pin_rel.pin_rel_updater_sp	The stored procedure for balance update processing. Include this parameter if you want the rated events to update balances as well as being loaded. Ordinarily, this parameter will be included.
updater_threads	0 : 1, 10000 : 2, 50000 : 3, 100000 : 4	Use this to set the thread counts per file being processed. You can use a single numeric value, or you can set values to determine the number of threads based on the size of the file being processed. To do the latter, you enter a list of comma-separated *threshold:threads* value pairs. The example value would result in one thread for files with 1 through 9,999 events, two threads for files with 10,000 through 49,999 events, three threads for files with 50,000 through 99,999 events, and four threads for files with 100,000 or more events.
updater_flags	1	This parameter determines whether the updater is allowed to create bill items if they are not precreated. If this is set to **0**, the update of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database update transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
updater_timeout_seconds	60	The time, in seconds, that the update procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.

Table 7-9   (Cont.) ECE RE Manager Plug-in Loading Configuration Parameters

Name	Example	Description
pre_updater_proc_name	pin_rel.pin_rel_pre_updater_sp	The name of the stored procedure which is run prior to balance updates as part of the loading stage.
pre_updater_threads	1	Use this to set the thread counts per file being processed by the preupdater. The format of this parameter is the same as the format for the **updater_threads** parameter.
pre_updater_flags	1	This parameter determines whether the preupdater is allowed to create bill items if they are not precreated. If this is set to **0**, the preupdate of an item will fail if the bill item is not already created. If this is set to **1**, RE Manager will create the bill item if it does not already exist.
pre_updater_batch_size	25	This tuning parameter is used to set the number of events processed in each database preupdate transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
pre_updater_timeout_seconds	60	The time, in seconds, that the preupdate procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
insert_batch_size	250	This tuning parameter is used to set the number of events processed in each database insert transaction. A higher value reduces overhead, but may also create additional contention with other BRM applications, such as pin_bill_accts. If contention is observed (balance group locking), then consider reducing this value.
insert_timeout_seconds	30	The time, in seconds, that the insert procedure may take to run. If this time is reached, the updater will be interrupted and a rollback will be performed.
commit_batch_size	250	This tuning parameter sets the number of rows in an event table insert batch. Larger values will tend to reduce the number of database round trips and improve throughput, especially if direct-loading is being used
max_rows_per_load_thread	50000	The maximum number of rows that will be inserted by a single processing thread. This permits very large files to be processed with parallel inserts which may improve throughput.
use_end_time	true	Indicates whether an event's start_time or end_time is used to determine whether to load events into delayed event tables. Use **true** for end_time and **false** for start_time.
enable_serveroutput	0	Enable procedure output for the updater to be collected in the diagnostic log. The following values are permitted: **0**: Enables diagnostic collection with the Oracle Server output buffer set to unlimited. **-1**: Disables diagnostic collection for procedures (no output will be generated even if the diagnostic logger is set to DEBUG) **1** or higher: Set an explicit size for the Oracle Server output buffer. If this is set to a value which is too small for the data logged by a given update step, the updater may fail. The output will be enabled and collected only when DEBUG level logging is enabled for the RE Manager diagnostic logger (configured as **com.oracle.brm.rated_event_manager.diagnostic** in the **log4j2.xml** file).

Table 7-9   (Cont.) ECE RE Manager Plug-in Loading Configuration Parameters

Name	Example	Description
success.mode	delete	Determines what to do with files upon successful loading. The following options are available: **move**: Move all files to the value specified in **success.target_directory**. **rename**: Rename all files by appending the value of **success.postfix** to each file name. **delete**: Delete all files. **noop**: Do nothing with the files. Leave them in the processing directory.
success.target_directory	${HOME}/data/cdr_files_archives	Directory to which the successfully loaded files will be moved if **success.mode** is set to **move**.
success.postfix	.backup	The text appended to the names of successfully loaded files if **success.mode** is set to **rename**.

ECE RE Manager Plug-in Retry Configuration

Retry Processing

Before configuring the ECE RE Manager Plug-in retry processing, you should understand how the retry processing works.

The retry processing in the ECE RE Manager Plug-in is designed to allow the robust handling of transient processing errors. Transient processing errors are those which result from environmental, rather than data, issues. For example, lost database connections or network connection issues can be transient errors. Non-transient errors are not retried, because retying them is not likely to have any effect.

The general retry process is that the preferred processing mode will be retried a configurable number of times, using a configurable algorithm to determine the delay between the retries. Then, you can configure another processing mode to be tried (and retried), or you can omit the next processing mode, which tells the plug-in to stop and fail. You can configure as many instances of this try-and-move-on process as there are processing modes, but once you omit the next processing mode from the configuration, retry processing will stop.

Retry Configuration

You configure the parameters in this section to set how the ECE RE Manager Plug-in handles transient errors, such as a lack of connection to the database.

The parameters in Table 7-10 are all preceded by **ref.retry.***mode***.** (including the period at the end), where *mode* is the operating mode for which you are defining the retry processing: **direct**, **zip_db**, **zip_file**, or **cdr**.

Table 7-10   ECE RE Manager Plug-in Retry Configuration Parameters

Name	Example	Description
num_retries	5	The number of retry attempts that will be performed for this state.
delay_millis	3000	The base number for the delay (in milliseconds) that will be applied between retry attempts. The exact delay will depend on the delay algorithm you choose.

Table 7-10 (Cont.) ECE RE Manager Plug-in Retry Configuration Parameters

Name	Example	Description
delay_algorithm	linear	The algorithm used to determine the exact delay for retries. Valid values are: **simple** - The value in **delay_millis** will be used between all retries. For example, if **delay_millis** is set to 3000, each retry will be attempted 3 seconds after the previous attempt. **linear** - The value in **delay_millis** is multiplied by the retry number to determine the delay for each retry. That is, delay = **delay_millis** x retry_number. For example, if **delay_millis** is set to 3000, the first retry will be attempted after 3 seconds, the second after 6 seconds, and the third after 9 seconds. **double** - The value in delay_millis is multiplied by two to the power of the retry number to determine the delay for each retry. That is, delay = **delay_millis** x 2 ^ retry_number. For example, if **delay_millis** is set to 3000, the first retry will be attempted after 6 seconds, the second after 12 seconds, and the third after 24 seconds. **exponential** - The value of delay_millis is raised to the retry number power to determine the delay for each retry. That is, delay = **delay_millis** ^ retry_number. For example, if **delay_millis** is set to 3000, the first retry will be attempted after 3 seconds, the second after 9 seconds, and the third after 21 seconds.
next_mode	zip_db	The operating mode to fall back to if the retries are exhausted without successful processing. Valid values are: **direct**, **zip_db**, **zip_file**, and **cdr**.

Retry Configuration Example

In the following configuration example, the following retry attempts will be made

1. Three retry attempts in DIRECT mode, with delays of 2, 4, and 8 seconds.

2. Three retry attempts in ZIP_DB mode, with delays of 3, 6, and 9 seconds.

3. Three retry attempts in ZIP_FILE mode, with delays of 4, 8, and 16 seconds.

```
# DIRECT mode falls back to ZIP_DB mode ...
ref.retry.direct.num_retries = 3
ref.retry.direct.delay_millis = 2000
ref.retry.direct.delay_algorithm = exponential
ref.retry.direct.next_mode = zip_db
# ZIP_DB mode falls back to ZIP_FILE mode ...
ref.retry.zip_db.num_retries = 3
ref.retry.zip_db.delay_millis = 3000
ref.retry.zip_db.delay_algorithm = linear
ref.retry.zip_db.next_mode = zip_file
# ZIP_FILE mode falls back to FAILURE ...
ref.retry.zip_file.num_retries = 3
ref.retry.zip_file.delay_millis = 2000
ref.retry.zip_file.delay_algorithm = double
ref.retry.zip_file.next_mode =
```

ORACLE

ECE RE Manager Plug-in Stream Processor Configuration

You configure the parameters in this section to set how the RE Manager processes event streaming to Kafka.

When using REM REF Plug-in in CDR streaming mode, a running Apache Kafka installation is required. The Kafka topic needs to be created before using this feature. By default, the topic name is configured as **RatedEvents**.

The parameters in Table 7-11 are all preceded by **ref.event_stream.** (including the period at the end). For more information about the parameters in the table that are preceded by **producers**, see the Kafka documentation for Producer Configs.

Table 7-11   ECE RE Manager Plug-in Stream Processor Configuration Parameters

Name	Example	Description
batch_size	1	The batch size. A value of 0 indicates that the size is unlimited, and a positive integer indicates the number of events per message. **Note:** If using batching, you should enable **compression** and limit the batch size to ensure you do not exceed the message broker's limits.
compression	GZIP	Indicates whether and how messages are compressed. Valid values are: **NONE**, **ZIP**, and **GZIP**. This compression is in addition to any compression that is already being used for the message format, that is, if you set this to **GZIP** and the messages are using **ZIPFILE** format, you will have a GZIP of ZIPFILEs.
format	ZIPFILE	The format used to represent rated event records. Valid values are: • **JSONFILE**: JSON representation of the BRM **/event** objects • **ZIPFILE**: a compressed version of the actual CDR files • **CUSTOM**: a custom class with another implementation
custom.class	com.oracle.brm.ref_utils.EventToCustomPayloadConverterSample	The name of the class to use to format the events if the **format** parameter is set to **CUSTOM**.
custom.csv	true	If you set the **format** parameter to **CUSTOM** and set this parameter to **true**, it will generate the rated event data in CSV format, which can then be sent to Kafka.
schema	NONE	The schema used for the message format. Valid values are: **NONE** and **CLOUDEVENTS**.
topic	RatedEvents	You can configure one topic for all messages using this parameter, or you can configure a topic and partition per schema using the **topic.1**, **topic.2**... **topic.**n parameters below.
topic.1 (topic.2...topic.n)	RatedEvents-1	Topic configuration parameters used instead of the **topic** parameter if you are configuring different topics and partitions per schema.

Table 7-11   (Cont.) ECE RE Manager Plug-in Stream Processor Configuration Parameters

Name	Example	Description
producer.bootstrap.servers	${HOSTNAME}:9092	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. This list should be in the form: `host1:port1,host2:port2,...`
producer.enable.idempotence	true	When set to **true**, the producer will ensure that exactly one copy of each message is written in the stream. If it is set to **false**, producer retries due to broker failures and so on, may write duplicates of the retried message in the stream. Note that enabling idempotence requires **producer.max.in.flight.requests.per.connection** to be less than or equal to **5**, **producer.retries** to be greater than **0**, and **producer.acks** to be set to **all**.
producer.acks	all	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Valid values are: • **0**: The producer will not wait for any acknowledgment from the server. • **1**: The leader will write the record to its local log but will respond without waiting for acknowledgments from all followers. If the leader fails immediately after acknowledging the record but before the followers have replicated it then the record will be lost. • **all**: The leader will wait for the full set of in-sync replicas to acknowledge the record.
producer.retries	3	The number of times to resend any record that fails with a potentially transient error.
producer.max.in.flight.requests.per.connection	1	The maximum number of unacknowledged requests the client will send on a single connection before blocking. If **producer.enable.idempotence** is set to **true**, this value must be less than or equal to **5**.
producer.client.id	REF-KAFKA	A logical ID string to pass to the server, to identify the application making the request in the logs.
producer.key.serializer	org.apache.kafka.common.serialization.StringSerializer	The serializer class for keys for the Kafka interface.
producer.value.serializer	org.apache.kafka.common.serialization.ByteArraySerializer	The serializer class for values for the Kafka interface.
producer.reconnect.backoff.ms	2000	The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.
producer.reconnect.backoff.max.ms	10000	The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.
producer.max.request.size	1048576	The maximum size of a request in bytes.
schemas.cloudevents.config.specversion	1.0	The CloudEvents specification version.

Table 7-11   (Cont.) ECE RE Manager Plug-in Stream Processor Configuration Parameters

Name	Example	Description
schemas.cloudevents.config.type	com.oracle.brm.event	The CloudEvents configuration type.
schemas.cloudevents.config.source	com.oracle.ece	The CloudEvents configuration source.
schemas.cloudevents.config.extensions.extension1 (schemas.cloudevents.config.extensions.extension2... schemas.cloudevents.config.extensions.extension*n*)	extension-1-value	The CloudEvents configuration extensions.
schemas.cloudevents.headers.h1	CE1	The CloudEvents first header.
schemas.cloudevents.headers.h2	CE2	The CloudEvents second header.
schemas.none.config.dummy	dummy	The dummy parameter to use if **schema** is set to **NONE**.
schemas.none.headers.h1	N1	The first header to use if **schema** is set to **NONE**.
schemas.none.headers.h2	N2	The second header to use if **schema** is set to **NONE**.

Sample ECE RE Manager Plug-in REM.properties File

Following is a sample of the **REM.properties** file for ECE RE Manager Plug-in on ECE.

```
#----------------------------------------------------------------------------#
# REF Plugin Configuration
#----------------------------------------------------------------------------#

# ECE RE Manager Plug-in General Configuration

ref.load_thread_capacity = 30
ref.update_thread_capacity = 20
ref.concurrent_updaters = 4
ref.updater_backlog_limit = 50
ref.mode = DIRECT
ref.direct_path_loading = false
ref.blob_timeout_seconds = 30

# ECE RE Manager Plug-in JDBC Configuration

ref.jdbc_pool.connectionURL = jdbc:oracle:thin:@//${HOSTNAME}:1521/PINDB
ref.jdbc_pool.user = pin
ref.jdbc_pool.wallet_location = ${ECE_HOME}/oceceserver/wallet
ref.jdbc_pool.wallet_entry_name = brm.db.password
```

```
# ref.jdbc_pool.password =
WzBZBgkqhkiG9w0BBQ0wTDArBgkqhkiG9w0BBQwwHgQIyLXajS0B/
bACAWQCARAwDAYIKoZIhvcNAgcFADAdBglghkgBZQMEAQIEECP6WTLyMj4F2kMyRWl2GSGqc3wiMuR
dOKQ1dhvObext
ref.jdbc_pool.validateSchema = true
ref.jdbc_pool.initialPoolSize = 1
ref.jdbc_pool.minPoolSize = 1
ref.jdbc_pool.maxPoolSize = 50
ref.jdbc_pool.connectionWaitTimeout = 5
ref.jdbc_pool.validateConnectionOnBorrow = false
ref.jdbc_pool.sqlForValidateConnection = SELECT SYSDATE FROM DUAL
ref.jdbc_pool.fastConnectionFailoverEnabled = false
ref.jdbc_pool.onsConfiguration =
ref.jdbc_pool.maxStatements = 50
ref.jdbc_pool.deferredStart = false
ref.jdbc_pool.jmxEnabled = true
ref.jdbc_pool.metricUpdateInterval = 10
ref.jdbc_pool.poolLogLevel = INFO
ref.jdbc_pool.nonTransientErrorCodes = 1,1450,12899,1722

# ECE RE Manager Plug-in Loading Configuration

ref.default.creation_process = RATING_PIPELINE
ref.default.object_cache_type = 2
ref.default.updater_proc_name = pin_rel.pin_rel_updater_sp
ref.default.updater_threads = 0 : 1, 10000 : 2, 50000 : 3, 100000 : 4
ref.default.updater_flags = 1
ref.default.updater_batch_size = 25
ref.default.updater_timeout_seconds = 60
ref.default.pre_updater_proc_name = pin_rel.pin_rel_pre_updater_sp
ref.default.pre_updater_threads = 0 : 1, 10000 : 2, 50000 : 3, 100000 : 4
ref.default.pre_updater_flags = 1
ref.default.pre_updater_batch_size = 25
ref.default.pre_updater_timeout_seconds = 60
ref.default.insert_batch_size = 250
ref.default.insert_timeout_seconds = 30
ref.default.commit_batch_size = 250
ref.default.max_rows_per_load_thread = 50000
ref.default.use_end_time = true
ref.default.enable_serveroutput = 0
ref.default.success.mode = delete

# ECE RE Manager Plug-in Retry Configuration

# DIRECT mode falls back to ZIP_DB mode
ref.retry.direct.num_retries = 3
ref.retry.direct.delay_millis = 2000
ref.retry.direct.delay_algorithm = exponential
ref.retry.direct.next_mode = zip_db
# ZIP_DB mode falls back to ZIP_FILE mode
ref.retry.zip_db.num_retries = 3
ref.retry.zip_db.delay_millis = 2000
ref.retry.zip_db.delay_algorithm = linear
ref.retry.zip_db.next_mode = zip_file
# ZIP_FILE mode falls back to CDR mode
ref.retry.zip_file.num_retries = 3
```

```
ref.retry.zip_file.delay_millis = 1000
ref.retry.zip_file.delay_algorithm = exponential
ref.retry.zip_file.next_mode = cdr
# STREAM mode falls back to ABORT
ref.retry.stream.num_retries = 3
ref.retry.stream.delay_millis = 2000
ref.retry.stream.delay_algorithm = linear
ref.retry.stream.next_mode = abort

# REF Event Streaming Configuration

ref.event_stream.batch_size = 0
ref.event_stream.compression = GZIP
ref.event_stream.format = ZIPFILE
ref.event_stream.custom.class =
com.oracle.brm.ref_utils.EventToCustomPayloadConverterSample
ref.event_stream.custom.csv = true
ref.event_stream.schema = NONE
#ref.event_stream.topic.1 = RatedEvents-1
#ref.event_stream.topic.2 = RatedEvents-2
ref.event_stream.topic = RatedEvents
ref.event_stream.producer.bootstrap.servers = ${HOSTNAME}:9092
ref.event_stream.producer.enable.idempotence = true
ref.event_stream.producer.acks = all
ref.event_stream.producer.retries = 3
ref.event_stream.producer.max.in.flight.requests.per.connection = 1
ref.event_stream.producer.client.id = REF-KAFKA
ref.event_stream.producer.key.serializer =
org.apache.kafka.common.serialization.StringSerializer
ref.event_stream.producer.value.serializer =
org.apache.kafka.common.serialization.ByteArraySerializer
ref.event_stream.producer.reconnect.backoff.ms = 2000
ref.event_stream.producer.reconnect.backoff.max.ms = 10000
ref.event_stream.producer.max.request.size = 1048576
ref.event_stream.schemas.cloudevents.config.specversion = 1.0
ref.event_stream.schemas.cloudevents.config.type = com.oracle.brm.event
ref.event_stream.schemas.cloudevents.config.source = com.oracle.ece
ref.event_stream.schemas.cloudevents.config.extensions.extension1 =
extension-1-value
ref.event_stream.schemas.cloudevents.config.extensions.extension2 =
extension-2-value
ref.event_stream.schemas.cloudevents.headers.h1 = CE1
ref.event_stream.schemas.cloudevents.headers.h2 = CE2
ref.event_stream.schemas.none.config.dummy = dummy
ref.event_stream.schemas.none.headers.h1 = N1
ref.event_stream.schemas.none.headers.h2 = N2
```

Configuring REL Manager

You configure Rated Event Loader Manager (REL Manager) to define the properties needed by the REL Manager utility. See "Rated Event Loader Manager Utility" for more information about the capabilities and usage of this utility.

Configuring the Rated Event Loader Manager RELM.properties file

You configure the Rated Event Loader Manager (REL Manager) using the RELM,properties file. This file is located in the *$BRM_HOME*/**apps/rel_manager** directory.

General REL Manager Configuration

You configure the parameters in this section to set general processing behavior for the REL Manager.

The parameters in Table 7-12 are all preceded by **rel_manager.** (including the period at the end).

Table 7-12   General REL Manager Configuration Parameters

Name	Example	Description
table_format_file	${PIN_HOME}/apps/ rel_manager/ TableFormat.properties	Location of the output table format file. See "Configuring the REL Manager Table Format Properties File" for more information about this file.
console_prompt	RELManager>	The REL command prompt displayed in the REL Manager.
view.strip_directory_path	true	Indicates whether file names displayed in results in search results should include the full path (**false**) or just the file name (**true**).
zip_directory	${HOME}/data/unzipped	Directory in which to place zipped data for correction (used when exporting ZIP data which has data errors that may need correcting)
history_file	${PIN_HOME}/apps/ rel_manager/history_file	The directory in which to store the command history. If no value is provided, the command history will be stored in the **$PIN_HOME/ apps/rel_manager/.rel_manager_history** directory.

REL Manager JDBC Pool Configuration

You configure the parameters in this section to set how the REL Manager connects to the BRM database.

The parameters in Table 7-13 are all preceded by **rel_manager.jdbc_pool.** (including the period at the end).

Table 7-13   REL Manager JDBC Pool Configuration Parameters

Name	Example	Description
connectionURL	jdbc:oracle:thin:@//$ {HOSTNAME}:1521/PINDB	The JDBC connection string for all database connections. Note that in a multischema BRM system, multiple independent RE Formatter or RE Manager processes must be running (one per schema).
user	pin	The database user name.
wallet_location	${BRM_WALLET}	The Oracle wallet containing the database password.
wallet_entry_name	infranet.rel.password	The entry in the Oracle wallet corresponding to the database password.
password	&aes|09|...	The database password, in BRM AES/OZT encryption (for BRM) or ECE encryption (for ECE). Do not include this parameter if you are using Oracle Wallet.
validateSchema	true	Sets whether the BRM schema should be validated when the pool is created. It checks the account_t root account entry to ensure the schema matches with the expected configuration.

Table 7-13   (Cont.) REL Manager JDBC Pool Configuration Parameters

Name	Example	Description
initialPoolSize	1	The number of database connections to establish when the application starts.
minPoolSize	1	The minimum number of database connections that the pool will attempt to maintain at all times.
maxPoolSize	50	This tuning parameter sets the maximum number of database connections that the pool will create. This number should be considered in relation to the number of application loading and updating threads. If this number is too low, performance may be compromised. This parameter provides an effective limit on the concurrent work that can be submitted to the database.
connectionWaitTimeout	10	The amount of time, in seconds, that a thread will wait for a connection. If maxPoolSize is configured to handle all of the application threads concurrently, this parameter will usually not be significant. However, if you are using the maxPoolSize parameter to limit the database workload, you may need to set this parameter to a larger number to avoid transient processing failures.
validateConnectionOnBorrow	false	Determines whether a connection will be validated when it is acquired (by sending a SQL statement).
sqlForValidateConnection	SELECT SYSDATE FROM DUAL	The SQL statement used to validate a connection.
fastConnectionFailoverEnabled	false	Defines whether the Oracle Fast Connection Failover feature is enabled. See the Oracle Database documentation for more information.
onsConfiguration	propertiesfile=/usr/ons/ons.properties or nodes=racnode1:4200,racnode2:4200	If fastConnectionFailoverEnabled is set to true, set this to define the Oracle Notification Services configuration for Fast Connection Failover. See the Oracle Database documentation for more information.
maxStatements	25	The size of the statement cache for each connection.
deferredStart	false	Defines whether the pool should be instantiated without being started. This should usually be set to false.
jmxEnabled	true	Determines whether Java Management Extensions (JMX) is enabled for the Oracle Universal Connection Pool (UCP).
metricUpdateInterval	10	The interval (in seconds) between pool metric updates by UCP.
poolLogLevel	INFO	The JDBC logging level. Possible values include INTERNAL_ERROR, SEVERE, WARNING, INFO, CONFIG, and FINE. See the Oracle Database documentation for more information.

REL Manager Command Configuration

This section contains the parameters that define which actions will be available in REL Manager. You should not change or add to the entries in this file. However, if you want to disable any commands that are included by default, you can comment those entries out of the file. For example, if you are not using any of the ZIP processes, you might comment out the commands related to ZIP files.

The parameters in Table 7-14 are all preceded by **rel_manager.commands.** (including the period at the end).

Table 7-14   REL Manager Command Configuration Parameters

Name	Value	Description
1	com.oracle.brm.rel_manager. SearchComplete	Enables the search_complete command.
2	com.oracle.brm.rel_manager. SearchUpdateFailure	Enables the search_update_error command.
3	com.oracle.brm.rel_manager. SearchLoadFailure	Enables the search_load_error command.
4	com.oracle.brm.rel_manager. SearchPoid	Enables the search_poid command.
5	com.oracle.brm.rel_manager. RetrySession	Enables the retry_session command.
6	com.oracle.brm.rel_manager. PurgeComplete	Enables the purge command.
7	com.oracle.brm.rel_manager. SummaryReport	Enables the summary_report command.
8	com.oracle.brm.rel_manager. SearchDetailReport	Enables the search_detail command.
9	com.oracle.brm.rel_manager. ZipSummaryReport	Enables the zip_summary_report command.
10	com.oracle.brm.rel_manager. ZipDetailReport	Enables the zip_detail command.
11	com.oracle.brm.rel_manager. ZipFileWrite	Enables the zip_file_write command.
12	com.oracle.brm.rel_manager. ZipFileSearch	Enables the zip_file_search command.
13	com.oracle.brm.rel_manager. ZipFileReload	Enables the zip_file_reload command.

Sample REL Manager RELM.properties File

Following is a sample of the **RELM.properties** file for REL Manager on BRM.

```
# REL Manager Global

rel_manager.table_format_file = ${PIN_HOME}/apps/rel_manager/
TableFormat.properties
rel_manager.console_prompt = RELManager>
rel_manager.view.strip_directory_path = true
rel_manager.zip_directory = ${HOME}/data/unzipped
rel_manager.history_file = ${PIN_HOME}/apps/rel_manager/
alternative_history_file

# REL Manager JDBC

rel_manager.jdbc_pool.connectionURL = jdbc:oracle:thin:@//${HOSTNAME}:1521/
PINDB
rel_manager.jdbc_pool.user = pin
rel_manager.jdbc_pool.wallet_location = ${BRM_WALLET}
rel_manager.jdbc_pool.wallet_entry_name = infranet.rel.password
#rel_manager.jdbc_pool.password = &aes|09|
0D5E11BFDD97D2769D9B0DBFBD1BBF7E918CF0C3F5090EE918B725E18308004CDD
```

ORACLE®

```
rel_manager.jdbc_pool.validateSchema = true
rel_manager.jdbc_pool.initialPoolSize = 1
rel_manager.jdbc_pool.minPoolSize = 1
rel_manager.jdbc_pool.maxPoolSize = 2
rel_manager.jdbc_pool.connectionWaitTimeout = 5
rel_manager.jdbc_pool.validateConnectionOnBorrow = false
rel_manager.jdbc_pool.sqlForValidateConnection = SELECT SYSDATE FROM DUAL
rel_manager.jdbc_pool.fastConnectionFailoverEnabled = false
rel_manager.jdbc_pool.onsConfiguration =
rel_manager.jdbc_pool.maxStatements = 25
rel_manager.jdbc_pool.deferredStart = false
rel_manager.jdbc_pool.jmxEnabled = false
rel_manager.jdbc_pool.metricUpdateInterval = 10
rel_manager.jdbc_pool.poolLogLevel = INFO

# REL Manager Commands

rel_manager.commands.1 = com.oracle.brm.rel_manager.SearchComplete
rel_manager.commands.2 = com.oracle.brm.rel_manager.SearchUpdateFailure
rel_manager.commands.3 = com.oracle.brm.rel_manager.SearchLoadFailure
rel_manager.commands.4 = com.oracle.brm.rel_manager.SearchPoid
rel_manager.commands.5 = com.oracle.brm.rel_manager.RetrySession
rel_manager.commands.6 = com.oracle.brm.rel_manager.PurgeComplete
rel_manager.commands.7 = com.oracle.brm.rel_manager.SummaryReport
rel_manager.commands.8 = com.oracle.brm.rel_manager.SearchDetailReport
rel_manager.commands.9 = com.oracle.brm.rel_manager.ZipSummaryReport
rel_manager.commands.10 = com.oracle.brm.rel_manager.ZipDetailReport
rel_manager.commands.11 = com.oracle.brm.rel_manager.ZipFileWrite
rel_manager.commands.12 = com.oracle.brm.rel_manager.ZipFileSearch
rel_manager.commands.13 = com.oracle.brm.rel_manager.ZipFileReload
```

Configuring the REL Manager Table Format Properties File

You configure the parameters in this file to set the appearance of the output from the REL Manager search commands. The default file name and location for this file is **${PIN_HOME}/apps/rel_manager/TableFormat.properties**, but you can configure a different name and location using the **rel_manager.table_format_file** parameter in the **RELM.properties** file.

Configuring COLUMN Entries

This file consists of a block for each display table. The block contains a TABLE entry for the table and a COLUMN entry for each column. Each COLUMN entry is in the following format:

type;attributeName;textFormat;columnWidth;csvFormat;label;justification;operation

Table 7-15 describes the elements of the syntax.

Table 7-15   REL Manager COLUMN Configuration Variables

Name	Example	Description
type	BOTH	Indicates whether this column is available in a text report or in a comma-separated-value (CSV) report. Available values are **TEXT**, **CSV**, and **BOTH**. If you want to remove the column from all reports, comment out the line in the file.
attributeName	poid_id0	The name of one of the attribute names available for the table. See Table 7-16, Table 7-17, or Table 7-18 for a list of available values.
textFormat	%12d	Contains the format of the column for the text report using Java String format. This is used for text reports only.
columnWidth	14	Contains the width of the column in characters. Usually, this value will be two characters longer than the format, to allow for white space. This value is used for text reports only.
csvFormat	%d	Contains the format of the column for the CSV report using Java String format. This is used for CSV reports only.
label	Poid	Contains the heading for the column.
justification	RIGHT	Indicates the alignment for the column. Available values are **LEFT** and **RIGHT**, with **RIGHT** usually being used for numbers.
operation	none	Indicates whether the column should provide an aggregated value. Available values are: **none**: Do not use aggregation. **sum**: The sum of the values will be displayed. **avg**: The average of the values will be displayed. **min**: The minimum value will be displayed. **max**: The maximum value will be displayed.

Available Attribute Names for COLUMN Entries

There is a fixed list of attributes that can be used in the tables. The available attributes are different for different TABLE types.

Table 7-16 provides a list of the available values for the **rel_manager.batch_rel_list** and **rel_manager.batch_zip_list** tables.

Table 7-16   Attribute Name Values for the rel_manager.batch_rel_list and rel_manager.batch_zip_list Tables

Attribute Name	Description
day_str	The local date and time of the last time the object was modified
end_poid_id0	The highest POID for the batch of loaded events
end_t	The numeric end time for the load
input_file_name	The name of the file that was loaded
is_zip	YES/NO indicator of whether the object is a ZIP file
mod_t	The numeric time the object was last modified
num_total_records	The total number of events loaded in the batch
partition_number	The number of the partition set, usually 1
poid_id0	the POID for the object
recipient_name	The receiving operator (for Transferred Account Procedure (TAP) records

Table 7-16   (Cont.) Attribute Name Values for the rel_manager.batch_rel_list and rel_manager.batch_zip_list Tables

Attribute Name	Description
sender_name	The sending operator (For TAP records)
start_poid_id0	The lowest POID for the batch of loaded events
start_t	The numeric start time for the load
state	The text description of the status
status	The number indicating the status
tables	The list of BRM tables loaded for this batch
zip_size_gb	The size of the ZIP files in GB
zip_size_kb	The size of the ZIP files in KB
zip_size_mb	The size of the ZIP files in MB
zip_state	The text description of the status of the ZIP file
zip_status	The number indicating the status of the ZIP file

Table 7-17 provides a list of the available values for the **rel_manager.batch_summary** table.

Table 7-17   Attribute Name Values for the rel_manager.batch_summary Table

Attribute Name	Description
processed_str	The date and time the file was processed
row_count	The number of objects in the batch
state	The text description of the status of the batch
status	The number indicating the status of the batch
total_records	The total number of CDRs processed

Table 7-18 provides a list of the available values for the **rel_manager.zip_summary** table.

Table 7-18   Attribute Name Values for the rel_manager.zip_summary Table

Attribute Name	Description
processed_str	The date and time the file was processed
rel_state	The text description of the status of the batch
rel_status	The number indicating the status of the batch
row_count	The number of objects in the batch
total_size_gb	The size of the ZIP data in GB
total_size_kb	The size of the ZIP data in KB
total_size_mb	The size of the ZIP data in MB
total_records	The total number of CDRs processed
zip_state	The text description of the status of the ZIP file
zip_status	The number indicating the status of the ZIP file

Mapping TABLE Entries to REL Manager Commands

There are four TABLE entries in this file. Each TABLE provides the output format for query commands in REL Manager. Table 7-19 provides a list of the commands affected by each TABLE entry.

Table 7-19   TABLE and REL Manager Command Mapping

TABLE Entry	REL Manager Command Names
rel_manager.batch_rel_list	search_complete search_detail search_load_error search_poid search_update_error zip_file_search
rel_manager.batch_summary	summary_report
rel_manager.zip_summary	zip_summary_report
rel_manager.batch_zip_list	zip_detail

Following is a sample **TableFormat.properties** file.

```
TABLE=rel_manager.batch_rel_list
    COLUMN=BOTH;poid_id0;%12d;14;%d;Poid;RIGHT;none
    COLUMN=BOTH;mod_time;%-19.19s;21;%s;Mod Time;LEFT;none
    COLUMN=BOTH;input_file_name;%-60.60s;62;%s;Input Filename;LEFT;none
    COLUMN=BOTH;status;%6d;8;%d;Status;RIGHT;none
    COLUMN=BOTH;state;%-18.18s;20;%s;State;LEFT;none
    COLUMN=BOTH;num_total_records;%8d;10;%d;Records;RIGHT;SUM
    COLUMN=BOTH;start_time;%-19.19s;21;%s;Start Time;LEFT;none
    COLUMN=BOTH;end_time;%-19.19s;21;%s;End Time;LEFT;none
    COLUMN=BOTH;is_zip;%-6.6s;8;%s;Is ZIP?;LEFT;none

TABLE=rel_manager.batch_summary
    COLUMN=BOTH;processed_str;%-13.13s;15;%s;Day Processed;LEFT;NONE
    COLUMN=BOTH;status;%6d;8;%d;Status;RIGHT;NONE
    COLUMN=BOTH;state;%-18.18s;20;%s;State;LEFT;NONE
    COLUMN=BOTH;row_count;%10d;12;%d;Files;RIGHT;SUM
    COLUMN=BOTH;total_records;%10d;12;%d;Events;RIGHT;SUM

TABLE=rel_manager.zip_summary
    COLUMN=BOTH;processed_str;%-13.13s;15;%f;Day Processed;LEFT;none
    COLUMN=BOTH;zip_status;%6d;8;%d;Status;RIGHT;NONE
    COLUMN=BOTH;zip_state;%-18.18s;20;%s;State;LEFT;NONE
    COLUMN=BOTH;rel_state;%-18.18s;20;%s;Load State;LEFT;NONE
    COLUMN=BOTH;row_count;%10d;12;%d;Files;RIGHT;SUM
    COLUMN=BOTH;total_records;%10d;12;%d;Events;RIGHT;SUM
    COLUMN=BOTH;total_file_size_mb;%10.3f;12;%f;Size(Mb);RIGHT;SUM

TABLE=rel_manager.batch_zip_list
    COLUMN=BOTH;poid_id0;%12d;14;%d;Poid;RIGHT;none
    COLUMN=BOTH;mod_time;%-19.19s;21;%s;Mod Time;LEFT;none
    COLUMN=BOTH;input_file_name;%-60.60s;62;%s;Input Filename;LEFT;none
    COLUMN=BOTH;status;%6d;8;%d;Status;RIGHT;none
    COLUMN=BOTH;state;%-18.18s;20;%s;State;LEFT;none
    COLUMN=BOTH;num_total_records;%8d;10;%d;Records;RIGHT;SUM
    COLUMN=BOTH;zip_status;%10d;12;%d;Zip Status;RIGHT;none
    COLUMN=BOTH;zip_state;%-18.18s;20;%s;Zip State;LEFT;none
```

ORACLE

```
COLUMN=BOTH;zip_size_mb;%12.3f;14;%f;Zip Size(Mb);RIGHT;SUM
COLUMN=BOTH;is_extracted;%-10.10s;12;%s;Extracted?;LEFT;none
```

Configuring RE Manager and REL Manager Logging

The RE Manager (both for BRM and ECE) and the REL Manager use log4j2 for logging. Following is the location of the **log4j2.xml** configuration files for the different components:

- RE Manager: *BRM_home***/apps/rated_event_manager**
- ECE RE Manager Plug-in: *ECE_home***/oceceserver/config**
- REL Manager: *BRM_home***/apps/rel_manager**

RE Manager and ECE RE Manager Plug-in Loggers

The same loggers are defined for the RE Manager on both BRM and ECE. The RE Manager log4j2 configuration is by itself in the **log4j2.xml** file in the directory above. The ECE RE Manager Plug-in log4j2 configuration is in the main ECE **log4j2.xml** file with other ECE loggers. The loggers used in these files are:

```
<Loggers>
  <Logger name="com.oracle.brm.rated_event_manager" level="INFO"
additivity="false">
    <AppenderRef ref="REM"/>
  </Logger>
  <Logger name="com.oracle.brm.rated_event_manager.diagnostic" level="INFO"
additivity="false">
    <AppenderRef ref="REM_DIAG"/>
  </Logger>
  <Root level="INFO">
    <AppenderRef ref="REM"/>
  </Root>
</Loggers>
```

The **com.oracle.brm.rated_event_manager** logger is the main application log. It is sent to the REM appender. You can set the log level to any of the valid values for log4j2. Some of the log levels available are:

- TRACE: This log level provides verbose information, including each row loaded into the database.
- DEBUG: This log level provides information about the steps for each loading function.
- WARN: This log level provides non-critical warnings.
- INFO: This log level provides a one-line summary of each file processed.
- ERROR: This log level provides only error information.

The **com.oracle.brm.rated_event_manager.diagnostic** logger provides diagnostic data. It is sent to the REM_DIAG appender. You can set the log level to any of the valid values for log4j2. Some of the log levels available are:

- DEBUG: This diagnostic level provides information about stored procedures, if the **enable_serveroutput** parameter is set to an appropriate value in the **REM.properties** file for RE Manager.
- INFO: This diagnostic level provides data about each loader, pre-updater, and updater.

- ERROR: This diagnostic level provides error information.

REL Manager Loggers

The log configuration for REL Manager contains the following loggers:

```
<Loggers>
  <Logger name="com.oracle.brm.rated_event_manager" level="INFO"/>
  <Logger name="com.oracle.brm.rel_manager" level="INFO"/>
  <Logger name="com.oracle.brm.table_formatter" level="INFO"/>
  <Root level="INFO">
    <AppenderRef ref="RELM"/>
  </Root>
</Loggers>
```

The **com.oracle.brm.rel_manager** logger is the main application log. There is also a logger for the table formatter. The **com.oracle.brm.rated_event_manager** logger, also in use for the RE Manager, is included because the REL Manager can use the JDBC pool of the RE Manager. Log information for the REL Manager is sent to the RELM appender. You can set the log level to any of the valid values for log4j2.

8

Customizing Rated Event Loader

Learn how to customize Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader) to load events from the Pipeline Manager. This information applies only to the original processing method.

Topics in this chapter:

- Customizing RE Loader
- Adding New Types of Events for RE Loader to Load
- Creating Custom Error Codes

See also:

- About Loading Rated Events into the BRM Database
- Configuring the Rated Event Loader Infranet.properties File

Customizing RE Loader

Some of the steps required to customize RE Loader should be performed by a programmer and database administrator. To customize RE Loader, you should be familiar with the following topics:

- BRM system architecture. See "BRM System Architecture" in *BRM Concepts* and "About Loading Rated Events into the BRM Database".
- BRM storable classes. See "Understanding Storable Classes" in *BRM Developer's Guide*.
- BRM database configuration. See "Database Configuration and Tuning" in *BRM Installation Guide*.
- SQL and creating SQL control files. See your SQL documentation.

You can customize RE Loader using the following steps:

- Adding New Types of Events for RE Loader to Load
- Creating Custom Error Codes

> ✎ **Note:**
>
> *Do not modify the* **rel_updater_sp.sql** *stored procedure or any other stored procedure.* Modifying a stored procedure can corrupt data and cause maintenance and upgrade problems. Stored procedures are delivered in source code format due to database limitations and are not designed to be modified. To modify a stored procedure, you must obtain permission to do so from Oracle.

> **✎ Note:**
>
> Sub-balances are stored in events in an internal format that optimizes performance and storage efficiency. As a result, the table that stores sub-balances is not visible in the data dictionary. You should not base your customizations on this specific internal format. All sub-balance data is accessible by using the BRM API. To access this internal format, contact Oracle.

Adding New Types of Events for RE Loader to Load

When you offer a new service, you create a new storable class for the service event.

To use RE Loader to load events from a new service or new service subclass, you must create a delayed event for your new service and configure RE Loader to load it.

It is possible to load a subclass of a preconfigured service event without configuring that subclass. However, BRM will be unaware of the subclass because the subclass event is inserted into the parent class table. To track the activity of the subclass events, configure RE Loader to load the specific subclass.

You must create a new delayed event for RE Loader pre-rated events. The name of the new event storable class must start with **/event/delayed** so that BRM can distinguish it from real-time events. For example, **/event/delayed/session/***new_event_type*.

> **✎ Note:**
>
> Avoid loading prerated events by using RE Loader and another application such as an optional component.

To add an event for RE Loader to load:

1. If necessary, add the new event storable class to BRM by using Storable Class Editor. See the Storable Class Editor Help. For information about storable classes, see "About Storable Classes and Objects" in *BRM Developer's Guide*.

> **✎ Note:**
>
> If you installed GSM Manager, the **/telephony**, **/fax**, **/data**, and **/sms** subclasses of **/event/delayed/session/telco/gsm** already exist in the BRM database and do not need to be created. However, if you want to track activity specific to one of these subclasses, you must perform this entire procedure.

2. Create partitions for the event by running the **partition_utils** utility from the *BRM_home***/apps/partition_utils** directory.

   For example, the following command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

   ```
   partition_utils -o enable -t delayed -c /event/session/telco/gsm
   ```

> **✎ Note:**
>
> You must create partitions for all subclasses of a specific event that you want to load.

See "Enabling Delayed-Event Partitioning" and "partition_utils" in *BRM System Administrator's Guide*.

3. Create a control file for the new event. A control file or format file specifies the format for a single database table (array or substruct). If you added new fields to an existing array or substruct, modify the control or format file for that table. If you added a new array or substruct, create a new control or format file for the new table. For instructions on creating a control or format file, see your Oracle documentation.

4. If you created or modified any control files, modify the RE Loader preprocess script (*BRM_home***/apps/pin_rel/pin_rel_preprocess_cdr.pl**) to read the new event fields from the event record data and write the fields to the files loaded by SQL Loader. You can follow the steps used for **/event/session/telco/gsm** in the **pin_rel_preprocess_cdr.pl** file as a guide.

5. Create a new RE Loader directory corresponding to the RE Formatter output directory.

6. Add the following entries to the RE Loader **Infranet.properties** file in each directory:

   • The new event

   • A new service record type corresponding to the new event

   • The new control file that loads the new event

   • The new event tables that hold the new event

7. If you are running RE Loader automatically, you must also do the following:

   a. Configure the RE Loader batch handler in the new directory to load the new event.

   b. Add entries for the new RE Loader batch handler in the Batch Controller configuration file. See "Handler Identification" in *BRM System Administrator's Guide*.

Creating Custom Error Codes

You can create custom error codes for RE Loader scripts and utilities by using the RE Loader **CustomErrorCodes.properties** file. You use this file to list your custom error codes and messages. All entries should follow the FQEC scheme and be grouped with the correct component. See "Checking the RE Loader Log Files for Error Codes".

To create custom error codes:

1. Modify the RE Loader script or utility to report the error. For more information, see the comments in the appropriate script or utility.

2. Open the *BRM_home***/apps/pin_rel/CustomErrorCodes.properties** file in a text editor.

3. Add your custom error code to the file, making sure you use a minor code in the customer-reserved range.

   For example, the following entry creates a custom error code for the load utility:

   ```
   5100 = Sample load utility error message for a custom return code of 100.
   ```

4. Save and close the file.

9
Troubleshooting Rated Event Loading

Learn how to troubleshoot the Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader).

Topics in this chapter:

See also:

About Troubleshooting Rated Event Loading

There are two distinct error-handling actions that RE Loader takes, depending on when the error occurs:

- If an error occurs while events are being loaded, the process is canceled and all events loaded in the session are deleted from the BRM database. The SQL loader errors are logged in a file (*BRM_home*/**apps/pin_rel/***file_name*.**bad**) and a fatal error is recorded in the RE Loader log file (*Processing_directory*/**rel.pinlog**).

- If an error occurs while RE Loader is updating account balances, bill items, or journals, the loaded events are left in the database and an error is recorded in the RE Loader log file (*Processing_directory*/**rel.pinlog**). If RE Loader stops due to errors while updating account balances, bill items, or journals, correct the problem and run RE Loader again.

Some error messages are sent to standard BRM error handling. Check the **rel.pinlog** log file. See "Checking the RE Loader Log Files for Error Codes".

RE Loader checks for status in two places:

- The **/batch/rel** session status object.

  This object stores the status of the last RE Loader process. When you start RE Loader, it checks that status. If you try to reload a file that RE Loader has already successfully updated, the file is rejected because the session status indicates that the update for that file is complete.

- The REL_SUB_PROCESSES_T table.

This tables stores information about loading errors that occurred during the preupdating stage. See "Checking for Errors that Occurred during the PreUpdate Process".

Checking the RE Loader Log Files for Error Codes

RE Loader uses the SQL Loader utility, **sqlldr**, to load events into the BRM database. The **sqlldr** process creates a new log file for each input file so that log files from a previous process are not overwritten.

The log files and the temporary files created during preprocessing incorporate the name of the input file in their file names, making it easier to debug if an error occurs.

Error codes follow the fully qualified error code (FQEC) scheme, which consists of a major code that represents the component and a minor code that represents the error number. All BRM-defined errors use a minor code from 0 through 99, and all custom errors use minor codes 100 and above.

For information on how to create custom error codes for RE Loader scripts and utilities, see "Creating Custom Error Codes".

> **✎ Note:**
>
> Because modifying a stored procedure can corrupt data and cause maintenance and upgrade problems, custom error codes cannot be created for stored procedures.

The major and minor error codes for each RE Loader component are shown in Table 9-1.

Table 9-1   RE Loader Major and Minor Error Codes

Component	Description	Major Code	BRM Reserved Minor Codes	Customer Reserved Minor Codes
All	Universal code for success.	0	N/A	N/A
RE Loader driver	**pin_rel** script and Java driver code.	1000	0 - 999	N/A
Failure script	Script called when RE Loader attempts to load a data file that previously failed to load into the BRM database.	2000	0 - 99	100 - 255
Transform script	**pin_rel_transform_cdr.pl** script, which converts discount files into event record format.	3000	0 - 99	100 - 255
Preprocess script	**pin_rel_preprocess_cdr.pl** script, which preprocesses the data files and creates bulk-loadable (**.blk**) files.	4000	0 - 99	100 - 255
Load utility	**sqlldr** utility, which loads data into the BRM database.	5000	0	1 - 999
Preupdate stored procedure	Stored procedure for updating the loaded data before releasing the partition to other RE Loader sessions.	7000	0 - 99	Not available
Update stored procedure	Stored procedure for updating account balances, bill items, and journals.	8000	0 - 99	Not available

Table 9-1   (Cont.) RE Loader Major and Minor Error Codes

Component	Description	Major Code	BRM Reserved Minor Codes	Customer Reserved Minor Codes
Success script	Script that runs automatically when RE Loader successfully loads a data file into the BRM database.	9000	0 - 99	100 - 255
Database consistency check stored procedure	Stored procedure for verifying that the database indexes are correct before loading data into the database.	10000	0 - 99	Not available

Table 9-2 shows the BRM-defined error codes and messages, where *value* is the value returned in the error message:

Table 9-2   BRM-Defined Error Codes

RE Loader Error Number	Error Message
1000	REL encountered an error.
1002	The **infranet.rel.dbtype** properties value found is not supported: *value* Supported values are: *value*
1003	The **infranet.rel.partition_set_number** properties value found is not valid: *value* Valid values are between *value* and *value.*
1004	A table name properties value is missing for the given storable class: *value*
1005	A duplicate table name properties value was found: *value*
1006	The **load_util** properties value is missing for the given storable class: *value*
1007	A control file properties value is missing for the given storable class: *value*
1008	The control file name could not be found in the command line.
1009	REL cannot be executed until the Event Extraction Manager is complete.
1010	An unexpected SQL exception has occurred.
1011	An error occurred while attempting to connect to the BRM database.
1012	An error occurred while attempting to connect to the CM. Please validate the **infranet.connection** property value and ensure the CM is running.
1013	An error occurred while attempting to perform an opcode call.
1014	An interrupt has occurred and caused an error.
1015	The following file was not found: *value*
1016	An unexpected I/O error was encountered.
1017	The POID selected from the database sequence exceeds the maximum supported range of 2^{44}: *value*
1018	REL failed to select the partition name from the database.
1019	The poid_db could not be found in the input file.
1020	The poid_db found in the input file does not match the BRM database number for this CM connection. Found: *value* Expected: *value*

Table 9-2   (Cont.) BRM-Defined Error Codes

RE Loader Error Number	Error Message
1021	The header record could not be found in the input file.
1022	The storable class was not defined, or was not found in the header record.
1023	The time format found in the header record is not valid: *value*
1024	The creation process found in the header record is not supported: *value* Valid values are: *value*
1026	An invalid command-line was provided.
1027	The CM and JDBC BRM database connections are not configured to the same database schema.
1028	The REL session has timed out waiting for another REL session to complete.
1029	The file has previously completed successfully so it will not be loaded again: *value*
1030	The file is currently being processed by another REL session: *value*
1031	The *value* key is missing from the properties file.
1032	The *value* value is missing from the properties file.
1033	The configured number of tables for this storable class does not match the configured tables: *value*
1034	A number formatting error was encountered in the properties value for: *value*
1035	The **infranet.rel.updater_threads** properties value found is not valid: *value* Valid values are between *value* and *value*. To have REL auto-choose an appropriate number of threads, use the value: *value*
1036	An error occurred while attempting to parse a number for: *value*
1038	Cannot have control file with 'TRUNCATE' option when running REL in parallel loading mode between multiple REL processes.

Table 9-3 shows the BRM-defined failure script error codes.

Table 9-3   Failure Script Error Messages

Failure Script Error Number	Error Message
2000	The failure script encountered an error. The given command-line was: *value*
2001	The failure script command-line given arguments are not supported. The given command-line was: *value*
2002	The failure script command-line given flags value provided is not supported. The given command-line was: *value*
2003	The failure script command-line given directory could not be read. The given command-line was: *value*

Table 9-4 shows the BRM-defined transform script error codes.

Table 9-4   Transform Script Error Messages

Transform Script Error Number	Error Message
3000	The transform script encountered an error.
3001	The transform script command-line given arguments are not supported. The given command-line was: *value*
3002	The transform script command-line given input file could not be read. The given command-line was: *value*
3003	The transform script command-line given output file could not be created.
3004	The transform script command-line given negative discount carry over value is invalid. The given command-line was: *value*

Table 9-5 shows the BRM-defined preprocess script error codes.

Table 9-5   Preprocess Script Error Messages

Preprocess Script Error Number	Error Message
4000	The preprocess script encountered an error. The given command-line was: *value*
4001	The preprocess script command-line given arguments are not supported. The given command-line was: *value*
4002	The preprocess script failed to open a file.
4003	The preprocess script found the input file to be missing a balance record. The given command-line was: *value*
4004	The preprocess script found the input file to be missing a detail record. The given command-line was: *value*
4005	The preprocess script command-line given tables are not supported. The given command-line was: *value*
4006	The preprocess script command-line given increment_by value is not valid. The given command-line was: *value*
4007	The preprocess script did not find the expected number of records in the input file. The given command-line was: *value*
4008	The preprocess script found the input file to be missing an event record. The given command line was: *value* Used by SE Loader.
4009	The preprocess script did not find the expected size for an event record. The given command line was: *value* Used by SE Loader.
4010	The preprocess script failed to parse fields mapping data for generating the control file. The given command line was: *value* Used by SE Loader.

Table 9-6 shows the BRM-defined load utility error codes.

Table 9-6   Load Utility Error Messages

Load Utility Error Number	Error Message
5000	The database load utility encountered an error.

Table 9-7 shows the BRM-defined insert stored procedure error codes.

Table 9-7   Insert Stored Procedure Error Messages

Insert Stored Procedure Error Number	Error Message
6000	The insert stored procedure encountered an error.

Table 9-8 shows the BRM-defined preupdate stored procedure error codes.

Table 9-8   Preupdate Stored Procedure Error Messages

Preupdate Stored Procedure Error Number	Error Message
7000	The preupdate stored procedure encountered an error.
7001	The preupdate stored procedure encountered an error on a select statement.
7002	The preupdate stored procedure encountered an error on an insert statement.
7003	The preupdate stored procedure encountered an error on an update statement.
7004	The preupdate stored procedure encountered an error on a delete statement.
7008	The preupdate stored procedure encountered a parsing error.
7010	The preupdate stored procedure could not find an item for an account.
7011	The preupdate stored procedure encountered an unexpected error.

Table 9-9 shows the BRM-defined update stored procedure error codes.

Table 9-9   Update Stored Procedure Error Messages

Update Stored Procedure Error Number	Error Message
8000	The update stored procedure encountered an error.
8001	The update stored procedure encountered an error on a select statement.
8002	The update stored procedure encountered an error on an insert statement.
8003	The update stored procedure encountered an error on an update statement.
8004	The update stored procedure encountered an error on a delete statement.
8008	The update stored procedure encountered a parsing error.
8009	The update stored procedure found its record is already being processed.
8010	The update stored procedure could not find an item for an account.
8011	The update stored procedure encountered an unexpected error.
8012	The update stored procedure encountered an invalid record count error.
8013	The update stored procedure encountered an error when updating the account balances.

ORACLE

Table 9-9   (Cont.) Update Stored Procedure Error Messages

Update Stored Procedure Error Number	Error Message
8014	The update stored procedure encountered an error when updating the item balances.
8015	The update stored procedure encountered an error at TREL precommit.
8016	The update stored procedure encountered an error at TREL postcommit.

Table 9-10 shows the BRM-defined success script error codes.

Table 9-10   Success Script Error Messages

Success Script Error Number	Error Message
9000	The success script encountered an error. The given command-line was: *value*
9001	The success script command-line given arguments are not supported. The given command-line was: *value*
9002	The success script command-line given flags value provided is not supported. The given command-line was: *value*
9003	The success script command-line given directory could not be read. The given command-line was: *value*

Table 9-11 shows the BRM-defined database consistency check error codes.

Table 9-11   Database Consistency Check Error Messages

Database Consistency Check Error Number	Error Message
10000	The database consistency check encountered an error.
10005	The database consistency check found an unpartitioned index.
10006	The database consistency check found an incorrectly partitioned index.
10007	The database consistency check found an unusable index.

Checking for Errors that Occurred during the PreUpdate Process

Errors that occur during the preupdate stage of the loading process are stored in the REL_SUB_PROCESSES_T table. To check for values in the table, run SQL*Plus.

Table 9-12 shows the error codes stored in the REL_SUB_PROCESSES_T table:

Table 9-12   Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_SELECTING	−20001	An error occurred when selecting data from a table or tables.

Table 9-12   (Cont.) Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_INSERTING	−20002	An error occurred during the insert process.
ERROR_UPDATING	−20003	An error occurred during the update process.
ERROR_DELETING	−20004	An error occurred during the delete process.
ERROR_UNPARTITIONED_INDEX	−20005	An error occurred because the index is not partitioned.
ERROR_INCORRECT_PART_INDEX	−20006	An error occurred because the index is global partitioned.
ERROR_UNUSABLE_INDEX	−20007	The index partitions are unusable.
ERROR_PARSING	−20008	An error occurred during the data parsing process.
ERROR_ALREADY_BEING_PROCESSED	−20009	An error occurred because the record is being processed by another thread.
ERROR_ITEM_NOT_IN_ACCOUNT	−20010	An error occurred because the item is already billed and **pre_updater_flag** is not enabled.
ERROR_UNEXPECTED	−20011	An unexpected error occurred in the pre-update procedure.
ERROR_UPDATE_ACCT_BALANCES	−20013	An error occurred while updating account balances.
ERROR_UPDATE_ITEM_BALANCES	−20014	An error occurred while updating item balances.

Fixing Event Loading Errors

To troubleshoot event loading errors, check the RE Loader log file *BRM_home*/**apps/pin_rel/ rel.pinlog**, where *BRM_home* is the directory in which you installed BRM components. See "Checking the RE Loader Log Files for Error Codes".

At times, when RE Loader fails, the **rel.pinlog** file does not list the error. If this occurs, check the status column in the BATCH_T table in the BRM database for the status of the REL process. Table 9-13 lists the status entries (and the corresponding code attributes).

Table 9-13   Status Entries in the BATCH_T Table

Value	Decimal Value	Hex Value	Description
UPDATE_COMPLETE	0	0x0000	The load and update completed successfully. This is a terminal state: further action will not take place without intervention.
LOAD_ERROR	1	0x0001	An error was detected during the SQL Loader phase. This is a terminal state: further action will not take place without intervention.
UPDATE_ERROR	2	0x0002	An error was detected during the updater phase. This is a terminal state: further action will not take place without intervention.
PREUPDATE_ERROR	8	0x0008	An error was detected during the pre-updater phase. This is a terminal state: further action will not take place without intervention.
CREATED	9	0x0009	(ZIP_DB mode only) The row has been created, but not yet processed. The data has been loaded from REF, but not yet processed by the BRM REM.
REL_START	16	0x0010	The REL process has started - initial registration.

Table 9-13   (Cont.) Status Entries in the BATCH_T Table

Value	Decimal Value	Hex Value	Description
PRE_PROCESS	48	0x0030	REL is doing file pre-processing (using the Perl script).
START_LOAD	64	0x0040	REL is setting up SQL Loader control files/threads.
LOADING	80	0x0050	REL is waiting for SQL Loader threads to complete.
LOAD_COMPLETE	96	0x0060	Load has completed successfully (all SQL Loader processes OK).
START_PREUPDATE	1024	0x0400	REL is setting up pre-processing procedure threads.
PREUPDATING	1280	0x0500	REL is waiting for the pre-updater procedures to finish.
PREUPDATE_COMPLETE	1536	0x0600	Pre-updater phase has completed.
START_UPDATE	4096	0x1000	REL is setting up the updater threads.
UPDATING	8192	0x2000	REL is waiting for updater procedures to complete.

The correct troubleshooting effort for an event loading error depends upon the error scenario:

- RE Loader fails to start:

  The RE Loader log file (**rel.pinlog**) displays the error code **107**.

  The error occurs if REL is not running.

  Start REL using the following command:

  ```
  rel<rated event file>
  ```

- Load failure:

  The RE Loader log file (**rel.pinlog**) displays the error code **5000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **1** (see Table 9-13).

  In this error scenario, RE Loader failed either before or during the loading of the events in the event file. The events are deleted from the event tables.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the loading:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **80** (see Table 9-13).

  In this error scenario, RE Loader failed during the loading of the events and RE Loader was unable to update the session status or execute the cleanup process.

  Use the **-override** option to start a new process to reload the events. For example:

  pin_rel -override *event_file_name*

  where *event_file_name* is the event file.

- Error occurs during the pre-update stored procedure:

  The RE Loader log file (**rel.pinlog**) displays pre-update stored procedure error codes starting at **7000** and below **8000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **8** (see Table 9-13).

In this error scenario, RE Loader crashed during the execution of the pre-update stored procedure.

To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the updating of events:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **8192** (see Table 9-13).

  In this error scenario, RE Loader crashed during the updating of the events and RE Loader was unable to update the session status or execute the cleanup process.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- Error occurs during the update stored procedure:

  The RE Loader log file (**rel.pinlog**) displays update stored procedure error codes starting at **8000** and below **9000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **2**.

  In this error scenario, RE Loader successfully loaded the events but failed during the execution of the update stored procedure. The BATCH_REL_SUB_PROCESSES_T table lists the last commit, indicating the point at which the database update failed.

  Reload the events normally. The update starts from this point.

Debugging Mismatches between Data Files and Control Files

RE Loader customizations can sometimes cause data files and control files to become unsynchronized, resulting in SQL Loader failures. To help you debug these situations, use the **pin_rel_enum_blk.pl** script, which enumerates fields in your bulk-loadable files. You can then manually compare the data file entries to the control file.

To debug mismatches between your data files and control files, enter the following commands:

```
% cd BRM_home/apps/pin_rel
% pin_rel_enum_blk.pl file_name [Line_num]
```

where:

- *file_name* specifies the name of the bulk-loadable file. For example, **test2.blk**.

- *Line_num* specifies the line number of the bulk-loadable file that you want to enumerate. The default is **1**.

Retrieving Data About Events You Load

BRM stores information about events loaded by RE Loader in a **/batch/rel** object. This object contains the input file name, number of records loaded, and other session information.

> **Note:**
>
> If you use multiple database schemas, the **/batch/rel** object is created in the schema specified in the RE Loader **Infranet.properties** file.

Troubleshooting ZIP File Processing

You can configure BRM to process call detail records (CDRs) as ZIP files. The Rated Event Loader Manager (REL Manager) provides tools you can use to find, correct, and reload ZIP files that have errors. See "Rated Event Loader Manager Utility" for more information about how to use this utility.

The general process for handling ZIP files is:

1. Find the files that have errors using zip_detail.

2. Write the failed files to the file system using zip_file_write. Each ZIP file will be extracted to individual CDR files in a directory specific to the ZIP file.

3. If needed, find out what files you have extracted with **zip_file_write** using zip_file_search. You can also use **zip_detail** again to see which files have been extracted.

4. Correct the data as needed.

5. Load the modified ZIP file back into the BRM database using zip_file_reload.

6. Depending on the settings you used in **zip_file_reload**, the data is either reprocessed automatically, or you can reprocess the data using retry_session.

Table 9-14 provides the list of ZIP file error codes.

Table 9-14   ZIP File Status Codes

Code	Status Name	Description
0	COMPLETED	The ZIP file has been successfully loaded into the database.
1	INSERTING	The ZIP File is being inserted into the database; the transaction is not complete.
2	READY	The ZIP file has been committed to the database and is ready to be processed by RE Manager.
3	PROCESSING	The ZIP file is being processed by RE Manager.
4	DATA_ERROR	The ZIP file contains data which cannot be loaded into BRM. You must correct the data manually. Consult the RE Manager of RE Formatter logs for more information about the error.
5	PROCESS_ERROR	An unknown transient processing error occurred while the ZIP file data was being loaded. You can retry using REL Manager.

Process Example

The following is a simple example of the process described above.

First, you use the **zip_detail** command to see the status of the ZIP files that have been sent for processing:

```
RELManager> zip_detail 2022-06-10
+-------------+-------------------
+--------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+--------------+------------+
|        Poid | Mod Time           | Input
Filename                                         | Status | State             |
Records | Zip Status | Zip State         | Zip Size(Mb) | Extracted? |
+-------------+-------------------
+--------------------------------------------------------+--------+-------------------
```

```
+----------+-----------+-------------------+-------------+-----------+
|  112539709 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:41:02Z_2022-06-10T15:41:12Z        |      0 | COMPLETE
|        4 |          0 | COMPLETE          |       0.006 | NO         |
|  112531503 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:40:52Z_2022-06-10T15:41:02Z        |      2 | LOAD_ERROR
|       12 |          0 | COMPLETE          |       0.007 | NO         |
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
|            |
|                                                          |        |
|       16 |            |                   |       0.013 |           |
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
2 rows processed.
```

You can see that the ZIP file with the POID **112531503** has a **State** of **LOAD_ERROR**.

Next, extract the contents of the ZIP file to the file system using **zip_file_write**:

```
RELManager> zip_file_write 112531503
Successfully written ZIP data for POID '112531503'
Wrote 1 of 1 POIDs provided
Files can be found in directory: /home/brmuser/data/unzipped
```

The directory that contains the directory for the POID is displayed in the result of **zip_file_writ**e.

You can see that you have extracted the file using **zip_file_search**, which only displays extracted files:

```
RELManager> zip_file_search
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
|        Poid | Mod Time         | Input
Filename                                              | Status | State          |
Records | Start Time        | End Time           | Is ZIP?|
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
|    112531503 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:40:52Z_2022-06-10T15:41:02Z         |      1 | LOAD_ERROR
|       12 | 2022-06-10 08:41:11 | 2022-06-10 08:41:11 | YES    |
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
1 rows processed.
```

Another way you can see that the file has been extracted is using **zip_detail** again:

```
RELManager> zip_detail 2022-06-10
+-------------+--------------------
```

```
+------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
|       Poid | Mod Time          | Input
Filename                                         | Status | State              |
Records | Zip Status | Zip State        | Zip Size(Mb) | Extracted? |
+-------------+--------------------
+------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
|   112539709 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:41:02Z_2022-06-10T15:41:12Z        |      0 | COMPLETE
|       4 |          0 | COMPLETE          |       0.006 | NO          |
|   112531503 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:40:52Z_2022-06-10T15:41:02Z        |      1 | LOAD_ERROR
|      12 |          0 | COMPLETE          |       0.007 | YES         |
+-------------+--------------------
+------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
|             |        |
|                                                            |        |          |
|      16 |            |                   |       0.013 |             |          |
+-------------+--------------------
+------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+-----------+
2 rows processed.
```

You can see in the last column that the file for POID **112531503** has been extracted.

Next, go to the directory that contains the uncompressed files and correct whatever data caused the error.

When the data has been corrected, you can reload the data into the BRM database using **zip_file_reload**:

```
RELManager> zip_file_reload false 112531503
Successfully updated ZIP data for POID '112531503'
Updated 1 of 1 POIDs provided
Successfully loaded files have been renamed with suffix
'.reloaded_<timestamp>' in directory: /scratch/ri-user-1/data/unzipped
```

Because the example used "**false**," the ZIP file was reloaded but not reprocessed. To reprocess the file, you run **retry_session**:

```
RELManager> retry_session 112531503
Successfully submitted Poid[112531503]
File[BRMCDR_GSM_2022-06-10T15:40:52Z_2022-06-10T15:41:02Z] for retry
Submitted 1 sessions for retry (0 failures)
```

You can check that the file was processed successfully using **zip_detail**:

```
RELManager> zip_detail 2022-06-10
+--------------+--------------------
+------------------------------------------------------------+--------
+-------------------+----------+-----------+-------------------
+-------------+-----------+
|         Poid | Mod Time          | Input
```

```
Filename                                                | Status |
State                 | Records | Zip Status | Zip State         | Zip
Size(Mb) | Extracted? |
+-------------+--------------------
+-----------------------------------------------------------+-------
+-------------------+---------+-----------+-------------------
+-------------+------------+
|    112539709 | 2022-06-10 08:47:18 |
BRMCDR_GSM_2022-06-10T15:41:02Z_2022-06-10T15:41:12Z         |      0 |
COMPLETE            |         4 |         0 | COMPLETE          |
0.006 | NO           |
|    112531503 | 2022-06-10 09:02:58 |
BRMCDR_GSM_2022-06-10T15:40:52Z_2022-06-10T15:41:02Z         |      0 |
COMPLETE            |        12 |         5 | PROCESS_ERROR     |
0.007 | NO           |
+-------------+--------------------
+-----------------------------------------------------------+-------
+-------------------+---------+-----------+-------------------
+-------------+------------+
|             |
|                                                                   |
|                   |        16 |           |                   |
0.013 |           |
+-------------+--------------------
+-----------------------------------------------------------+-------
+-------------------+---------+-----------+-------------------
+-------------+------------+
2 rows processed.
```

The **State** of POID **112531503** is now shown to be **COMPLETE**.

10

Improving Rated Event Loader Performance

Learn how to improve the performance of the Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader).

This information applies only to the original processing method. To tune the performance of the Rated Event Manager (for the new methods), see the tuning parameters in "Configuring the RE Manager and ECE RE Manager Plug-in REM.properties Files".

Topics in this chapter:

- Improving RE Loader Performance
- Increasing the Number of Account Balance and Bill Item Updates
- Turning Off Index Verification to Improve Database Loading Performance
- Turning Off Database Verification to Improve Processing Performance
- Pruning Your RE Loader Control and Audit Tables

See also:

- About Loading Rated Events into the BRM Database
- Installing Rated Event Loader
- Configuring BRM Server Components for Rated Event Loading
- Configuring the Rated Event Loader Infranet.properties File

Improving RE Loader Performance

You can improve your RE Loader system performance by:

- Increasing the Number of Account Balance and Bill Item Updates
- Turning Off Index Verification to Improve Database Loading Performance
- Turning Off Database Verification to Improve Processing Performance
- Pruning Your RE Loader Control and Audit Tables

Increasing the Number of Account Balance and Bill Item Updates

RE Loader performance can be improved by increasing the number of account balance, bill item, and journal updates performed before committing the transaction.

You can modify the pre-update batch size and update batch size in the **Infranet.properties** file to specify how many updates to perform before committing the transaction. For example, if **updater_batch_size** is set to **5**, the stored procedure commits the transaction after every five updates. Increasing the number of updates might increase performance, but the updated account balances, bill items, and journals are not available until the transaction is committed. The default **batch_size** value is **5**.

> **Note:**
>
> Setting the **batch_size** value too high can result in deadlock. The value for best performance depends on your system configuration. You should test to find the best value for your system.

To change the **preupdater_batch_size** and **updater_batch_size** values:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.

> **Note:**
>
> If you have already set up your RE Loader processing directories, make sure you edit the **Infranet.properties** file in each directory.

2. If necessary, edit the **infranet.connection** entry to point to the correct database.

   For example:

   ```
   infranet.connection=pcp://root.0.0.0.1:password@localhost:37180/service/pcm_client
   ```

3. Specify the preupdater batch size value in the **preupdater_batch_size** entry.

   For example:

   ```
   infranet.rel.default.preupdater_batch_size = 8
   ```

4. Specify the updater batch size value in the **updater_batch_size** entry:

   ```
   infranet.rel.default.updater_batch_size = 8
   ```

5. Save and close the file.

Turning Off Index Verification to Improve Database Loading Performance

By default, RE Loader automatically verifies that your indexes are correct before loading data into the BRM database. This extra step helps you discover configuration errors when testing your system in a development environment.

In production systems, however, you should turn off index verification to improve database loading performance.

When configured to verify indexes, RE Loader performs the following before it runs the SQL Loader utility:

1. Checks whether the indexes to load are partitioned, local, and usable.

2. Performs one of the following:

   - If the indexes are incorrect, RE Loader aborts the loading process and logs which indexes encountered problems.

   - If the indexes are correct, RE Loader runs the SQL Loader utility to load events into the database.

When configured to skip verification, RE Loader automatically runs the SQL Loader utility to load events into the database. When the indexes are incorrect, SQL Loader fails and RE Loader logs only that the database load utility encountered an error.

To turn off index verification:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.

2. Set the **Infranet.rel.validate_indexes** entry to **False**:

   ```
   Infranet.rel.validate_indexes = False
   ```

3. Save and close the file.

Turning Off Database Verification to Improve Processing Performance

By default, RE Loader automatically verifies that it is loading events into the correct database schema by validating the database number in the event record file's first account object with the PCM database number. This extra step helps you discover configuration errors when testing your multischema system in a development environment.

In production systems, however, you should turn off database verification to improve RE Loader database loading performance.

To turn off database verification:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.

2. Set the **infranet.rel.validate_dbnumber** entry to **False**:

   ```
   infranet.rel.validate_dbnumber = False
   ```

3. Save and close the file.

Pruning Your RE Loader Control and Audit Tables

RE Loader control and audit tables grow indefinitely, so you should prune them periodically to increase system performance and reduce memory usage. To make pruning easier, you can use the RE Loader **purge_batch_rel_objects** stored procedure, which automatically prunes the tables for you.

To prune your control and audit tables:

1. Connect to the Oracle database with SQL*Plus:

   ```
   sqlplus system@DatabaseAlias
   Enter password: password
   ```

2. Enter the following command to run the stored procedure:

   ```
   SQL> pin_rel.purge_batch_rel_objects(int:Number)
   ```

   where *Number* specifies how many days worth of data to keep in the tables.

3. Type **exit** to exit SQL*Plus.

11

Rated Event Loader pin_rel Utility

Learn how to use the Oracle Communications Billing and Revenue Management (BRM) Rated Event Loader (RE Loader) pin_rel utility.

This information applies only to the original processing method.

The **pin_rel** utility loads batches of rated event records into the BRM database.

There are two ways to use this utility. When you initially run **pin_rel**, use the command without any options and use the file name as the only command-line parameter. You use the **override** option when the utility has not successfully completed its process and must be rerun.

The **pin_rel** utility looks for the event record file in the directory specified in the **Infranet.rel.rated_event_file** entry in the **Infranet.properties** file. Before you run the **pin_rel** utility, make sure the input event record file is in the specified directory and the **Infranet.properties** file is configured. See "Configuring the Rated Event Loader Infranet.properties File".

Location

*BRM_home***/apps/pin_rel**

Syntax

```
pin_rel [-override] event_file_name
```

Parameters

-override
This option starts a new **pin_rel** process. Use this option to restart **pin_rel** when it has abnormally stopped or you have stopped it manually.

> ✎ **Note:**
>
> Use this option only when you know there are no other RE Loader processes running.

RE Loader maintains the status of its operations. Because only one RE Loader process can load events into the same database at the same time, the status must indicate the loading process is complete before another process can start loading. If you manually stop **pin_rel**, its status may not reflect its true state. The **-override** parameter overrides the status and permits a new loading process to start, providing one is not already running.

event_file_name
The name of the event file to load, including its extension.

Results

If **pin_rel** is successful, it returns PROCESSED SUCCESSFULLY in the RE Loader log file (*BRM_home***/apps/pin_rel/rel.pinlog**).

If an error occurs during loading, this utility aborts the loading process. An error is logged in the **rel.pinlog** file, SQL loader errors are logged in a "bad" file (*BRM_home***/apps/pin_rel/** *event_file_name***.bad**), and the records loaded in this session are deleted from the database.

If an error occurs during account updating, the error is logged in the **rel.pinlog** file. Loaded records are not deleted from the database.

12

Rated Event Loader Manager Utility

Learn how to use the Rated Event Loader Manager (REL Manager) utility to control and get information about the rated events in the database.

It enables you to query the state of rated events loaded to BRM database and to resubmit any previously failed session for processing without needing to manipulate files on the BRM server. This utility can be used even if you are not using the Rated Event Manager (RE Manager), although the following commands will work or return data only if you are using the RE Manager in one of the modes that use ZIP files:

- zip_detail
- zip_file_reload
- zip_file_search
- zip_file_write
- zip_summary_report

Using the Rated Event Loader Manager Utility

Before using REL Manager, you must configure it. See "Configuring REL Manager" for information about configuring REL Manager.

To start REL Manager, enter the following command in the *BRM_HOME*/**apps/rel_manager** directory, where BRM_home is the directory in which you installed BRM components:

```
rel_manager
```

The system will respond with the following prompt:

```
RELManager>
```

Enter the REL Manager commands at the prompt.

REL Manager Commands

The following commands are available in the REL Manager.

Table 12-1   REL Manager Commands

Command	Description
help	Provides information about the available commands.
purge	Enables removing successfully completed items from the BRM database.
quit	Exits the REL Manager utility.

Table 12-1   (Cont.) REL Manager Commands

Command	Description
retry_session	Enables retrying rated events with specific Portal object IDs (POIDs).
search_complete	Provides information about all of the successful sessions completed on a specified day, month, or year.
search_detail	Provides information about sessions processed on a specified date, optionally with a specified status.
search_load_error	Provides information about all of the sessions with load errors on a specified day, month, or year.
search_poid	Provides information about rated events with specified POIDs.
search_update_error	Provides information about all of the sessions with update errors on a specified day, month, or year.
summary_report	Provides a summary report of the numbers and statuses of sessions processed on a specified day, month, or year.
zip_detail	Provides information about ZIP files that have been processed on a specified day, month, or year, optionally with a specified status.
zip_file_reload	Enables reloading of corrected data from the file system, ready for reprocessing.
zip_file_search	Provides information about ZIP file data that has been extracted to the file system.
zip_file_write	Extracts data for specified POIDs from the database to the file system, usually for correction and reloading
zip_summary_report	Provides a summary report of the numbers and statuses of ZIP file objects processed on a specified day, month, or year.

> **Note:**
>
> All dates and times in this utility refer to the date and time set on the BRM server.

help

When you use the **help** command without a command name specified, it provides a list of the available commands in REL Manager. When you use the **help** command with a command name specified, it provides the syntax for the command.

Syntax

help *[command_name]*

Parameters

command_name
The REL Manager command for which you want more information.

Results

The requested information is provided, for example:

```
RELManager> help

Available commands are:

   purge               : Purge fully completed /batch/rel records by age.
   retry_session       : Retry processing for the specified POID(s).
   search_complete     : Search for fully completed /batch/rel records.
   search_detail       : Search for specific /batch/rel records by date and status.
   search_load_error   : Search for loading errors in /batch/rel.
   search_poid         : Search for specific /batch/rel records by POID.
   search_update_error : Search for update errors in /batch/rel.
   summary_report      : Report on loading status grouped by day.
   zip_detail          : Search for specific /batch/rel records that are ZIP-managed by date
and zip-statue.
   zip_file_reload     : Reload a given Zip file from the configured zip directory - into /
batch/rel
   zip_file_search     : Search for ZIP file data that has been written to the file for
manual modification.
   zip_file_write      : Write a given Zip file to the configured zip directory
   zip_summary_report  : Report on ZIP-loaded files grouped by status.
   help                : This message
   quit                : Exit the application (q, exit are also allowed)

Found 15 commands. Please choose one.
```

purge

The **purge** command is used to remove successfully completed objects from the temporary tables in the BRM database. You can use this command periodically to remove obsolete data. It is particularly important to do this if you are loading ZIP files into the database and the **delete_zip_after_load** parameter is set to **0** in the ZIP processor section of the RE Manager **REM.properties** file. (See "RE Manager ZIP Processor Configuration" for more information about this parameter.) This data can use a significant amount of space in the database if it is not cleaned out.

Syntax

purge *YYYY-MM-DD*

Parameters

YYYY-MM-DD
All data up to and including this date will be removed.

Results

The purge command will delete all data for successfully processed records with a modified time earlier than or equal to the specified date.

quit

The **quit** command exits the REL Manager utility.

Syntax

```
quit
```

Parameters

There are no parameters for this command.

Results

You will exit the REL Manager utility.

retry_session

The **retry_session** command enables you to retry failed POIDs. You cannot use this command for POIDs that have completed successfully.

Syntax

```
retry_session POID_1, POID_2, ...
```

Parameters

POID
The POID of a failed object in the Rated Event Manager tables in the BRM Database.

Results

The retry_session command will move the objects for the specified POIDs to the retry table so that they can be retried You can use the search_poid command to see the result of the retry.

search_complete

Use the **search_complete** command to provide information about successfully completed objects for a specified day, month, or year.

Syntax

```
search_complete [YYYY[-MM[-DD]]] [limit]
```

Parameters

[YYYY[-MM[-DD]]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

If no date value is provided, information for all dates will be returned. If you do not provide a date, consider using the **limit** parameter to prevent a huge data set from being returned.

limit
Enter a number to limit the number of records returned.

Results

The requested information is provided, for example:

```
RELManager> search_complete 2021-09-09 5
+-------------+--------------------
+-------------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
|        Poid | Mod Time          | Input
Filename                                              | Status | State           |
Records | Start Time        | End Time           | Is ZIP?|
+-------------+--------------------
+-------------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
|    847818825 | 2021-09-09 09:05:37 |
BRMCDR_GSM_2021-09-09T04:02:34Z_2021-09-09T04:02:39Z          |      0 | COMPLETE
|      125 | 2021-09-09 09:05:37 | 2021-09-09 09:05:37 | NO     |
|    847810436 | 2021-09-09 09:05:37 |
BRMCDR_GSM_2021-09-09T04:02:29Z_2021-09-09T04:02:34Z          |      0 | COMPLETE
|      125 | 2021-09-09 09:05:36 | 2021-09-09 09:05:37 | NO     |
|    847827236 | 2021-09-09 09:05:38 |
BRMCDR_GSM_2021-09-09T04:02:39Z_2021-09-09T04:02:44Z          |      0 | COMPLETE
|      125 | 2021-09-09 09:05:38 | 2021-09-09 09:05:38 | NO     |
|    847852436 | 2021-09-09 09:05:38 |
BRMCDR_GSM_2021-09-09T04:02:54Z_2021-09-09T04:02:59Z          |      0 | COMPLETE
|      125 | 2021-09-09 09:05:38 | 2021-09-09 09:05:39 | NO     |
|    847835636 | 2021-09-09 09:05:38 |
BRMCDR_GSM_2021-09-09T04:02:44Z_2021-09-09T04:02:49Z          |      0 | COMPLETE
|      125 | 2021-09-09 09:05:38 | 2021-09-09 09:05:38 | NO     |
+-------------+--------------------
+-------------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
|             |
|                                                             |        |
|      625 |                   |                   |        |
+-------------+--------------------
+-------------------------------------------------------------+--------+-------------------
+----------+-------------------+-------------------+--------+
5 rows processed.
```

search_detail

Use the **search_detail** command to provide information about objects, optionally for specified statuses and on a specified day, month, or year.

Syntax

```
search_detail { YYYY[-MM[-DD]] | all } [status1 status2 …]
```

Parameters

YYYY[-MM[-DD]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

all
Use to indicate that you would like information from all dates.

status
Provide one or more status codes for the objects you would like to see. For a list of the available statuses, see Table 9-13.

Results

The requested information is provided, for example:

```
RELManager> search_detail 2021-09-10 1
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|        Poid | Mod Time           | Input
Filename                                                 | Status | State             |
Records | Start Time          | End Time            | Is ZIP?|
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|    847913365 | 2021-09-10 05:21:28 |
BRMCDR_GSM_2021-09-10T12:21:09Z_2021-09-10T12:21:14Z          |     1 | LOAD_ERROR
|       58 | 2021-09-10 05:21:25 | 2021-09-10 05:21:25 | YES    |
|    847921617 | 2021-09-10 05:21:33 |
BRMCDR_GSM_2021-09-10T12:21:14Z_2021-09-10T12:21:19Z          |     1 | LOAD_ERROR
|       42 | 2021-09-10 05:21:30 | 2021-09-10 05:21:30 | YES    |
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|             |
|                                                          |        |
|     100 |                      |                         |        |
+-------------+--------------------
+----------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
2 rows processed.
```

search_load_error

Use the **search_load_error** command to provide information about objects processed with a load error for a specified day, month, or year.

Syntax

search_load_error *[YYYY[-MM[-DD]]] [limit]*

Parameters

[YYYY[-MM[-DD]]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

If no date value is provided, information for all dates will be returned. If you do not provide a date, consider using the **limit** parameter to prevent a huge data set from being returned.

limit
Enter a number to limit the number of records returned.

Results

The requested information is provided, for example:

```
RELManager> search_load_error 2021-09-17
+--------------+---------------------
+----------------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|         Poid | Mod Time            | Input
Filename                                          | Status | State              |
Records | Start Time          | End Time            | Is ZIP?|
+--------------+---------------------
+----------------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|    848185328 | 2021-09-17 03:50:08 |
BRMCDR_GSM_2021-09-17T10:49:49Z_2021-09-17T10:49:54Z          |      1 | LOAD_ERROR
|      134 | 2021-09-17 03:50:08 | 2021-09-17 03:50:08 | YES    |
|    848193656 | 2021-09-17 03:50:09 |
BRMCDR_GSM_2021-09-17T10:49:54Z_2021-09-17T10:49:59Z          |      1 | LOAD_ERROR
|      375 | 2021-09-17 03:50:09 | 2021-09-17 03:50:09 | YES    |
|    848202225 | 2021-09-17 03:50:14 |
BRMCDR_GSM_2021-09-17T10:49:59Z_2021-09-17T10:50:04Z          |      1 | LOAD_ERROR
|      477 | 2021-09-17 03:50:14 | 2021-09-17 03:50:14 | YES    |
+--------------+--------------------
+----------------------------------------------------------------+--------+-------------------
+----------+--------------------+--------------------+--------+
|              |
|                                                                |        |
|      986 |                     |                     |        |
+--------------+---------------------
```

ORACLE

```
+------------------------------------------------------------------+--------+-------------------
+---------+--------------------+--------------------+--------+
3 rows processed.
```

search_poid

Use the **search_poid** command to provide information about objects with specified POIDs.

Syntax

search_poid *poid1 poid2 …*

Parameters

poid
Enter the POIDs for the objects you would like to see.

Results

The requested information is provided, for example:

```
RELManager> search_poid 848025372 847818825
+-------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+---------+--------------------+--------------------+--------+
|         Poid | Mod Time             | Input
Filename                                               | Status | State             |
Records | Start Time           | End Time             | Is ZIP?|
+-------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+---------+--------------------+--------------------+--------+
|    848025372 | 2021-09-13 09:45:11 |
BRMCDR_GSM_2021-09-13T04:44:54Z_2021-09-13T04:44:59Z         |      1 | LOAD_ERROR
|       50 | 2021-09-13 09:45:11 | 2021-09-13 09:45:11 | YES    |
|    847818825 | 2021-09-09 09:05:37 |
BRMCDR_GSM_2021-09-09T04:02:34Z_2021-09-09T04:02:39Z         |      0 | COMPLETE
|      125 | 2021-09-09 09:05:37 | 2021-09-09 09:05:37 | NO     |
+-------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+---------+--------------------+--------------------+--------+
|              |
|                                                           |       |
|      175 |                     |                     |       |
+-------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+---------+--------------------+--------------------+--------+
2 rows processed.
```

search_update_error

Use the **search_update_error** command to provide information about objects processed with an update error for a specified day, month, or year.

Syntax

```
search_update_error [YYYY[-MM[-DD]]] [limit]
```

Parameters

[YYYY[-MM[-DD]]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

If no date value is provided, information for all dates will be returned. If you do not provide a date, consider using the **limit** parameter to prevent a huge data set from being returned.

limit
Enter a number to limit the number of records returned.

Results

The requested information is provided, for example:

```
RELManager> search_update_error
+-------------+--------------------
+---------------------------------------------------------------+--------+-------------------
+---------+-------------------+--------------------+--------+
|         Poid | Mod Time          | Input
Filename                                             | Status | State            |
Records | Start Time         | End Time          | Is ZIP?|
+-------------+--------------------
+---------------------------------------------------------------+--------+-------------------
+---------+-------------------+--------------------+--------+
|    112498699 | 2022-06-10 08:29:11 |
BRMCDR_GSM_2022-06-10T15:28:42Z_2022-06-10T15:28:52Z          |      2 | UPDATE_ERROR
|       10 | 2022-06-10 08:29:10 | 2022-06-10 08:29:10 | NO     |
|    112523297 | 2022-06-10 08:32:21 |
BRMCDR_GSM_2022-06-10T15:32:02Z_2022-06-10T15:32:12Z          |      2 | UPDATE_ERROR
|       12 | 2022-06-10 08:32:21 | 2022-06-10 08:32:21 | NO     |
+-------------+--------------------
+---------------------------------------------------------------+--------+-------------------
+---------+-------------------+--------------------+--------+
|              |
|                                                               |        |
|       22 |                    |                    |        |
+-------------+--------------------
+---------------------------------------------------------------+--------+-------------------
+---------+-------------------+--------------------+--------+
2 rows processed.
```

summary_report

Use the **summary_report** command to provide summary information about objects processed on a specified day, month, or year.

Syntax

```
summary_report [YYYY[-MM[-DD]]]
```

Parameters

[YYYY[-MM[-DD]]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

Results

The requested information is provided, for example:

```
RELManager> summary_report
+---------------+--------+-------------------+-----------+------------+
| Day Processed | Status | State             |     Files |     Events |
+---------------+--------+-------------------+-----------+------------+
| 2021-09-09    |      0 | COMPLETE          |         8 |       1000 |
| 2021-09-09    |      1 | LOAD_ERROR        |         5 |          6 |
| 2021-09-10    |      1 | LOAD_ERROR        |         3 |        200 |
| 2021-09-13    |      0 | COMPLETE          |         8 |         71 |
| 2021-09-13    |      1 | LOAD_ERROR        |         2 |         99 |
| 2021-09-13    |      9 | CREATED           |         1 |         20 |
| 2021-09-16    |      0 | COMPLETE          |        16 |        172 |
| 2021-09-17    |      0 | COMPLETE          |        34 |      18194 |
| 2021-09-17    |      1 | LOAD_ERROR        |         3 |        986 |
| 2021-09-17    |   1280 | PREUPDATING       |         1 |        290 |
+---------------+--------+-------------------+-----------+------------+
|               |        |                   |        81 |      21038 |
+---------------+--------+-------------------+-----------+------------+
```

zip_detail

Use the **zip_detail** command to provide information about ZIP-file objects, optionally for specified statuses and on a specified day, month, or year. The ZIP-file objects can be created either because you are using the ZIP_DB mode or because you are using the DIRECT mode, the original processing failed, and you have set next_mode for retry to ZIP_DB. See "ECE RE Manager Plug-in Retry Configuration" for more information about retry configuration.

Syntax

```
zip_detail { YYYY[-MM[-DD]] | all } [status1 status2 …]
```

Parameters

YYYY[-MM[-DD]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

all
Use to indicate that you would like information from all dates.

status
Provide one or more status codes for the objects you would like to see. For a list of the available statuses, see "Troubleshooting ZIP File Processing".

Results

The requested information is provided, for example:

```
RELManager> zip_detail 2021-09-13
+--------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+------------+
|         Poid | Mod Time            | Input
Filename                                                  | Status | State             |
Records | Zip Status | Zip State         | Zip Size(Mb) | Extracted? |
+--------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+------------+
|    848000640 | 2021-09-13 09:29:39 |
BRMCDR_GSM_2021-09-13T04:22:49Z_2021-09-13T04:22:54Z           |      1 | LOAD_ERROR
|        20 |             4 | DATA_ERROR          |       0.010 | NO         |
|    848025372 | 2021-09-13 09:45:11 |
BRMCDR_GSM_2021-09-13T04:44:54Z_2021-09-13T04:44:59Z           |      1 | LOAD_ERROR
|        50 |             4 | DATA_ERROR          |       0.015 | NO         |
+--------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+------------+
|              |
|                                                                  |        |
|        70 |             |                     |       0.025 |            |
+--------------+--------------------
+------------------------------------------------------------------+--------+-------------------
+----------+-----------+-------------------+-------------+------------+
2 rows processed.
```

zip_file_reload

Use the **zip_file_reload** command to reload corrected data from the file system where REL Manager is running and replace the equivalent session data in the Oracle Database for reprocessing.

Syntax

zip_file_reload *retry poid1 poid2 …*

Parameters

retry
Set to **true** to set the loaded data to READY state, so that it will automatically be reprocessed. Set to **false** to set the data to PROCESS_ERROR. In that case you will need to run **retry_session** when you are ready to reprocess the data.

poid
Enter the POIDs for the objects you would like to reprocess.

Results

The corrected items will be compressed and the resulting ZIP files will be loaded into the BRM database. The original directory names will be appended with **.reloaded_*timestamp***. This data is now ready to be reprocessed using the retry_session command. Once the data has been successfully reprocessed in BRM, you may delete the data in the directory.

zip_file_search

Use the **zip_file_search** command to provide information about ZIP files that have been extracted from the database for reprocessing using the zip_file_write command. You can optionally retrieve data only for specified statuses and on a specified day, month, or year.

Syntax

zip_file_search [*limit*]

Parameters

limit
Use to limit the number of results returned.

Results

The requested information is provided, for example:

```
RELManager> zip_file_search 3
+--------------+---------------------
+----------------------------------------------------------+--------+--------------------
+----------+--------------------+--------------------+--------+
|        Poid | Mod Time            | Input
Filename                                             | Status | State              |
Records | Start Time          | End Time            | Is ZIP?|
+--------------+---------------------
+----------------------------------------------------------+--------+--------------------
+----------+--------------------+--------------------+--------+
|    847929853 | 2021-09-10 05:31:21 |
BRMCDR_GSM_2021-09-10T12:31:04Z_2021-09-10T12:31:09Z          |      1 | LOAD_ERROR
|      100 | 2021-09-10 05:31:20 | 2021-09-10 05:31:20 | YES    |
|    848000640 | 2021-09-13 09:29:39 |
BRMCDR_GSM_2021-09-13T04:22:49Z_2021-09-13T04:22:54Z          |      1 | LOAD_ERROR
|       20 | 2021-09-13 09:23:03 | 2021-09-13 09:23:03 | YES    |
|    848025372 | 2021-09-13 09:45:11 |
BRMCDR_GSM_2021-09-13T04:44:54Z_2021-09-13T04:44:59Z          |      1 | LOAD_ERROR
|       50 | 2021-09-13 09:45:11 | 2021-09-13 09:45:11 | YES    |
```

ORACLE

```
+-------------+-------------------
+--------------------------------------------------------------+-------+-------------------
+----------+-------------------+-------------------+-------+
|              |
|                                                             |         |
|       170 |                       |                      |         |
+-------------+-------------------
+--------------------------------------------------------------+-------+-------------------
+----------+-------------------+-------------------+-------+
3 rows processed.
```

zip_file_write

Use the **zip_file_write** command to extract the contents of ZIP files from the database and expand them onto the local file system. You specify the POIDs of files that have not been processed successfully. This enables you to edit the CDR data to correct any load errors. You can then reload the data using the zip_file_reload command.

> **✎ Note:**
>
> You should use the **zip_file_write** command to write files only for POIDs that are in the LOAD_ERROR (1) or CREATED (9) states. The system will not allow you to reload files in any other state.

Syntax

zip_file_write *poid1 poid2 …*

Parameters

poid
Enter the POIDs for the objects you would like to extract.

Results

The entries for the specified POIDs are retrieved from the database, extracted from the ZIP file, and written to the file system. The command also indicates the location of the extracted files. The directory that will contain the extracted files is set using the **zip_directory** parameter in the **RELM.properties** file. See "General REL Manager Configuration" for more information about this parameter. If the directory for a particular POID already exists, the command will fail. This will prevent overwriting data that may already have been edited.

zip_summary_report

Use the **zip_summary_report** command to provide summary information about ZIP-file-based objects processed on a specified day, month, or year.

Syntax

zip_summary_report *[YYYY[-MM[-DD]]] [limit]*

Parameters

[YYYY[-MM[-DD]]]
Provide one of the following to specify the dates for which you want more information:

- Year

- Year and month

- Year, month, and day

If no date value is provided, information for all dates will be returned. If you do not provide a date, consider using the **limit** parameter to prevent a huge data set from being returned.

limit
Enter a number to limit the number of records returned.

Results

The requested information is provided, for example:

```
RELManager> zip_summary_report 2021-09-09
+--------------+--------+------------------+------------------+-----------+------------
+------------+
| Day Processed | Status | State            | Load State       |     Files |     Events
|    Size(Mb) |
+--------------+--------+------------------+------------------+-----------+------------
+------------+
| 2021-09-09   |      4 | DATA_ERROR       | LOAD_ERROR       |         1 |          1
|      0.006 |
| 2021-09-09   |      5 | PROCESS_ERROR    | LOAD_ERROR       |         2 |          3
|      0.012 |
+--------------+--------+------------------+------------------+-----------+------------
+------------+
|              |        |                  |                  |         3 |          4
|      0.018 |
+--------------+--------+------------------+------------------+-----------+------------
+------------+
2 rows processed.
```