Preface................................................................................................................................................................ x i
Audience................................................................................................................................................................ xi
Related Documentation................................................................................................................................................ xi
Documentation Accessibility........................................................................................................................................... xi

1 About Configuring Charging in Elastic Charging Controller

About Configuring Usage Charging in ECE........................................................................................................ 1-1
About Balance Management in a Prepaid Session......................................................................................... 1-2
About ECE Notifications......................................................................................................................................... 1-3

2 About ECE APIs

About the ECE API.................................................................................................................................................... 2-1
About the Charging API ........................................................................................................................................ 2-2
  About Charging Operation Types........................................................................................................................ 2-2
About the Authentication API.............................................................................................................................. 2-3
About the Custom Plug-in API.................................................................................................................................. 2-3

3 List of Charging Features

Charging Features .................................................................................................................................................... 3-1

4 Managing Online Charging Sessions

Configuring How ECE Manages Online Sessions.......................................................................................... 4-2
Configuring ECE to Generate Midsession Rated Events........................................................................ 4-2
Configuring Incremental Rating for Midsession Rating Condition Changes.............................................. 4-5
Configuring Incremental Rating for Tariff Changes....................................................................................... 4-7
  Configuring ECE to Support Tariff Time Change............................................................................................. 4-8
Using Session Connect Time for Charging.................................................................................................... 4-9
Configuring Reservation Expiration and Validity............................................................................................ 4-9
Configuring a Minimum Quantity for Reservation....................................................................................... 4-11
Configuring Reservation Quota for Services.................................................................................................... 4-12
Managing Dynamic Quotas for Online Sessions............................................................................................ 4-13
  Triggering RAR Notifications for Ongoing Sessions...................................................................................... 4-13
Configuring Rounding for a Resource ............................................................................................................... 4-13
Configuring Rounding for Reverse Rating on Multiple RUMs................................................................. 4-15
Enabling Server-Initiated Reauthorization Requests ......................................................... 4-15
Configuring Systemwide Consumption Rules for Balances ............................................. 4-17
Configuring ECE to Return Remaining-Balance Information in Usage Responses .......... 4-18
Configuring Taxation ........................................................................................................ 4-19
Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers ................................................................................................................................. 4-20
Configuring How ECE Manages Active Sessions When Network Elements Fail ............ 4-21
Configuring ECE to Redirect Subscriber Sessions to a Service Portal ......................... 4-21
Enabling Match Factor in ECE .......................................................................................... 4-26

5 Configuring Subscriber Preferences
Configuring Subscriber Preferences .................................................................................. 5-1
  Maintaining Subscriber Preferences with Customer Center ........................................... 5-1
  About Regulating Permissions to Update Subscriber Preferences ................................ 5-2
  Configuring ECE to Enrich External Notifications with Subscriber Preference Information. 5-2
  Customizing Subscriber Preferences ............................................................................ 5-4

6 Configuring Top-Ups
Integrating Top-Up clients ............................................................................................... 6-1
Detecting Duplicate Top-Up Requests ............................................................................. 6-1
Top-up API ....................................................................................................................... 6-3

7 Configuring Balance Queries
Integrating Balance Query clients ................................................................................... 7-1
About Sending Authentication Queries ............................................................................. 7-2
About Sending Balance Queries ..................................................................................... 7-2
Configuring Debit Request History ................................................................................ 7-2
Balance API .................................................................................................................... 7-3

8 Advice of Charge and Advice of Promotion
About Advice of Charge .................................................................................................... 8-1
About Advice of Promotion ............................................................................................... 8-1

9 Configuring Multiple Services Credit Control
About Multiple Services Credit Control ............................................................................ 9-1

10 Synchronizing Data Between ECE and the BRM Database
Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates ........... 10-1
Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE .................................................. 10-2
Configuring Item Assignment for Rated Events .............................................................. 10-2
Configuring Life Cycle States in ECE for BRM .............................................................. 10-4
Configuring ECE to Send Rated Events to BRM ............................................................ 10-5
  Configuring Rated Event Publisher ............................................................................... 10-5
  Configuring Rated Event Formatter ............................................................................. 10-6
  Troubleshooting Rated Event Formatter Processing .................................................... 10-9
11 Generating POIDs for Rated Events

About Generating POIDs in ECE
Configuring ECE to Generate POIDs for Prepaid Events
- Enabling Prepaid Event Partitions in BRM
- Configuring Cluster ID
- Enabling POID Generation for Prepaid Events in ECE

12 Connecting ECE to a Diameter Client

Overview of Network Integration Using Diameter Gateway
Network Integration for Sp and Sy Interface (Policy) Requests
Network Integration for Gy Interface Requests
- How Diameter Gateway Creates Usage Requests
- About Usage Request Fixed Attributes
- Editing the Mediation Specification File
- Network Integration for Gy Balance Query Requests
- Network Integration for Gy Top-Up Requests
- Sending Multiple-Service Credit Control (MSCC) Requests from Diameter Gateway
- Configuring Subscriber ID Lookups
Adding Custom AVPs for Usage Requests
Using Incremental or Cumulative Accounting for Usage Requests
- Configuring Accounting Mode for Diameter Gateway
Configuring Notifications for Diameter Gateway
Configuring Alternative Diameter Peers for Notifications
Viewing Active Diameter Peers
Handling Requests When Charging Servers Are Unavailable

13 Connecting ECE to a RADIUS Client

Overview of Authentication and Accounting Using RADIUS Gateway
About RADIUS Gateway Authentication
Authenticating Access Requests by Using PAP
- Authenticating Access Requests by Using CHAP
- Authenticating Access Requests by Using EAP
Loading Data Keys Extracted from BRM into ECE
Customizing the RADIUS Data Dictionary
- About the RADIUS Data Dictionary
- Creating a Custom Data Dictionary
- Selecting a RADIUS Data Dictionary When Using Different NAS Vendors
- Adding Custom Vendor-Specific Attributes
Loading the RADIUS Mediation Specification Data
About Mapping RADIUS Network Attributes to Event Attributes
- Mapping RADIUS Network Attributes to Event Attributes
About RADIUS Gateway Accounting
Sample Extensions ................................................................. 16-20
Diameter Gateway Extension - Service........................................ 16-20
Pre-Rating Extension - Dynamic Quota Management ...................... 16-20
  Dynamic Quota Management - Modifying Quota Based on Network Type 16-21
  Dynamic Quota Management - Modifying Requested Quota .......... 16-21
  Dynamic Quota Management - Modifying Default Quota Configuration 16-21
Pre-Rating Extension - Retrieving Function Values for Discount Expressions 16-21
Pre-Rating Extension - Overriding Price in Product Offerings .......... 16-22
Rating Extension - Custom Item Assignment ................................ 16-22
Post-rating Extension - Complex Taxation .................................. 16-22
Rating/Charging Extension - Triggering RAR Notifications .......... 16-23
Post-Rating Extension - Adding or Deleting Rating Periods .......... 16-23
Post-Charging Extension - Adding Custom Data to Usage Responses and Notifications 16-23
Post-Charging Extension - Overriding Dynamic Quota ................. 16-23
Post-Charging Extension - Adding or Modifying Redirection Rules .. 16-23
Post-Update Extension - Enriching External Notifications .......... 16-24
Extensions Data Load Sample ................................................. 16-24
How To Use the Sample Extensions ........................................ 16-24
Validating Sample Extensions .................................................. 16-26

17 ECE Sample Programs
About the Sample Programs ..................................................... 17-1
Compiling and Running the Sample Programs ......................... 17-5
Example of SampleDebitRefundSession ................................ 17-6

18 Charging Utilities
  pin_update_items_journals ..................................................... 18-2
  query ....................................................................................... 18-3

Part I Appendices
A ECE Notification Reference
  Charging Notifications .......................................................... A-1
  BRM Notifications ............................................................... A-2
  About the Notifications API ............................................... A-2

B Sample Notification Payloads
  Aggregated Threshold Breach Event (Aggregated Based on Balance Element ID) B-1
  Billing Event ........................................................................ B-2
  Credit Ceiling Breach Event ................................................ B-2
  Credit Floor Breach Event .................................................... B-2
  External Top-up Event .......................................................... B-3
  First Usage Validity ............................................................. B-3
  Life-Cycle Transition .......................................................... B-4
  Replenish POID ID Event ..................................................... B-4
  Spending Limit .................................................................... B-5
<table>
<thead>
<tr>
<th>Event Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscriber Preference Event</td>
<td>B-5</td>
</tr>
<tr>
<td>Threshold Breach Event (Breach Direction Down)</td>
<td>B-6</td>
</tr>
<tr>
<td>Threshold Breach Event (Breach Direction Up)</td>
<td>B-7</td>
</tr>
<tr>
<td>Top-up Event</td>
<td>B-7</td>
</tr>
</tbody>
</table>

### C Specifications and Standards Compliance in ECE

#### About Specifications and Standards Compliance

C-1
Preface

This guide describes how to implement charging in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Audience

This guide is intended for application administrators and charging experts who customize and administer ECE.

Related Documentation

Before reading this book, read BRM Concepts.

For information about creating the product offerings used in charging, see BRM Creating Product Offerings.

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.
This document provides an overview of how to implement charging in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

- About Configuring Usage Charging in ECE
- About Balance Management in a Prepaid Session
- About ECE Notifications

See also:

- List of Charging Features
- About ECE APIs
- Managing Online Charging Sessions

### About Configuring Usage Charging in ECE

To configure usage charging in BRM, you configure ECE as follows:

- Configure ECE business rules that control aspects such as a default authorization amount, a default flat tax rate, and so on. To do so, you use a JMX editor such as JConsole to edit MBeans.

- Configure ECE notifications. See "About ECE Notifications".

- Configure how ECE integrates with the BRM server. See "Synchronizing Data Between ECE and the BRM Database".

- Configure ECE to receive pricing data from PDC. See the discussion about configuring Pricing Updater in BRM System Administrator’s Guide.

You can customize ECE as follows:

- Use the ECE API to integrate client applications with ECE (for example, implement a top-up client). For offline charging, ECE is preintegrated with Oracle Communications Offline Mediation Controller. For online charging, ECE uses Diameter Gateway for network integration (for creating ECE requests for all supported request types). See "About the ECE API" for information.

- Use ECE extensions to implement Diameter-request processing extensions or rating extensions. For example, implement pre-rating and post-rating customizations. See "Customizing Rating".
About Balance Management in a Prepaid Session

In prepaid sessions, ECE and Diameter Gateway work together with the network system to manage an event as it occurs. When a subscriber starts a prepaid call, the network collects information about the customer and sends authentication and authorization requests to ECE. ECE processes the requests and returns the results immediately so the network can connect the call. While the session is in progress, ECE tracks the subscriber’s balance to ensure that it is sufficient to pay for the call.

ECE performs the following functions:

1. Authenticates customers by comparing the ID the customer provides with the ID stored in ECE.
   - For telco services, the ID is typically the MSID.
   - For broadband services, the ID is typically a login name and password.
2. Authorizes customers to use the service. ECE can perform these checks:
   - **Credit limit checking.** Determines whether the customer’s account balance exceeds the specified limit.
   - **Service status checking.** Confirms that the requested service is active in the customer’s account.
   - **Duplicate session checking.** Checks for duplicate sessions.
3. Reserves a balance amount for the session. For example, customers can be authorized to download 100 bytes of data or to make a 30-minute telephone call.
   To reserve an amount, ECE does the following:
   - Receives the requested amount from the network and determines whether the user has a sufficient amount in his balance, based on the charge offers and discount offers that he owns and any amounts already reserved.
   - If the balance amount is insufficient, ECE calculates the maximum authorization based on the customer’s credit limits. The effects of discounts, discount sharing, and charge sharing are included in the calculation of the maximum amount to authorize.

   ECE sends the validity time for the active reservation or reservation validity to the network mediation client. Reservation validity specifies how long a session can continue before the client must ask for a reauthorization.

   ECE sends a reservation expiration to the network mediation client. Reservation expiration specifies how long a session can continue before the client must report the consumed usage to ECE.

   When a prepaid session is authorized, BRM reserves a portion of the customer’s balance for the event. This prevents customers from using that balance amount for other services while the session is in progress.

   BRM authorizes a customer to use a service for the following:
   - A specified duration or volume.
   - (Volume-based authorizations only) A specified validity period.

4. When the session is ended, ECE sends information about the rated event to the BRM database, and updates the subscriber’s balance on the BRM database. ECE then returns any unused reserved balance amounts to the customer’s balance.
Active session and balance reservations are checked for expiration and are removed if the object has expired. Usually, only the terminated or canceled charge offer is cleaned up. Other charge offers owned by the same customer and that share the balance object with the original charge offer, however, are also cleaned up.

During a session, ECE does the following:

- Reauthorizes customers for extended usage if necessary. Reauthorization for prepaid services extends the following:
  - Authorized duration or volume
  - Validity period
- Alerts the network that a change that might require reauthorization occurs in a customer’s account. For example, the customer is granted a balance amount that might apply to the current session. This is called server-initiation reauthorization.
- Cancels authorization for failed connections. After a session is authorized, the external network is sometimes unable to connect the service. This can occur for the following reasons:
  - The call’s destination was unavailable.
  - The validity period expired before the service was connected.
  - The customer terminated the session before the service was connected.
- Updates balances in the customer’s account.
- If your system is configured to receive in-session notifications from BRM (that is, when the piggyback business parameter is enabled), appends specific in-session notifications to the responses it provides for authorization and reauthorization requests sent by a supported network connectivity application. You can configure customer preferences for sending notifications (for example, by SMS in a certain language). See "Configuring Subscriber Preferences".
- If the session uses policy-driven charging, tracks the balance thresholds that trigger credit-limit notifications to the policy controller. Both in-session and out-of-session notifications are supported.

**About ECE Notifications**

ECE supports in-session notifications and external notifications.

*In-session notifications* are used during online charging (for example, to manage prepaid sessions by returning reserved balance amounts).

*External notifications* contain information that external applications need, for example:

- The network mediation system can use data in the external notification in conjunction with customer policy data for implementing network policy control.
- BRM can use data in the external notification for running billing for a specific customer. The BRM Gateway sends the relevant data to BRM in the external notification for triggering billing.

All notifications are disabled by default. Notifications can be enabled or disabled by using these settings:
About ECE Notifications

- **NONE**: No notification is sent.
- **ASYNCHRONOUS**: An asynchronous notification is sent.

Advice of Charge (AoC) events and threshold breach service events can also use these settings:

- **PIGGYBACK**: Events are included as an in-session notification on the usage response message (send as a block on the usage response), not as an asynchronous event.
- **ASYNC_PIGGYBACK**: Both asynchronous and in-session notifications are sent.

For ECE to publish external notifications, configure the JMS credentials for the JMS server on which the notification queue (JMS topic) resides. For details, see the individual notification configuration procedures.
This document lists the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) APIs.

Topics in this document:

- About the ECE API
- About the Charging API
- About the Authentication API
- About the Custom Plug-in API

See also:

- List of Charging Features
- Top-up API
- Balance API

### About the ECE API

ECE is preintegrated with Offline Mediation Controller. To integrate other clients, such as top-up systems, you use the ECE APIs. See *BRM Elastic Charging Engine Java API Reference* for information.

The ECE SDK includes:

- Client libraries that enable your applications to connect to ECE and build usage requests.
- Sample programs that demonstrate how to use the ECE client APIs.

ECE SDK includes a set of sample programs. You use these sample programs in the following ways:

- Use the sample programs as code samples for writing custom applications.
- Run sample programs to send requests to ECE and receive responses.
- Use the sample program scripts as a guide for integration of the ECE client into your build system (Maven, Ant and so on).

For information about how to use the sample programs, see "ECE Sample Programs".

The ECE SDK is installed in \texttt{ECE\_home/ocecesdk}. 


About the Charging API

The ECE charging API supports the following operation types:

- Initiate
- Update
- Terminate
- Cancel
- Debit_Unit
- Debit_Amount
- Refund_Unit
- Refund_Amount
- PriceEnquiry
- StartAccounting
- UpdateAccounting
- AccountingOn
- AccountingOff

To send usage requests to ECE, client applications must call the ECE charging APIs according to the usage request builder defined by the ECE event definition.

For details about the charging API, see oracle.communication.brm.charging.brs and oracle.communication.brm.charging.messages in BRM ECE Java API Reference.

About Charging Operation Types

The ECE API is designed to receive usage requests and send usage responses for common operation types in the charging industry.

ECE usage charging supports the following operation types:

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate</td>
<td>Commencement of a session-based charging operation.</td>
</tr>
<tr>
<td>Update</td>
<td>Continuation of a session-based charging operation.</td>
</tr>
<tr>
<td>Terminate</td>
<td>Conclusion of a single non-session based charging operation.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Complete cancellation of a session-based charging operation.</td>
</tr>
<tr>
<td>Refund_Amount</td>
<td>Refund a specific amount to a specific balance resource.</td>
</tr>
<tr>
<td>Refund_Unit</td>
<td>Refund a calculated amount, based on units consumed, to the impacted resource(s).</td>
</tr>
<tr>
<td>Debit_Amount</td>
<td>Debit a specific amount to a specific balance resource.</td>
</tr>
<tr>
<td>Debit_Unit</td>
<td>Debit a calculated amount, based on Units consumed, to the impacted resource(s).</td>
</tr>
<tr>
<td>Price_Enquiry</td>
<td>Generate a price estimation without any balance reservations occurring. It is used when there isn’t a high probability of receiving a charging request. For example, Price_Enquiry might be called to get the price of an event charge to display in a content portal.</td>
</tr>
</tbody>
</table>
Each charging operation type requires an input payload that supplies fields which are relevant to the charging operation.

The BALANCE_QUERY operation type is used for query requests. The query request is built using the Query Request Builder.

The ACCOUNTING_ON and ACCOUNTING_OFF operation types are used for management requests. Management requests are built using the Management Request Builder.

For offline charging, requests are typically submitted for a single event that represents the entire charge (using the Terminate operation type). Session-based operations such as Initiate and Update are not as common for offline charging; however, these operation types are used when using a stream protocol like Radius or Rf in which ECE is used to record the consumption of resources (quantity consumption) as the session continues.

ECE processes charging operations by forwarding usage requests to the applicable combinations of charge, alteration and distribution rate plans. ECE creates the rate plan expressions required for usage charging by using fields which are supplied in the request specification payloads.

The sample request specification files demonstrate the data ECE requires to support the charging operation types.

### About the Authentication API

Use the authentication API to query the login and password of subscribers.

Use the login and password information to do the following:

- Implement authentication methods outside of the ECE charging server
- Enable subscribers to validate their login and password credentials against a charge offer to which they are subscribed

For details about the authentication API, see `oracle.communication.brn.charging.messages.query` in BRM ECE Java API Reference.

### About the Custom Plug-in API

Use the custom plug-in API to implement a custom plug-in for formatting rated events into the format required by an external system.

A sample custom plug-in is available in the ECE SDK package. The sample is called `SampleRatedEventFormatterCustomPlugin.java`.

For detailed information about the custom plug-in API, see the discussion about `oracle.communication.brn.charging.ratedevent.custom.CustomPlugin` in BRM Elastic Charging Engine Java API Reference.
This document lists the charging features in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

To learn about charging, see *BRM Concepts*.

### Charging Features

This table lists charging features.

<table>
<thead>
<tr>
<th>Task</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage online charging sessions</td>
<td>Managing Online Charging Sessions</td>
</tr>
<tr>
<td></td>
<td>Configuring How ECE Manages Online Sessions</td>
</tr>
<tr>
<td></td>
<td>Configuring ECE to Generate Midsession Rated Events</td>
</tr>
<tr>
<td></td>
<td>Configuring Incremental Rating for Midsession Rating Condition Changes</td>
</tr>
<tr>
<td></td>
<td>Configuring Incremental Rating for Tariff Changes</td>
</tr>
<tr>
<td></td>
<td>Using Session Connect Time for Charging</td>
</tr>
<tr>
<td></td>
<td>Configuring Reservation Expiration and Validity</td>
</tr>
<tr>
<td></td>
<td>Configuring a Minimum Quantity for Reservation</td>
</tr>
<tr>
<td></td>
<td>Configuring Reservation Quota for Services</td>
</tr>
<tr>
<td></td>
<td>Managing Dynamic Quotas for Online Sessions</td>
</tr>
<tr>
<td></td>
<td>Configuring Rounding for a Resource</td>
</tr>
<tr>
<td></td>
<td>Configuring Rounding for Reverse Rating on Multiple RUMs</td>
</tr>
<tr>
<td></td>
<td>Enabling Server-Initiated Reauthorization Requests</td>
</tr>
<tr>
<td></td>
<td>Configuring Systemwide Consumption Rules for Balances</td>
</tr>
<tr>
<td></td>
<td>Configuring ECE to Return Remaining-Balance Information in Usage Responses</td>
</tr>
<tr>
<td>Task</td>
<td>Topics</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manage online charging sessions</td>
<td>Configuring Taxation</td>
</tr>
<tr>
<td></td>
<td>Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers</td>
</tr>
<tr>
<td></td>
<td>Configuring How ECE Manages Active Sessions When Network Elements Fail</td>
</tr>
<tr>
<td></td>
<td>Configuring ECE to Redirect Subscriber Sessions to a Service Portal</td>
</tr>
<tr>
<td></td>
<td>Enabling Match Factor in ECE</td>
</tr>
<tr>
<td>Configure subscriber preferences; for example, how to receive</td>
<td>Configuring Subscriber Preferences</td>
</tr>
<tr>
<td>notifications from the network.</td>
<td></td>
</tr>
<tr>
<td>Configure ECE to interface with top-up systems to manage balances.</td>
<td>Configuring Top-Ups</td>
</tr>
<tr>
<td>Configure ECE to interface with external systems to manage</td>
<td>Configuring Balance Queries</td>
</tr>
<tr>
<td>balance queries.</td>
<td></td>
</tr>
<tr>
<td>Configure Advice of Charge.</td>
<td>About Advice of Charge</td>
</tr>
<tr>
<td>Configure Advice of Promotion.</td>
<td>About Advice of Promotion</td>
</tr>
<tr>
<td>Configure Multiple Services Credit Control (MSCC).</td>
<td>Configuring Multiple Services Credit Control</td>
</tr>
<tr>
<td>Synchronize data between ECE and the BRM database; for example,</td>
<td>Synchronizing Data Between ECE and the BRM Database</td>
</tr>
<tr>
<td>customer data, and service life-cycle states.</td>
<td></td>
</tr>
<tr>
<td>Configure ECE to generate POIDs for events created in ECE.</td>
<td>Generating POIDs for Rated Events</td>
</tr>
<tr>
<td>Configure ECE to integrate with a Diameter client.</td>
<td>Connecting ECE to a Diameter Client</td>
</tr>
<tr>
<td>Configure ECE to integrate with a RADIUS client.</td>
<td>Connecting ECE to a RADIUS Client</td>
</tr>
<tr>
<td>Implement policy-driven charging.</td>
<td>Configuring Policy-Driven Charging</td>
</tr>
<tr>
<td>Test ECE.</td>
<td>Testing ECE</td>
</tr>
<tr>
<td>Customize usage rating in ECE.</td>
<td>Customizing Rating</td>
</tr>
</tbody>
</table>
Managing Online Charging Sessions

This document describes how to manage Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) online charging sessions.

Topics in this document:

- Configuring How ECE Manages Online Sessions
- Configuring ECE to Generate Midsession Rated Events
- Configuring Incremental Rating for Midsession Rating Condition Changes
- Configuring Incremental Rating for Tariff Changes
- Using Session Connect Time for Charging
- Configuring Reservation Expiration and Validity
- Configuring a Minimum Quantity for Reservation
- Configuring Reservation Quota for Services
- Managing Dynamic Quotas for Online Sessions
- Configuring Rounding for a Resource
- Enabling Server-Initiated Reauthorization Requests
- Configuring ECE to Return Remaining-Balance Information in Usage Responses
- Configuring Taxation
- Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers
- Configuring How ECE Manages Active Sessions When Network Elements Fail
- Configuring ECE to Redirect Subscriber Sessions to a Service Portal
- Enabling Match Factor in ECE

See also:

- List of Charging Features
- Configuring Subscriber Preferences
- Connecting ECE to a Diameter Client
- Connecting ECE to a RADIUS Client
- Configuring Top-Ups
- Configuring Balance Queries
Configuring How ECE Manages Online Sessions

You can configure the following systemwide reservation properties:

- How ECE generates midsession rated events. See "Configuring ECE to Generate Midsession Rated Events" for information.
- Enable ECE to support tariff time change. See "Configuring ECE to Support Tariff Time Change" for information.
- The systemwide reservation expiration and reservation validity periods. See "Configuring Reservation Expiration and Validity" for information.
- The minimum quantity to reserve for a usage request. See "Configuring a Minimum Quantity for Reservation" for information.
- The systemwide minimum reservation amount, if the usage request does not specify a requested amount. See "Configuring Reservation Quota for Services" for information.
- The systemwide rounding rule for currency and noncurrency resources. See "Configuring Rounding for a Resource" for information.
- The systemwide rounding rule to round up the fractional value of the authorized resource. See "Configuring Rounding for Reverse Rating on Multiple RUMs" for information.
- Enable ECE to ask the network to send a reauthorization request when customer data changes might require a reauthorization. See "Enabling Server-Initiated Reauthorization Requests" for information.
- How to handle a network element failure. See "Configuring How ECE Manages Active Sessions When Network Elements Fail" for information.

Configuring ECE to Generate Midsession Rated Events

By default, ECE generates a rated event for a network session only when a Diameter terminate operation ends the session. You can also configure ECE to generate a rated event whenever a Diameter update operation occurs during the network session. Such events are called *midsession rated events*.

To generate midsession rated events, you enable the feature and then define conditions, called *triggers*, that initiate the generation of such events. Triggers are based on one or more of the following criteria:

- Duration (for example, every 4 hours that a session is active)
- Quantity (for example, whenever downloaded data totals 70 MB or more)
- Time of day (for example, daily at 23:00:00 during the life of the session)

Each trigger is associated with a service-event pair. If an ongoing session meets the trigger conditions at the time an update operation occurs, a midsession rated event for the specified service is generated.

*Note:* Trigger conditions are examined only during update operations. If a trigger condition is "every 200 MB" but an update operation does not occur until the total is 288 MB, the rated event is for 288 MB, not 200 MB. The same applies to duration criteria.
For example, the following code triggers the generation of a rated /data_usage event for a DATA service's ongoing network session if at least one of the following conditions is true:

- The combined values of the event's input_volume and output_volume fields total 70 MB or more.
- The current time minus the time the last midsession rated event was generated is greater than or equal to 7 hours.
- The current time is greater than or equal to 11 p.m.

```xml
<midSessionCdrConfiguration
    config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MidSessionCdrConfiguration'
    midSessionCdrEnabled='true'>

    <productConfigurationGroup config-class='java.util.ArrayList'>
        <productLifecycleConfiguration
            config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MidSessionCdrConfiguration'
            productName='DATA'>

            <eventConfigurationGroup config-class='java.util.ArrayList'>
                <eventConfiguration
                    config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MidSessionCdrConfiguration'
                    eventType='DATA_USAGE'>

                    <triggerConfiguration
                        config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MisSessionCdrTriggerConfiguration'
                        durationunit='HOURS'
                        durationvalue='7'/>

                    <triggerConfiguration
                        config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MidSessionCdrTriggerConfiguration'
                        quantifyfields='input_volume;output_volume'
                        quantityunit='MEGABYTES'
                        quantityvalue='70'/>

                    <triggerConfiguration
                        config-class='oracle.communication.brm.charging.appconfiguration.beans.midsessioncdr.MidSessionCdrTriggerConfiguration'
                       timeofday='23:00:00'/>

                </eventConfiguration>
            </eventConfigurationGroup>
        </productLifecycleConfiguration>

    </productConfigurationGroup>
</midSessionCdrConfiguration>
```
To configure ECE to generate midsession rated events:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/occeserver/config/eeTopology.conf file, where ECE_home is the directory in which ECE is installed.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

2. Expand the ECE Configuration node.

3. Expand charging.midSessionCdrConfiguration.

4. Expand Attributes.

5. Set the MidSessionCdrEnabled attribute to true.

6. Define trigger conditions for one or more service-event pairs:
   a. Expand Operations.
   b. Click addOrUpdateMidSessionCdrTriggerDetails.
   c. Specify values for the fields listed in Table 4–1:

Table 4–1  Fields for Defining Midsession Rated Event Triggers

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>productType</td>
<td>Name of the service for which you are creating the trigger (for example, &quot;DATA&quot;).</td>
</tr>
<tr>
<td>eventType</td>
<td>Name of the event for which you are creating the trigger (for example, &quot;DATA_USAGE&quot;).</td>
</tr>
<tr>
<td>triggerName</td>
<td>Name of the trigger you are defining.</td>
</tr>
<tr>
<td>qtyFields</td>
<td>Name of one or more event fields to which a quantity condition applies (for example, &quot;input_volume; output_volume&quot;). Use a semicolon (;) to separate field names. Values in the fields are summed.</td>
</tr>
<tr>
<td>qtyUnit</td>
<td>Unit of measure for conditions based on quantity (for example, &quot;MEGABYTES&quot;).</td>
</tr>
<tr>
<td>qtyValue</td>
<td>Total quantity of the unit that triggers event generation (for example, &quot;70&quot;).</td>
</tr>
<tr>
<td>durationUnit</td>
<td>Unit of measure for conditions based on duration (for example, &quot;HOURS&quot;).</td>
</tr>
</tbody>
</table>
Configuring Incremental Rating for Midsession Rating Condition Changes

You can configure ECE to apply midsession rating condition changes to data sessions that generate a reauthorization request (RAR) and affect the session's rating. When changes in charging occur during an ongoing data session, they trigger a RAR. You can configure ECE to generate a rated event whenever charging changes are triggered during the session. In all scenarios, ECE considers changes in charging conditions for the portion of the session for which they are applicable.

Note: Incremental rating for midsession rating condition changes is supported only for those sessions that generate a RAR.

To configure ECE to use incremental rating for midsession rating condition changes:

1. Access the ECE MBeans:
   a. Log on to the driver machine, which is the machine on which you installed ECE.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceceserver/config/eceTopology.conf` file.
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor's MBean hierarchy, expand the ECE Configuration node.

2. Expand charging.reservationConfig.


4. For each product that you offer, do the following:
   a. Select enabledOrDisableNonLinear.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>durationValue</code></td>
<td>Amount of the unit that triggers event generation (for example, &quot;70&quot;).</td>
</tr>
<tr>
<td><code>timeOfDay</code></td>
<td>A particular time of day in a 24-hour clock at which to generate the event (for example, &quot;23:00:00&quot;, which indicates 11 p.m.). Use the <code>hh:mm:ss</code> format.</td>
</tr>
</tbody>
</table>

A trigger with one `TriggerConfiguration` block is created for the specified service-event pair. All conditions in the block (quantity, duration, time of day) must be met to generate a midsession rated event.

d. (Optional) Do one of the following:

   To define another trigger, click the plus sign in the panel's upper right corner, and repeat step 4 for a different service-event pair.

   To add a `TriggerConfiguration` block to the current trigger, click the plus sign in the panel's upper right corner, and repeat step 4 for the same service-event pair.
b. Specify values for the following parameters:

   **productType.** Specify the name of the product defined in the ECE request specification data (for example, DATA).

   **enableOrDisable.** Set to **true** to enable incremental rating.

c. Click the **enabledOrDisableNonLinear** button.

The following sections describe a couple of scenarios of a non-predictable rating condition change during a data session and the balances and charges during and at the end of the session.

**Scenario: Noncurrency Voucher Top-up During a Session**

Given a subscriber with:

- A charge offer named **CO Usage Data**
- A charge of $1 per 1 megabyte (MB) of usage
- A discount offer which consumes from available free MB
- An initial balance of 0 bytes of **Free Use Data** and USD = 0
- Incremental rating enabled for product type **TelcoGPRS**
- Eligible offers selected based on duration

The usage charging flow is as follows:

1. The subscriber starts a data session at 23:50 PM on April 1 with an INITIATE request for 30 MB.
2. Granted 40 MB with validity of 1 hour.
3. Reserves the balance USD 30.
4. When non-currency voucher top-up is done at 00:00 hours, which grants 10 MB of data with validity of 5 days, a RAR notification is sent.
5. The UPDATE request is received with requested units as 40 MB and used units as 12 MB at 00:02 on April 3.
6. A CDR is generated for 12 MB (23:50:00-00:00:00 April2 (10MB @1/MB) = 10$, 00:00:00-00:02:00 (10MB -2Mb) = -8MB).
7. Granted 40 MB with validity of 1 hour.

   Balances will be:
   - Current balance: USD 10
   - Free MB = -8MB
8. The TERMINATE request is received with Used units as 20 MB at 01:30 on April 3.

   Balances will be:
   - Current balance: USD 22
   - Free MB = 0

**Scenario: Recharge Consumed in a Session**

Given a subscriber with a charge offer named **Data Standard**, with a **Standard Bytes** (chargeable balance) followed by **Free Use Data** (counter), with a starting balance of 50 MB Standard Bytes and 0 bytes of **Free Use Data**.

Given a subscriber with:
A charge offer named *Data Standard*

A consumption cascade of *Standard Bytes* (chargeable balance) followed by *Free Use Data* (counter)

A starting balance of 50 MB Standard Bytes

0 bytes of *Free Use Data*

The usage charging flow is as follows:

1. The subscriber starts a data session at 1:00 PM
2. Consumes 50 MB over 20 minutes at high quality of service (QoS)
3. Consumes 2 MB over 5 minutes at low QoS
4. At 1:25 PM, recharges with 100 MB *Standard Bytes* valid for 30 minutes
5. Consumes 100 MB over 20 minutes at high QoS
6. Consumes 5 MB over 15 minutes at low QoS before terminating the session

Table 4–2 shows the subscriber’s balances and charges at the end of the session.

<table>
<thead>
<tr>
<th>Balance Name</th>
<th>Balance Value</th>
<th>Units Charged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Standard</td>
<td>0 MB</td>
<td>150 MB</td>
</tr>
<tr>
<td>Free Use Data</td>
<td>7 MB</td>
<td>7 MB</td>
</tr>
</tbody>
</table>

### Configuring Incremental Rating for Tariff Changes

ECE performs incremental rating of an active network session based on the tariff changes that occur during a session. You can configure ECE to generate a rated event whenever a tariff change occurs during a network session. The tariff change can include: peak and off-peak rate changes in offers, availability or expiry of charge offers, alteration offers (discount offer) or distribution offers (charge sharing offers), availability or expiry of customer balances.

The incremental rating for tariff changes enables ECE to rate long network sessions incrementally based on the exact data consumed between tariff changes. It also enables operators to show subscribers the running balance based on the actual data consumption after each tariff change.

You can enable incremental rating for tariff changes by doing the following:

- Configuring Incremental Rating for Midsession Rating Condition Changes
- Configuring ECE to Support Tariff Time Change

When this feature is enabled, ECE determines if there is a tariff change when authorizing and reserving a balance for a session request from the network. ECE bases the reservation on the requested service units of the session request and sends the Tariff-Time-Change AVP in the usage response to the network to record the exact data consumed before and after the tariff change. ECE also performs the reverse rating to calculate the amount of usage that the subscriber can afford and reserves the balance for the requested service units based on the worst-case charging condition (the maximum charge that can be applied for the requested service units). This ensures that the overall usage does not exceed the credit limit of a customer and there is no revenue leakage whether the balance is consumed before or after the tariff change. For more information on reverse rating, see *BRM Concepts*. 

For incremental rating, ECE supports only one tariff change for a session request. If more than one tariff change is determined during authorization, ECE considers the tariff change that occurs first for balance reservation and adjusts the validity time to expire at the next tariff change. This ensures that there is only one tariff change that occurs before the session expires. ECE then rates the exact balance consumed based on the rating condition changes before and after the tariff change and generates rated events each time a tariff change occurs in an ongoing session.

The following section describes a scenario of tariff change during a data session and the balances and charges applied during and at the end of the session.

**Scenario: Offer Becomes Valid and Consumed in a Session**

Given a subscriber with:
- A charge offer named *CO Data Standard* with a *Standard Bytes* (Chargeable Balance) followed by *Free Use Data* (Counter)
- A starting balance of 50 megabyte (MB) *Standard Bytes* and 0 Bytes *Free Use Data*
- A discount offer named *Data Boost* which is valid from 1:30 PM the same day (100 MB *Standard Bytes* valid from 1:30 PM for 30 minutes from the first use).

The usage charging flow is as follows:
- When the subscriber starts a data session at 1.00 PM
- Consumes 50 MB over 20 minutes at high QoS
- Consumes 2 MB over 10 minutes at low QoS
- The purchased *Data Boost* discount offer becomes valid at 1:30 PM
- Consumes 100 MB over 20 minutes at high QoS
- Consumes 5 MB over 15 min at low QoS before terminating the session

Subscriber’s balances and charges are calculated as follows:

<table>
<thead>
<tr>
<th>Balance Name</th>
<th>Balance Value</th>
<th>Units Charged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Standard</td>
<td>0 MB</td>
<td>150 MB</td>
</tr>
<tr>
<td>Free Use Data</td>
<td>7 MB</td>
<td>7 MB</td>
</tr>
</tbody>
</table>

**Configuring ECE to Support Tariff Time Change**

To configure ECE to support tariff time change:

---

**Note:** Configuring ECE to support tariff time change is a systemwide setting and it is applied irrespective of the product type. When configured, the tariff time change is considered for all the products for which incremental rating is enabled.

---

1. Access the ECE MBeans:
   - Log on to the driver machine, which is the machine on which you installed ECE.
   - Start the ECE charging servers (if they are not started).
   - Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
Configuring Reservation Expiration and Validity

Using Session Connect Time for Charging

By default, the session attempt time, which is the time the session is initiated, is used as the session start time for calculating charges for usage sessions. For example, when a customer initiates a call at 10:00:00AM and the call actually gets connected at 10:00:30 AM, 10:00:00 AM is considered as the call start time.

You can configure ECE to use the session connect time, which is the time the session actually begins, as the session start time for calculating charges for usage sessions.

To use the session connect time for calculating charges:

1. Access the ECE MBeans:
   a. Log on to the driver machine, which is the machine on which you installed ECE.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to start CohMgt = true in the ECE_home/oceceserver/config/eceTopology.conf file.

      The eceTopology.conf file also contains the host name and port number for the node.
   e. In the editor's MBean hierarchy, expand the ECE Configuration node.

2. Expand charging.server.
3. Expand Attributes.
4. Select the tariffTimeChangeSupported attribute and set the value to true.

Configuring Reservation Expiration and Validity

In an online session, the network sends the following to ECE:
Usage updates: Keep ECE informed about the balance impact of an event. In response, ECE tells the network if the balance is sufficient to continue the session, or if a reauthorization is needed.

Reauthorization requests: Request an extension of the session. In response, ECE determines whether the customer’s balance is sufficient, and if so, reauthorizes the call.

You can configure two system-wide settings to handle usage reports and reauthorization requests:

- In a usage response, ECE sends a reservation expiration time to the network to specify how long a session can continue before the client must send a usage report to ECE. If the network mediation client does not communicate the used service units (USU) within the reservation expiration, ECE considers the reserved balances to be available for subsequent session requests. The available reserved balances are cleaned up by housekeeping processes when the session terminates.

  You do not want the value of the reservation expiration time to be too low because it takes network activity to report usage. You also do not want the value to be too high because that increases the risk of revenue leakage if the customer uses up his balance before the reservation expiration time expires.

- In a usage response, ECE sends the reservation validity time to the network to specify how long a session can continue before the client must ask for a reauthorization for further usage.

  The validity time is set for each session and is reset whenever an interim request is received for the session. After the validity time for a session expires, any reserved balances are released and become available to other active sessions for the same charge offer.

To configure reservation expiration and validity:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

2. Expand the ECE Configuration node.

3. Expand charging.reservationConfig.

4. Expand Attributes.

5. Specify values for the following attributes:
   - validityTime: Enter the amount of time, in seconds, to use as the reservation validity. This specifies how long a session can continue before the client must ask for a reauthorization. The default value is 3600 (one hour).
   - reservationDuration: Enter the amount of time, in seconds, to use as the reservation expiration. This specifies how long a session can continue before the client must report the consumed usage to ECE. The default value is 3600 (one hour).
Configuring a Minimum Quantity for Reservation

You can configure a minimum quantity for reservation for charging events. Some charging events cannot be charged in fragments. For example, you cannot charge for half of an SMS message. In this case, you would set a minimum quantity reservation of 1 for charging an SMS event.

If the customer does not have enough balance to reserve the minimum quantity of the charging event, ECE tells the network that there is not enough in the balance to fulfill the request.

To configure the minimum quantity for reservation:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/occeserver/config/eeetopology.conf file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.
2. Expand the ECE Configuration node.
3. Expand charging.reservationConfig.
4. Expand Operations.
5. Select setMinAuthorizedQuota.
6. Specify values for the following parameters:
   - **productType**: Enter the name of the product for which you are setting a minimum quantity reservation. Enter the name as it is defined in the ECE request specification data (for example, VOICE or SMS).
   - **rum**: Enter the name of the attribute defined in the ECE request specification data.

   **Important**: Though the parameter name is rum, its value must be the attribute name specified in the REQUESTED_UNITS block of the request specification data, not the rateable usage metric (RUM) name. For example, if you send attribute INPUT_VOLUME in the usage request, enter INPUT_VOLUME as the rum attribute’s value.

   - **minAuthorizeQuota**: Enter the minimum amount of the specified unit that can be reserved for this product-RUM combination.
   - **unit**: Enter the unit of measurement for the quota, such as seconds, minutes, events, or megabytes.
7. Click the setMinAuthorizedQuota button.
Configuring Reservation Quota for Services

When ECE receives a usage request that does not specify a requested amount, ECE uses a default usage amount. You configure a systemwide initial quota and a systemwide incremental quota for each combination of service and RUM. When initiating a call, ECE applies the initial quota for the reservation. For update requests, ECE applies the incremental quota for the reservation.

To configure the reservation quota for services:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

2. Expand the ECE Configuration node.

3. Expand charging.reservationConfig.

4. Expand Operations.

5. For each service that you offer, do the following:
   a. Select setDefaultReservationQuota.
   b. Specify values for the following parameters:
      - **productType**: Enter the name of the product defined in the ECE request specification data (for example, VOICE or SMS).
      - **rum**: Enter the name of the attribute defined in the ECE request specification data.

      **Important**: Though the parameter name is rum, its value must be the attribute name specified in the REQUESTED_UNITS block of the request specification data, not the RUM name. For example, if you send attribute INPUT_VOLUME in the usage request, enter INPUT_VOLUME as the rum attribute’s value.

      - **initialQuota**: Enter the initial quota for this service-RUM combination. The value must be decimal-compliant (Java BigDecimal). ECE uses this value to populate the REQUESTED_UNITS blocks of all Initiate-type usage requests whose Requested-Service-Units AVP value is missing.
      - **incrementalQuota**: Enter the incremental quota for this service-RUM combination. The value must be decimal-compliant (Java BigDecimal). ECE uses this value to populate the REQUESTED_UNITS blocks of all Update-type usage requests whose Requested-Service-Units AVP value is missing.
      - **unit**: Enter the unit of measurement for the quota, such as seconds, minutes, events, or megabytes.

   c. Click the setDefaultReservationQuota button.
Managing Dynamic Quotas for Online Sessions

Dynamic quota allows you to allocate the available quota dynamically for each parallel session of a subscriber based on the rules you configure in Pricing Design Center (PDC). For configuring dynamic quota selectors, see *BRM PDC Product Offerings*.

When ECE receives the usage requests from the network in which the requested service unit (RSU) is not set, it evaluates and applies the rules from the dynamic quota selectors on the usage request to derive the quota to be allocated and the quota attributes, such as:

- **Quota holding time.** Specifies how long a granted quota can be idle before the reservation is released.
- **Volume quota threshold.** Specifies how much of the granted quota must be consumed before a subscriber can request additional quota. This attribute is configured per service, event, and number of granted units.
- **Validity time.** Specifies whether validity time can be set to a fixed value per service-event combination at runtime. This attribute is independent of the number of units in the granted quota.

If dynamic quota selector rules are not configured for a service-event combination, ECE uses the default quota configuration for deriving the quota and the quota attributes. ECE returns the derived quota (as granted service unit (GSU)), quota holding time, and volume quota threshold values in the usage response to the network.

You can also customize the dynamic quota allocation to suit your business requirements. For more information, see "Sample Extensions".

Triggering RAR Notifications for Ongoing Sessions

When you use dynamic quotas for long running sessions to reduce network signaling, you can trigger server-initiated reauthorization requests to get the exact reservation balance before performing other business operations.

To generate server-initiated reauthorization requests, you must generate RAR notifications. To generate these notifications, you can implement custom logic by using the following ECE extensions in the rating/charging flow:

- Pre-rating extension
- Post-rating extension
- Post-charging extension

For more information, see "Rating/Charging Extension - Triggering RAR Notifications".

Configuring Rounding for a Resource

By default, ECE uses the rounding rules configured in Pricing Design Center (PDC) for a currency or noncurrency resource to round the balance impact amount for processing stages like charging, discounting, and taxation. These rules can be different for each processing stage. For information on configuring the rounding rules for resources in PDC, see the PDC Help.

However, you can configure systemwide rounding in ECE for currency and noncurrency resources to apply the rule across all processing stages.
Example of Currency Rounding for a Charge
If you allow two digits to the right of the decimal point and you round down towards zero (DOWN rounding mode), ECE takes a calculated charge of 0.509 USD and rounds it to 0.50 USDs.

Example of Noncurrency Rounding for a Charge
If you allow zero digits to the right of the decimal point and you round towards positive infinity (UP rounding mode), ECE takes a charge of 0.509 bonus point and rounds the value to 1 bonus point.

Examples of Currency Rounding for Discounts
If you allow zero digits to the right of the decimal point and you round down towards zero (DOWN rounding mode), ECE takes a discount of -2.5 USD and rounds the value to -2 USD.

If you allow zero digits to the right of the decimal point and you round towards negative infinity (FLOOR rounding mode), ECE takes a discount of -2.5 USD and rounds the value to -3 USD.

If you allow two digits to the right of the decimal point and you round down towards zero (DOWN rounding mode), ECE takes a discount of -0.075 USD and rounds the value to -0.07 USD.

To configure rounding for a resource:
1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to start CohMgt = true in the ECE_home/occeserver/config/eceTopology.conf file.
      The eceTopology.conf file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.
2. Expand charging.server.
3. Expand Attributes.
4. Specify values for the following currency and noncurrency resource attributes as appropriate:
   - currencyScale or nonCurrencyScale: Enter the number of digits you allow to the right of the decimal point for a calculated impact amount.
     For example, enter 2 if you allow two digits to the right of the decimal point.
     The default is 2.
   - currencyRoundingMode or nonCurrencyRoundingMode: Enter the rounding mode that determines the rounding behavior by entering the string representation of the Java math rounding enum.
     For more information, see the Java SE technical documentation web site:
     http://docs.oracle.com/javase/6/docs/api/java/math/RoundingMode.htm
Enabling Server-Initiated Reauthorization Requests

For example, enter UP to round up away from zero or DOWN to round down towards zero.

The default value is HALF_UP.

**Configuring Rounding for Reverse Rating on Multiple RUMs**

When ECE performs the reverse rating service in which events are rated by using multiple RUMs, fractional values may result for the authorized resource. You can configure a systemwide rounding rule to round up the fractional value of the authorized resource.

Rounding up the authorized resource quantity may result in customers exceeding their credit limits. Configure this only if your business requires that your customers must be able to use all of their balances.

To configure whether to round up the fractional value of the authorized resource quantity by authorizing an additional RUM unit:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to start CohMgt = true in the ECE_home/occeserver/config/eceTopology.conf file.

      The eceTopology.conf file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

2. Expand charging.server.

3. Expand Attributes.

4. Set the reverseRateUseAllBalances attribute to one of the following values:
   - To round up the fractional value of the authorized balance quantity, enter true.
     This option allows customers to use all balances even if they might exceed their credit limits by a small amount.
   - To disallow the fractional value of the authorized balance quantity to be rounded up, enter false.
     This option does not allow customers to exceed their credit limits.

   The default is false.

**Enabling Server-Initiated Reauthorization Requests**

ECE can perform server-initiated reauthorization requests (RAR) during an ongoing session. This enables you to update a session in response to changes that occur to a customer’s product offerings or balance (for example, a change to a charge offer or to a Friends and Family promotion). When ECE notifies the network, the network sends a reauthorization request, and if there is a change in the charge, ECE can base the reauthorization on the new charge.

A server-initiated reauthorization can be triggered from the following conditions:
Enabling Server-Initiated Reauthorization Requests

- Changes to offers, such as the creation, modification, or deletion of a subscriber’s charge offer or alteration offer.
- Changes to balances that affect rating (for example, a balance that expires mid-session, a balance that becomes available from a top-up, or changes to the customer balance due to an accounts receivable action).
- Changes to promotions, such as changes to Friends and Family or a Special Day offer.
- Changes to charge sharing or alteration sharing groups. For example, a new member is added to the group or a member is removed mid-session.

For example:
1. A subscriber is in a call session. The subscriber adds the called number of that session to a Friends and Family list.
2. Because a Friends and Family discount might change the charge amount, ECE sends a request to the network.
3. In response, the network sends a reauthorization request.
4. ECE sends a reauthorization, using the Friends and Family charge amount.

---

**Note:** A reauthorization request is not triggered by a top-up or by rerating when balances are added to a sharing group owner’s account.

To enable server-initiated reauthorization requests:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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_homeoceceserver/config/eceTopology.conf` file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.
4. Expand **Attributes**.
5. Set the **rarNotificationMode** attribute to **ASYNCRONOUS**.
   This enables RAR notifications, which are required for server-initiated reauthorization requests. ECE generates an external notification and sends it to a notification queue (JMS topic) when the **RAR_NOTIFICATION_EVENT** service event is created. When specific condition changes occur during a session, ECE generates a RAR notification to inform the network to request a reauthorization.
6. Under the **ECE Configuration** node, expand **charging.server**.
7. Expand **Attributes**.
8. Set the **offerEligibilitySelectionMode** attribute to **PERIOD**.
In **PERIOD** mode, ECE selects applicable charge offers valid any time between the start and end time of the session to determine charges for events. You use this mode when implementing server-initiated reauthorization requests so that ECE can rate based on changes to a customer’s subscription, such as the purchase of a promotional offer, during the session.

By default, this parameter is set to **END_TIME**.

In **END_TIME** mode, ECE selects charge offers valid at the end time of the session to determine charges for events. **END_TIME** mode must be used when using a version of BRM that does not support **PERIOD** mode.

---

**Note:** Events rated in **PERIOD** mode might result in a different charge from the charge calculated when the event is rerated. This happens because the event is rerated using only the pricing applicable at the event end time.

---

### Configuring Systemwide Consumption Rules for Balances

When more than one validity-based subbalance is available for a usage request, consumption rules determine from which balance bucket ECE is to consume first. For example, if a customer has several groups of free minutes that expire at different times, you use consumption rules to indicate which minutes to use first, based on the validity period start time and end time. Consumption rules are typically configured at the balance element level when you define pricing in the pricing application such as PDC. Consumption rules can also be configured at the customer balance level by the customer and subscription management components of the BRM system. For information about configuring consumption rules in PDC and BRM, see the PDC documentation and the BRM documentation.

When ECE receives a usage request for which no consumption rules are configured, ECE applies its own systemwide consumption rules for processing the usage request.

To configure ECE systemwide consumption rules:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceceserver/config/eceTopology.conf` file.
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor's MBean hierarchy, expand the **ECE Configuration** node.

2. Expand **charging.server**.

3. Expand **Attributes**.

4. Set the **systemConsumptionRule** attribute to one of the following systemwide consumption rules:
   - **EARLIEST_START**
   - **LATEST_START**
   - **EARLIEST_EXPIRATION**
Configuring ECE to Return Remaining-Balance Information in Usage Responses

You can configure ECE to return the customer’s remaining-balance information in the usage response (as an in-session notification). For example, you could use the information to send customers a low-balance notification when they are about to use up all of their available balance for a service or they reach a balance amount set in your system to trigger such notifications.

ECE sends remaining-balance information for initiate and update usage requests.

The remaining-balance information that ECE returns pertains to all balances impacted by the session (that is, the balances to which the session applied balance impacts).

For charge distribution scenarios (charge sharing), ECE returns the remaining-balance information for the balances impacted by the sharer’s usage.

To configure ECE to return remaining-balance information in usage responses:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/oceseserver/config/eceTopology.conf` file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.
2. Expand the ECE Configuration node.
3. Expand charging.server.
4. Expand Attributes.
5. Set the `remainingBalanceCalcMode` attribute to one of the following values:
   - NONE: (Default) Sends no remaining-balance information in usage responses. ECE does not calculate the remaining balance.
■ **CURRENT_BALANCE**: Sends remaining-balance information for the current balance, excluding the credit limit, in the usage response. Use this option to notify your customers of their plain vanilla remaining balance.

ECE calculates the remaining balance by adding all sub-balances valid for the session, including the consumed reserved amount of ongoing sessions. The remaining balance is calculated as follows:

\[
\text{remaining balance} = \text{sum of for valid sub-balances of (current balance + consumed reserved amount)}
\]

■ **UPTO_CREDIT_LIMIT**: Sends remaining-balance information capped at the credit limit in the usage response. Use this option to notify your customers of the credit limit up to which you allow them to use the balance.

ECE calculates the remaining balance by adding all sub-balances valid for the session, including the consumed reserved amount of ongoing sessions (the consumed reservation of the balances ECE reserved for ongoing sessions) and subtracts that value from the credit limit.

ECE calculates the remaining balance as follows:

\[
\text{remaining balance} = (\text{credit limit} - \text{sum of for valid sub-balances of (current balance + consumed reserved amount)})
\]

---

**Configuring Taxation**

ECE supports a fixed rate tax (a flat-rate taxation which is also known as GST or VAT). You can apply a tax on both charges and alterations (discounts).

To configure taxation:

---

**Note**: For taxation to work in the ECE runtime environment, you must set the following mandatory parameters.

---

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceserver/config/eceTopology.conf` file.
      
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the **ECE Configuration** node.

2. Expand **charging.taxation**.

3. Expand **Operations**.

4. Click **addTaxDetails**.

5. Specify values for the following parameters:
Important: These parameters are mandatory. You must set all of them when configuring taxation.

- **taxCode**: Enter the tax code used by the charge offer or discount offer to which the tax applies.
  
The tax code is used by charge offers and discount offers to point to the tax rate that must be applied when a usage request is processed for the charge offer or discount offer.
  
Enter the same tax code entered in PDC when the taxation section of the charge offer and discount offer was defined.

- **taxRate**: Enter the tax rate to apply.
  
  For example, entering 0.20 applies a 20% tax on the total usage impact.

- **taxGlId**: Enter the General Ledger ID used for the tax impact.

6. Specify an additional **taxCode**, **taxRate**, and **taxGlId** value for each charge offer or discount offer to which a tax applies.

---

**Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers**

When you design your pricing components in Pricing Design Center, you can create charges for which conditional balance impacts are configured.

You can configure a runtime option in ECE that aligns the validity start of a conditional balance impact with the validity start of the associated purchased charge offer and aligns the validity end of a conditional balance impact with the validity end of the associated purchased charge offer. For example, if a customer activates a conditional balance impact valid for three days and the charge offer with which it was purchased is not valid after one day, this configuration specifies whether the conditional balance impact can still be used after the charge offer validity has ended. If ECE does not align the validity end of the conditional balance impact with the validity end of the charge offer the customer purchased, the balance can be used by another charge offer.

To configure ECE to align validity start and end of conditional balance impacts and charge offers:

1. Access the ECE configuration MBeans:
   
a. Log on to the driver machine.

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

c. 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/oceserver/config/eceTopology.conf` file.

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

   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

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

3. Expand **charging.server**.

4. Expand **Attributes**.
5. Set the **alignRecurringImpactsToOffer** attribute to **true**.

At run time, if this is set to **true** and ECE receives a usage request for which a conditional balance impact applies, ECE compares the validity start and end of the conditional balance impact with the usage validity start and end of the associated charge offer that the customer purchased. If the validity start or end of the conditional balance impact breaches the validity start or end of the associated purchased charge offer, ECE aligns both the validity start and end of the conditional balance impact with those of the charge offer.

### Configuring How ECE Manages Active Sessions When Network Elements Fail

When a network element associated with active sessions in ECE fails, ECE receives an accounting on/off request from the network element. You can configure ECE to cancel or terminate active sessions when processing accounting on/off requests.

To configure how ECE manages active sessions when network elements fail:

1. Access the ECE configuration MBeans:
   - Log on to the driver machine.
   - Start the ECE charging servers (if they are not started).
   - 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/occeserver/config/ceceTopology.conf` file.
   - Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   - In the editor’s MBean hierarchy, find the ECE configuration MBeans.

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

3. Expand **charging.server**.

4. Expand **Attributes**.

5. Set the **accountingOnOffMode** attribute to one of the following values:
   - **TERMINATE**: Active sessions that have a state of Initiated are terminated when an accounting on/off request is processed.
   - **CANCEL**: Active sessions in ECE that have a state of Initiated are canceled when an accounting on/off request is processed.

### Configuring ECE to Redirect Subscriber Sessions to a Service Portal

Service providers can redirect a subscriber session to a service portal, a server outside of the online charging system, where specific services can be offered to the subscriber. During an online charging session, if a subscriber is about to deplete funds for the use of a service, the subscriber can be redirected to a web site to top up the account. You can configure ECE to send service portal addresses back to credit-control clients.

Credit-control clients use the information for redirecting a subscriber session to the service portal applicable to the business scenario.

ECE derives the service portal address (to send back to credit-control clients) based on configurable instructions that you define in **redirection rules**. Your redirection rules can be based on any of the following customer conditions (typically based on a combination of them):
Whether the customer has insufficient funds
- Whether the customer has an inactive account
- Whether the customer is roaming or not roaming
- Whether the customer belongs to a specific customer segment (for example, customer accounts associated with a BRM business profile for which the payment type is Prepaid or Postpaid or the subscription type is Voice or Data.)

Each redirection rule can send the session to a different service portal.

For example, you might configure two redirection rules for the following business scenarios:
- Given a customer with an account using a prepaid payment type who is roaming, redirect the subscriber to http://myPrePaidRoamingRedirect.com.
- Given a customer with an account using a prepaid payment type who is not roaming, redirect the subscriber to the http://myPrePaidHomeNetworkRedirect.com URL address.

After ECE derives the service portal addresses and address types based on your redirection rules, ECE sends the address back to the credit-control client.

When the credit-control client receives the Final-Unit-Indication in the answer from ECE, the credit-control client behavior depends on the value, TERMINATE or REDIRECT, indicated in the Final-Unit-Action AVP. If you do not configure redirection rules in ECE, then ECE indicates a Final-Unit-Action of TERMINATE in the usage response.

To configure ECE to redirect subscriber sessions to a service portal:
1. Create your redirection rules in a text editor and save the file.
   - If you have multiple redirection rules, you must separate them by semicolons and save them as a single line. The single-lined redirection configuration should contain all of the redirection rules for the business scenarios that require redirecting subscriber sessions to applicable service portals.
   - See “Creating Redirection Rules”.

2. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/oceseserver/config/ectopology.conf file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

3. Expand the ECE Configuration node.
4. Expand charging.redirectionConfiguration.
5. Expand Attributes.
6. Set the redirectionRule attribute to a copy of your redirection-rule configuration.
   - The default value is an empty string.
   - If no rule is provided, no redirection is done. ECE terminates the session.
ECE begins using the redirection-rule configuration at runtime.

If your redirection rule uses incorrect syntax, ECE logs the **Rule Evaluation Failed** error at runtime in the charging-server node log files (ecs log files) and leaves the redirection rule field in the usage response empty.

---

**Tip:** Modifying a redirection-rule configuration in JConsole may be error prone because you cannot see the entire rule. Modifying a redirection-rule configuration in the file where you created it is recommended. Pressing Ctrl + A in the Value column of the `redirectionRule` variable selects all contents.

---

**Creating Redirection Rules**

A redirection rule contains conditions that must be met for the subscriber session to be redirected to a service portal.

Your redirection configuration might contain a Voice redirection rule and a Data redirection rule for redirecting subscribers to service portals relevant to those services.

You must use allowed redirection-rule conditions.

The following scenario:

When the customer is roaming
The redirect address is http://RedirectRoaming.com
The redirect address type is URL

Is redirected by using the following redirection rule:

```
'{{ (@fui AND @roamingRequest) => [redirect_type:"URL",redirect_address:
'http://RedirectRoaming.com'];
```

The following scenario:

When the customer is Postpaid
And the customer is roaming
The redirect address is http://RedirectRoaming.com
The redirect address type is URL
The redirection must be performed within 900 seconds

Is redirected by using the following redirection rule:

```
'{{ (@fui AND ((business_profile({name:"POSTPAID"})}) == "true" }} AND @roamingRequest) => [redirect_type:"URL",redirect_address:
'http://RedirectRoaming.com',redirect_validity:"900"]
```

**Table 4–4** shows redirection-rule conditions that you can use to create redirection rules.
<table>
<thead>
<tr>
<th>ECE Redirection-Rule Conditions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@fui</td>
<td>Checks in the charging result if the customer has insufficient funds (finds the Final Unit Indicator in the service context). @fui is required.</td>
</tr>
<tr>
<td>{business_profile([name:&quot;BusinessProfileName&quot;])} == &quot;true&quot; )</td>
<td>Accesses a business profile by looking up a business profile name and comparing its value to true. Valid values for BusinessProfileName are names of attributes you defined in the attribute-value pairs of your BRM business profiles.</td>
</tr>
</tbody>
</table>
| @roamingRequest                                     | Checks if the request is for a customer who is roaming.  
@roamingRequest denotes roaming.  
!@roamingRequest denotes not roaming.  
The check is done on the value of the following Diameter credit-control-request fields:  
- GGSN-MCC-MNC-3GPP  
- IMSI-MCC-MNC-3GPP  
**Note:** These fields are not provisioned in ready-to-use event definitions. You must provision these network fields when you enrich your event definitions in PDC. |
| {request_attribute([name:"FieldName"])}           | Reads a payload field from a usage request. For example, the following condition reads the simple attribute 3GPP-IMSI-MCC-MNC from the payload of the usage request:  
{request_attribute([name:"3GPP-IMSI-MCC-MNC"])}  
Use this construct to use any request attribute field as a condition in your redirection rule. For example, if you want subscribers to be directed to a different URL if they have a 1234 cell phone ID, you might use the condition:  
{request_attribute([name:"CELL_ID"])} == "1234" |
| @productType                                        | Retrieves the service. For example, a redirection rule using this condition:  
( @productType == 'DATA'  
  AND  
  ( {request_attribute([name:"GGSN-MCC-MNC-3GPP"])} == "1234" ) ) => [redirect_type:"URL",redirect_address:"myDataTopUpRedirect.com"] |
Redirection-Rule-Configuration Syntax

You configure one or multiple redirection rules in a single-lined redirection configuration with each redirection rule separated by semicolons.

The syntax for a redirection rule is the following:

```
((redirection_condition AND redirection_condition) AND redirection_condition) =>
[redirect_type:"redirect_type",redirect_address:"redirect_address",redirect_validity:"redirect_validity"];
```

where:

- **redirection_condition** is a condition that must be met for ECE to send the specified redirect type, redirect address, and redirect validity in the ECE usage response. See Table 4–4 for accepted redirection-rule conditions.
- **redirect_type** is the type of the service portal address (for example, URL)
- **redirect_address** is the service portal address (for example, a web site address)
- **redirect_validity** is the time, in seconds, that the subscriber being redirected has to complete the task that must be done at the service portal. The value you enter here overrides the default reservation validity time of ECE. If you do not specify a redirect validity in your reservation rule, then the default reservation validity time of ECE is sent back to the credit-control client.

When you design your redirection rules, it can be helpful to create a user scenario for each and show the translation in a table, as shown in the following examples.

Example Redirection Rules

The following is an example of redirection rules.

Tip: For visual clarity, this example shows a carriage return after each redirection rule. Your redirection-rule configuration would be one line comprised of these four redirection rules separated only by semicolons.

```
"{( @fui AND ((business_profile([name:"POSTPAID"])) == "true" )) AND @roamingRequest) =>
[redirect_type:"URL",redirect_address:"http://myPostPaidRoamingRedirect.com",redirect_validity:"900"];

( @fui AND ((business_profile([name:"POSTPAID"])) == "true" )) AND !@roamingRequest) =>
[redirect_type:"URL",redirect_address:"http://myPostPaidHomeNetworkRedirect.com",redirect_validity:"900"];

( @fui AND ((business_profile([name:"PREPAID"])) == "true" )) AND @roamingRequest) => [redirect_type:"URL",redirect_address:"http://myPrePaidRoamingRedirect.com"];

( @fui AND ((business_profile([name:"PREPAID"])) == "true" )) AND !@roamingRequest) =>
[redirect_type:"URL",redirect_address:"http://myPrePaidHomeNetworkRedirect.com"]"
```

The four redirection rules support redirecting subscribers who have depleted funds in their account to a service portal for these scenarios:

- Given a subscriber with an account using a postpaid payment type who is roaming, redirect the subscriber to the [http://myPostPaidRoamingRedirect.com](http://myPostPaidRoamingRedirect.com) URL address and allow the subscriber to use network resources for 900 seconds.
- Given a subscriber with an account using a postpaid payment type who is not roaming, redirect the subscriber to the
http://myPostPaidHomeNetworkRedirect.com URL address and allow the subscriber to use network resources for 900 seconds.

- Given a subscriber with an account using a prepaid payment type who is roaming, redirect the subscriber to the http://myPrePaidRoamingRedirect.com URL address.
- Given a subscriber with an account using a prepaid payment type who is not roaming, redirect the subscriber to the http://myPrePaidHomeNetworkRedirect.com URL address.

### Enabling Match Factor in ECE

ECE supports match factor in discounting.

To enable match factor in ECE:

1. Access the ECE MBeans:
   a. Log on to the driver machine, which is the machine on which you installed ECE.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/occeserver/config/eceTopology.conf` file.
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

2. Expand charging.server.

3. Expand Attributes.

4. Set the `matchFactorEnabled` attribute to `true`. 
Configuring Subscriber Preferences

This document describes how to configure subscriber preferences in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE), for example how to receive notifications from the network.

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

Configuring Subscriber Preferences

BRM enables you to manage how each subscriber prefers to receive notifications from the network. For example, you can specify that a subscriber wants to receive notifications in French via SMS text messages.

By default, BRM enables you to manage the following subscriber preferences:
- Preferred channel of communication: IVR, SMS, e-mail, and so on
- Preferred language of communication: English, French, and so on
- Number of days prior to which customer wishes to receive the notification
- Interval between two successive notifications
- Timestamp of the last notification sent to the subscriber

BRM stores information about each subscriber’s preferences in a subscriber profile repository. BRM stores the types of preferences that you track and their default values in the /config/subscriber_preferences object. BRM stores each subscriber’s preferences at the account level and the service level in individual /profile/subscriber_preferences objects.

For more information on the /config/subscriber_preferences and /profile/subscriber_preferences objects, see BRM Storable Class Reference.

Maintaining Subscriber Preferences with Customer Center

When in-session notifications are enabled, you can configure and maintain subscriber preferences by using Customer Center. During the account creation and modification process, you specify the subscriber preferences in the Customer Center Subscriber Preferences page.
Customer Center uses the configurations in the `/config/subscriber_preferences_map` object to dynamically list the preferences that a subscriber can configure. You can customize the information as necessary.

**About Regulating Permissions to Update Subscriber Preferences**

By default, all customer service representatives (CSRs) can access and update subscriber preferences. You can restrict a CSR’s permissions to view and update a subscriber’s preferences for services and accounts.

**Configuring ECE to Enrich External Notifications with Subscriber Preference Information**

You can configure ECE to enrich external notifications with subscriber preference information.

BRM enables you to manage how each subscriber prefers to receive notifications from the network. For example, you can specify that a subscriber wants to receive notifications in French (Language preference) via SMS text messages (Channel preference). All subscriber preferences set for customers in BRM are also stored in ECE.

You can configure ECE to enrich the following types of ECE external notifications with subscriber preference information:

- Threshold breach notifications
- Aggregated threshold breach notifications
- Advice of Charge notifications
- Credit limit ceiling breach notifications
- Credit limit floor breach notifications
- Subscriber life cycle state transition notifications
- First usage validity initialization notifications

You can configure ECE to enrich each of the preceding external notifications with all subscriber preferences or with a subset of subscriber preferences.

If the same subscriber preference is defined as a customer preference and as a service preference, ECE uses the service preference. If a subscriber preference is not specified for the service but is specified for the customer, ECE uses the customer subscriber preference.

To configure ECE to enrich external notifications with subscriber preference information:

1. If you do not have it, obtain the list of subscriber preference names you have set in your BRM system.
   
   When configuring ECE to enrich the external notifications with a subset of subscriber preferences, you must enter the name of the subscriber preference(s) as you previously set it in your BRM system.

2. Access the ECE configuration MBeans:
   
   a. Log on to the driver machine.
   
   b. Start the ECE charging servers (if they are not started).
   
   c. 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.

d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
e. In the editor's MBean hierarchy, find the ECE configuration MBeans.

3. Expand the ECE Configuration node.

4. Expand charging.notification.

5. Expand Attributes.

6. Set the subscriberPreferenceUpdateNotificationMode attribute to ASYNCHRONOUS.

7. Select a notification type for which notification messages are to be enriched with subscriber preference information.

8. Specify values for the following attributes:

   ■ enrichName: Enter subscriberPreferences.
   ■ enrichValue: Enter one of the following values:

      No value: (Default) External notifications are not enriched with subscriber preferences.

      Individual subscriber preferences: External notifications are enriched with a subset of subscriber preferences. Enter the name of each preference, separated by commas. The names must match the preference names set in your BRM system.

         ALL: External notifications are enriched with all the customer's subscriber preferences.

For each notification type enabled to be enriched with subscriber preference information, ECE publishes subscriber preference information in the SubscriberPreferences block of the external notification messages.

The following is an example of the SubscriberPreferences block for a threshold breach notification enriched with the language subscriber preference of the customer.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
   <NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
   <PublicUserIdities>
      <PublicUserIdentity>6500000001</PublicUserIdentity>
   </PublicUserIdities>
   <BalanceElementId>840</BalanceElementId>
   <BalanceElementCode>USD</BalanceElementCode>
   <CurrentBalance>-3.00</CurrentBalance>
   <ThresholdAmount>-4</ThresholdAmount>
   <ThresholdPercent>98.0</ThresholdPercent>
   <BreachDirection>THRESHOLD_BREACH_UP</BreachDirection>
   <DuplicateEvent>False</DuplicateEvent>
   <SubscriberPreferences>
      <SubscriberPreference PublicUserIdentity="6500000001:VOICE">
         <SubscriberPreferencesInfo>
            <PreferenceName>Language</PreferenceName>
            <PreferenceValue>French</PreferenceValue>
         </SubscriberPreferencesInfo>
      </SubscriberPreference>
   </SubscriberPreferences>
</Notification>
```
Customizing Subscriber Preferences

To customize the subscriber profile data configuration by using config_subscriber_preferences_map.xml file:

1. Open the BRM_home/sys/data/config/config_subscriber_preferences_map.xml file.
2. Edit the file, which includes examples and instructions. Table 5–1 describes the parameters in the file:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the preference</td>
</tr>
<tr>
<td>ID</td>
<td>The ID associated with the preference</td>
</tr>
</tbody>
</table>
| Type        | The type of value that the preference can be assigned, from one of the following types:  
  ■ 1: STR (alphanumeric)  
  ■ 2: INT (integer)  
  ■ 3: ENUM (indicating that the preference is one of an ordered list of possible values. An array of values must be provided for this selection.) See "Values".  
  ■ 4: DECIMAL  
  ■ 5: TSTAMP (timestamp)  
  For example, to provide an set of possible values, you set Type to 3, and enter an array of values for this preference in Values. See "Values". |
| String ID   | Used for Localization. The ID in the /string storable class that would be associated with the localized string associated with the preference. Customer Center uses this information to display the preference name in a localized string form. |
| Default     | The field containing the default value the preference is to be assigned. |
| Values      | An array list of values that the preference can assume. The Values array list is present only if the selection for Type is ENUM. See "Type". |

3. For example, the following entry defines a new preference type called Subscription Level as the tenth preference for subscribers:

```xml
<SUBSCRIBER_PREFERENCES elem="10">
  <NAME>Subscription Level</NAME>
  <SUBSCRIBER_PREFERENCE_ID>10</SUBSCRIBER_PREFERENCE_ID>
  <STRING_ID>10</STRING_ID>
  <DEFAULT>Silver</DEFAULT>
  <TYPE>3</TYPE>
  <VALUES elem="0">
    <VALUE>Silver</VALUE>
  </VALUES>
  <VALUES elem="1">
    <VALUE>Gold</VALUE>
  </VALUES>
  <VALUES elem="2">
    <VALUE>Platinum</VALUE>
  </VALUES>
</SUBSCRIBER_PREFERENCES>
```
In this example:

- The **Name** of the preference is *Subscription Level*.
- The subscriber preference ID for the language preference is 10.
- The string ID for the localizing string is 10.
- The default value for the language preference is *Silver*.
- The type of value is 3 (which is ENUM, and so an array of values follows).
- The **Values** array lists the 3 possible subscription level selections: *Silver*, *Gold*, and *Platinum*.

4. Save the `config_subscriber_preferences_map.xml` file.
5. Open the `BRM_home/apps/load_config/pin.conf` file in a text editor.
6. Add the following as the last entry:
   ```
   - load_config validation_module libLoadValidTCF AAA LoadValidTelcoAAA_init
   ```
7. Save the `pin.conf` file.
8. Load the updated file by running the `load_config` utility:
   ```
   load_config config_subscriber_preferences_map.xml
   ```

---

**Important:**

- The `load_config` utility requires a configuration (`pin.conf`) file.
- If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file. For example,
  ```
  load_config BRM_home/sys/data/config/config_subscriber_preferences_map.xml
  ```

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

---

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

To verify that the updated preference configurations were loaded, you can display the `/config/subscriber_preferences_map` object by using the Object Browser, or use the `robj` command with the `testnap` utility.

For more information on the `/config/subscriber_preferences_map` object, see *BRM Storable Class Reference*.
This document describes how to configure top-ups in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:
- Integrating Top-Up clients
- Detecting Duplicate Top-Up Requests
- Top-up API

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client
- Configuring Balance Queries

**Integrating Top-Up clients**

ECE interfaces directly with top-up systems to manage balances. The top-up systems send the top-up amount to ECE. ECE updates the customer balance and sends the update to the BRM server.

ECE does not manage top-up authentication, authorization, or accounting operations.

For information about running sample programs that demonstrate how to use the ECE top-up API, see "ECE Sample Programs".

To configure top-up notifications see "About ECE Notifications" and "Charging Notifications".

**Detecting Duplicate Top-Up Requests**

Duplicate top-up requests might occur within ECE as a result of node failures (for example, if an ECE server unexpectedly shuts down, ECE might internally resubmit a top-up request when restarted). To detect and eliminate these internal duplicate top-up requests, ECE maintains a top-up history cache.
Note: Third-party top-up systems, such as voucher management systems, are expected to eliminate duplicate top-up requests coming from the network.

The ECE top-up history cache maintains a specified number of top-up message IDs for each customer. If the message ID of an incoming top-up request is already in the history, ECE considers the request to be a duplicate.

If ECE detects a duplicate top-up request, the following occurs:

- ECE does not apply the top up
- ECE includes the following in the top-up response message:
  - The reason code DUPLICATE_REQUEST
  - The current customer balance

If you do not retain a sufficient number of top-up request message IDs in your top-up history cache, ECE may not detect internal duplicate top-up requests. If ECE cannot detect an internal duplicate top-up request, the following occurs:

- ECE applies the top-up balance to the customer balance as though it were a new request and sends the top-up balance to the BRM server.
- The BRM server, which stores all top-up message IDs in the BRM database, detects the top-up request as a duplicate and does not apply the top up to the customer balance in the BRM database.
- BRM adds the Error from BRM: ERR_DUPLICATE error to the BRM Gateway log file.

You must manually track such errors in the BRM Gateway log file and correct the customer balance in ECE because the ECE customer balance is no longer synchronized with the customer balance in the BRM database.

To configure the number of top-up request message IDs in your top-up history cache:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/eeetopology.conf file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor's MBean hierarchy, find the ECE configuration MBeans.
2. Expand the ECE Configuration node.
3. Expand charging.externalTopUpConfig.
4. Expand Attributes.
5. Set the topUpHistoryCount attribute to the number of top-up request message IDs to store for each customer.

   The default value is 3.

   If you change the value of this attribute (for example, from 10 to 4) and the top-up history cache already contains 10 message IDs, ECE eliminates the message IDs of
the oldest 6 top ups when the next top up arrives so that only the message IDs of the 4 most recent top-up requests are stored.

**Top-up API**

To use the top-up API, clients call the submitUpdate API with ExternalTopupUpdateRequest.

Consider the following points for how the ECE top-up API validates top-up requests:

- For a currency balance, validity should not be passed as part of the request; if it is passed, it fails with an error. The customer’s balance is expected to have one valid currency balance item/bucket with infinite validity. If no balance item/bucket is present, a new one is created with infinite validity.

- Validity extend (ValidityExtend) is only for noncurrency balances (given that currency balances do not have a validity).

- Whenever validity must be set for a balance, both validity start and validity end must be sent as part of the request.

  To create bucket with infinite validity, set both validityStart and validityEnd to -1.

- If the request is to extend validity and the customer balance has multiple valid balance items/buckets, then an error response is sent.

- If the request is to create a firstUsage bucket, then validity start and validity end should not be set in the request, except for the FirstUsageValidityUnit.ABSOLUTE mode.

- ValidityExtend is not allowed on a first-usage bucket.

- If both the validity and first-usage information (such as offset and unit) are specified as part of the request, then the top-up request fails with an error.

- Top-ups from Third-party top-up systems are not allowed when ECE is in a short-lived phase of the rerating process called the CATCH_UP phase.

  If top-up requests are sent during the CATCH_UP phase of rerating, ECE sends a response that includes the reason code for the failure. If that occurs, you can resend the top-up, and ECE will process it.

- During testing, if a top-up request is sent to ECE with an event time that is earlier than the account creation time of the account to which the top-up applies, the balance is updated with the top-up in ECE but the balance is not updated in BRM. When you set event time stamps during testing, ensure the event time of the top-up request is later than the applicable account creation time.

For details about the top-up API, see the documentation for [oracle.communication.brm.charging.brs](#) and [oracle.communication.brm.charging.messages.update](#) in *BRM ECE Java API Reference*. 
This document describes how to configuring balance queries in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:
- Integrating Balance Query clients
- About Sending Authentication Queries
- About Sending Balance Queries
- Configuring Debit Request History
- Balance API

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client
- Configuring Top-Ups

**Integrating Balance Query clients**

You can write client applications to query data in ECE, such as query the login and password information of a customer, or query the customer’s account balance.

ECE returns the balance element ID of each balance in the ECE balance query response. ECE returns the balance element ID of balances for both SUMMARY and DETAILED balance query modes. Client applications could use this information, for example, when customer balances are stored in multiple subscriber profile repositories and it is required to map the balances between the repositories.

To use the query APIs:

- For the ECE authentication and query API: `oracle.communication.brm.charging.messages.query`
- For the ECE PriceEnquiry of the charging API: `oracle.communication.brm.charging.brs` and `oracle.communication.brm.charging.messages`

For information about running sample programs that demonstrate how to use the ECE query APIs, see "ECE Sample Programs".
About Sending Authentication Queries

Use the authentication API to query the login and password of subscribers. Use the login and password information for:

- Implementing authentication methods outside of the ECE charging server
- Enabling subscribers to validate their login and password credentials against a charge offer to which they are subscribed

About Sending Balance Queries

Use the balance API to query subscriber balances. Use the subscriber balance for:

- Making policy decisions
- Sending the balance information to subscribers so they can monitor their network-usage expenses, validate their credit limit, or monitor their active reservation

ECE returns the balance element numeric ID of each balance in the ECE balance query response. ECE returns the balance element numeric ID of balances for both SUMMARY and DETAILED balance query modes. Client applications could use this information, for example, when customer balances are stored in multiple subscriber profile repositories and it is required to map the balances between the repositories.

ECE returns grantor information in the ECE balance query response for DETAILED balance query mode. Grantor information consists of the Grantor ID and the Grantor Type. The different grantor types include purchased charge offerings, purchased alteration offerings, charge offerings, and alteration offerings.

Configuring Debit Request History

For a debit request, ECE returns a correlation ID in the usage response and stores the correlation ID in a debit map for each charge offer. If a refund request is later received for the debit request, ECE uses the correlation ID to validate the refund request (refund requests are valid only when they are associated with a debit request correlation ID).

The debit request information in the debit map is transient data and you can configure the number of debit requests to be retained per charge offer. By default, debit request information is stored for ten debit request operations (for each charge offer) at any given time. For example, if the debit map contains ten entries and a new debit request is received, the entry for the oldest debit request is deleted from the debit map, and an entry for the new debit request is added.

To configure the debit request history:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
In the editor’s MBean hierarchy, find the ECE configuration MBeans.

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

3. Expand **charging.server**.

4. Expand **Attributes**.

5. Set the **debitRefundSessionEvictionSize** attribute to the maximum number of debit requests to log in the debit map at one time.

This is the number of debit requests to keep in history so that refunds can be made against them.

**Balance API**

Use the balance API to query subscriber balances.

Use the subscriber balance for:

- Making policy decisions
- Sending the balance information to subscribers so they can monitor their network-usage expenses, validate their credit limit, or monitor their active reservation

When building a balance query request, you have the option to use the following balance query modes to restrict the contents of the balance query response that ECE returns:

- **SUMMARY**
  
  When using the SUMMARY balance query mode, the balance query response contains only the total balance at the balance element level.

- **DETAILED**
  
  When using DETAILED balance query mode, the balance query response contains the detailed balance for each of the balance elements; this includes balance element specification information (such as credit limits) as well as reservation information (such as the active and consumed reservation).

The balance query response mode you use may impact the overall performance of your system.

For details about the balance API, see the documentation for `oracle.communication.brm.charging.messages.query` in BRM ECE Java API Reference.
Advice of Charge and Advice of Promotion

This document describes how to configure Advice of Charge (AOC) and Advice of Promotion (AOP) in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

- About Advice of Charge
- About Advice of Promotion

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

About Advice of Charge

ECE supports the 3GPP Advice of Charge (AoC) supplementary service by which customers can be informed about the cost for a requested service either in monetary format or nonmonetary format. AoC may be provided at the beginning of a session, during a session or at the end of a session.

To support AoC, ECE calculates the cost of using a service and relays that information to the network mediation software program, which can then pass the message to the customer.

About Advice of Promotion

ECE enables you to provide Advice of Promotion (AoP) information to customers to notify them that a better price can be obtained for a service they are about to use. For example, a network operator can send the AoP information in an IVR pre-call announcement for a Voice service.

To support AoP, ECE determines whether better pricing for a service is available near the time the customer’s usage request is received. ECE sends that information to the network mediation software program, which sends a message to the customer.

ECE implements AoP as follows:

1. A customer makes a request to initiate a session, to debit a specific or calculated amount of a balance, or to generate a price estimation for using a balance.
2. The ECE charging server calculates the charge for the request.

3. If AoP is enabled, ECE adds a time offset to the start and end time of the request and recalculates the charge using the offset time period (the new start and end time).

4. If the recalculated charge is less expensive for the customer, ECE sends the information about potential savings to the network mediation software program in the usage response.

ECE applies AoP when AoP is configured at the ECE system level. Configure AoP at the system level by using the configuration service.

Note the following details about AoP:

- AoP is not configurable in PDC.
- AoP is a systemwide configuration (it is not configured on a per charge offer basis).
- The ready-to-use configuration of AoP gives advice based on time.
- When applying AoP, ECE uses the charge offers and discount offers eligible when the request is received to recompute the charge for the offset time period. If a different charge offer or a different discount offer applies to the future offset time period, AoP may advise a promotion when none exists or may not advise a promotion when a promotion is available.

When using AoP, ensure that your charge offers have tiered consumption configured accurately to prevent a credit breach of noncurrency balances.

To configure Advice of Promotion:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/occeserver/config/eeTopologies.conf` file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.

2. Expand the ECE Configuration node.

3. Expand charging.server.

4. Expand Attributes.

5. Specify values for the following attributes:
   - **aopEnabled**: Enter `true` to enable AoP or `false` to disable AoP.
   - **aopVariance**: Enter an amount of time in the ISO 8601 duration format (for example, `PT10M`, which specifies ten minutes).

      ECE uses the time you specify to offset the start and end times of the request and recalculate the charge for the offset period.

      For more information about the duration format, see the ISO 8601 documentation.
This document describes how to configure Multiple Services Credit Control in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

About Multiple Services Credit Control

ECE supports Multiple-Service Credit Control (MSCC) requests in which a Diameter application performs credit control for multiples services within the same session.

An MSCC request is a list of subrequests targeted to the same customer that share the same operation type and session ID but that individually apply to different charge offers.

When ECE receives MSCC requests, it assigns a different session ID to each of its subrequests. Doing this enables ECE to distinguish one subrequest from another when looking up the active session associated with each subrequest.

An MSCC request results in an MSCC response containing a subresponse for each subrequest. Each subresponse contains a status indicating whether the subrequest succeeded or failed.

If you have configured ECE to save the rated event information for MSCC requests in the Oracle NoSQL database, note the following:
- Rated event information is saved for each subrequest.
- The NoSQL key for the rated event is based on the session ID that ECE assigned (not on the original MSCC request session ID).
- The ECE session ID in the Oracle NoSQL database is a composite of the original usage request's session ID, the service, and the user identity, separated by underscore characters. For example:
  - Original MSCC request ID: 1313b2ab-d51e-4545-8bba-25c731daf10b
  - Usage request's service: VOICE
  - Usage request's user ID: 650123555
  - ECE session ID: 1313b2ab-d51e-4545-8bba-25c731daf10b_Voice_650123555
MSCC support applies to usage requests and query requests.

MSCC support does not include support for rating groups (Rating-Group AVP), credit pools (G-S-U-Pool-Reference AVP where units of the service are pooled in a credit pool), and credit control (as described in section 5.1.2 of IETF RFC 4006).

MSCC AVPs are part of the CCR and Diameter Gateway expects each Gy interface request type to be included in the MSCC group even if the request contains only a single service. When CCR is sent without MSCC AVPs, Diameter Gateway validates only the subscriber ID in the CCR and authenticates the subscriber.

See the SampleMultipleServicesLauncher sample program in the ECE SDK for an example of how to send MSCC requests to ECE.
Synchronizing Data Between ECE and the BRM Database

This document provides instructions for synchronizing data between the Oracle Communications Billing and Revenue Management (BRM) database and the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

- Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates
- Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE
- Configuring Item Assignment for Rated Events
- Configuring Life Cycle States in ECE for BRM
- Configuring ECE to Send Rated Events to BRM
- How ECE Gets Historical Data From the BRM Database

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates

When customer data is updated in the BRM database, the updates must be applied synchronously (in real time).

To enable real-time synchronization of BRM and ECE customer data updates:

1. Open the $BRM_home/sys/cm/pin.conf$ file in a text editor.
2. Add the following entries to the end of the file:

   ```
   -cm ece_real_time_sync_db_no 0.0.9.8
   -cm em_group ece PCM_OP_ECE_PUBLISH_EVENT
   -cm em_pointer ece ip emGateway_host emGateway_port
   ```

   where
Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE

- *emGateway_host* is the name or IP address of the server on which External Manager (EM) Gateway is running.
- *emGateway_port* is the number of the port through which EM Gateway connects to the host.

---

**Note:** By default, the publisher database number for EM Gateway is 0.0.9.8.

---

3. If the publisher database number of EM Gateway in your system is not 0.0.9.8, replace 0.0.9.8 with the correct publisher database number in the following entry:

   ```
   -cm ece_real_time_sync_db_no 0.0.9.8
   ```

4. Save and close the file.

5. If you changed the EM Gateway publisher database number in your CM `pin.conf` file, do the following:
   a. Open the `BRM_home/sys/eai_js/payloadconfig_ifw_sync.xml` file in a text editor (or the merged file if you merged payload configuration files).
   b. Locate the `PublisherDefs` section.
   c. In the `Publisher DB="DB_number"` entry, replace `DB_number` with the publisher database number of your EM Gateway.

6. Save and close the file.

7. If you changed the EM Gateway publisher database number, restart the Payload Generator External Module (also called the Enterprise Application Integration (EAI) Java Server or `eai_js`).

8. Restart the CM.

---

**Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE**

The CM connects to ECE through EM Gateway.

To configure the CM to get real-time balances for a service from ECE:

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

2. Add the following entry:

   ```
   - cm em_group ece_bal PCM_OP_BAL_GET_ECE_BALANCES
   ```

3. Set the following entry to match your environment:

   ```
   - cm em_pointer ece_bal ip emGateway_host emGateway_port
   ```

4. Save and close the file.

5. Stop and restart the CM.

---

**Configuring Item Assignment for Rated Events**

You configure item assignment in ECE so that customer balance impacts can be tracked appropriately. Typically, the default configuration is sufficient. If you have
custom item assignments, you might need to change the configuration for item assignment.

The item-type field maps to BRM items; this mapping is required for loading rated events from ECE to the BRM database. Each rated event record has an `item_type` field which is derived from the mapping specified in the `itemType` MBean attribute. The `itemType` MBean attribute lists the ECE service/event combinations used in event definitions.

For example, usage events are typically applied to the `/item/misc` object, also known `misc` item type. To map voice and data events to the `misc` item ECE maps `itemType="misc"` to the `itemTag="VOICE_DATA_misc"`. The XML file that stores this configuration shows how the mapping works.

```xml
<itemTypeDetail itemType="misc" itemTag="VOICE_DATA_misc">
  <itemTagDetail
    productType="VOICE"
    eventType="USAGE">
  </itemTagDetail>
  <itemTagDetail
    productType="DATA"
    eventType="DATA_USAGE">
  </itemTagDetail>
</itemTagDetails>
```

In this example, the `VOICE_DATA_misc` item tag includes to ECE service/event mappings: VOICE/USAGE and "DATA/DATA_USAGE." When a usage request is created the item mapping specifies that the `misc` item type is to be assigned to the events.

If you configured delayed billing in BRM, you must configure item assignment in ECE to process delayed usage requests in the appropriate accounting cycle.

To configure item assignment for rated events:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceceserver/config/eceTopology.conf` file.
      
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

2. Expand `charging.itemAssignmentConfig`.

3. Expand `Attributes`.

4. Review the following attributes:
   - `itemAssignmentEnabled`: Enter `true` to turn on item assignment or `false` to turn it off.
   - `poidQuantityPerSchema`: Double-click the `Value` field. A list of schemas and the quantity of POID IDs that is reserved at ECE startup for each schema appears.
- **delayToleranceIntervalInDays**: Enter the number of days during which delayed usage requests are processed for the current accounting cycle. This interval must be less than the delayed billing interval (the value of the `config_billing_delay` entry in the `BRM_home/sys/cm/pin.conf` file). See the discussion about processing delayed usage requests for more information.

5. To add a schema to the `poidQuantityPerSchema` list or to change the quantity of POID IDs for a schema in the list:
   a. Expand **Operations**.
   b. Select **setPoidQuantity**.
   c. Specify values for the following parameters:
      - **schema**: Enter the BRM schema number for which the POID IDs must be reserved. For example, in a multischema environment, enter 1 for the primary schema, 2 for the secondary schema, and so on.
      - **quantity**: Enter the number of POID IDs that must be reserved at ECE startup for the specified schema.
   d. Click the **setPoidQuantity** button.

**Configuring Life Cycle States in ECE for BRM**

ECE supports the BRM subscriber life cycle state feature. If the subscriber life cycle state feature is disabled in BRM, ECE supports only the default subscriber life cycle, which has the following states: Active, Inactive, and Closed. If the subscriber life cycle state feature is enabled in BRM, ECE supports custom subscriber life cycles, which has the following states: Preactive, Active, Recharge Only, Credit Expired, Fraud Investigated, Dormant, Suspended, and Closed. See the discussions about service life cycles and enabling BRM to use custom service life cycles in *BRM Managing Customers* for more information.

You can customize the pre-configured subscriber life cycle state. You must configure life cycle states, so they stay synchronized with life cycle states you add in BRM.

To configure life cycle states in ECE for BRM:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor's MBean hierarchy, find the ECE configuration MBeans.
2. Expand the **ECE Configuration** node.
3. Expand **charging.lifecycleConfiguration**.
4. For each subscriber life cycle state, configure the rules and transitions defined for each state for each service.
Configuring ECE to Send Rated Events to BRM

The ECE charging servers generate rated events as a result of usage-request processing. ECE sends the rated events to BRM so that customer balances in BRM can be updated.

For ECE to send rated events to BRM, you must configure the following components:

- Rated Event Publisher (if Oracle NoSQL Database is used for storing rated events)
  See "Configuring Rated Event Publisher".
- Rated Event Formatter
  See "Configuring Rated Event Formatter".
- BrmCdrPluginDirect Plug-in
  See "Configuring the Rated Event Formatter Output".
- Rated Event Loader
  See "Setting Up Rated Event Loader for ECE".

Assuming these components are configured and Rated Event Formatter is running, the ECE charging servers generate event data record (EDR) files as a result of usage-request processing. The BrmCdrPluginDirect Plug-in formats the EDR files in BRM call details record (CDR) format. The EDRs can then be loaded by Rated Event (RE) Loader into the BRM database.

Configuring Rated Event Publisher

If you are using Oracle NoSQL Database to store rated events, you must configure Rated Event Publisher. Rated Event Publisher publishes ECE-generated rated events to the Oracle NoSQL database data store.

To configure Rated Event Publisher:

1. Open the \ECE_home\oceceserver\config\management\charging-settings.xml file.
2. In the ratedEventPublisher section, set the required configuration parameters.

Table 10–1 provides configuration parameter descriptions and default values.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description and Guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataStoreConnection</td>
<td>&quot;localhost:5000&quot;</td>
<td>This parameter configures Rated Event Publisher to connect to the Oracle NoSQL database; it configures the data store connection to the Oracle NoSQL database system. The Oracle NoSQL database connection string uses the format hostname:port for connecting to a pre-configured Oracle NoSQL database system. The default is &quot;localhost:5000&quot; for connecting to a standalone Oracle NoSQL database system (KV-Lite).</td>
</tr>
</tbody>
</table>
Configuring ECE to Send Rated Events to BRM

1. Save the file.

Configuring Rated Event Formatter

To configure Rated Event Formatter:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to start CohMgt = true in the ECE_home/oceceserver/config/eceTopology.conf file.

      The eceTopology.conf file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

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 remaining attributes you need to set for the instance.

See Table 10–2 for attribute descriptions and default values.
**Table 10–2 Configuration Parameters for a Rated Event Formatter Instance**

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;formatter&quot;</td>
<td>The name of a Rated Event Formatter instance. Name Rated Event Formatter instances consistently and uniquely (for example, formatter1, formatter2, and so on). name must match the name of the Rated Event Formatter node instance in the ECE_home/ociceserver/config/eceTopology.conf file.</td>
</tr>
<tr>
<td>partition</td>
<td>&quot;1&quot;</td>
<td>The partition for the rated events to be processed by a Rated Event Formatter instance. 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 the discussion about 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.</td>
</tr>
<tr>
<td>connectionName</td>
<td>&quot;OraclePersistence1&quot;</td>
<td>The connection name of the persistence database. This attribute is applicable only if you are using Oracle Database for storing rated events.</td>
</tr>
<tr>
<td>dataStoreConnection</td>
<td>&quot;localhost:5000&quot;</td>
<td>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.</td>
</tr>
<tr>
<td>dataStoreName</td>
<td>&quot;kvstore&quot;</td>
<td>The data store name to be used to access an Oracle NoSQL system. This attribute is applicable only if you are using Oracle NoSQLDatabase for storing rated events.</td>
</tr>
<tr>
<td>retainDuration</td>
<td>0</td>
<td>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 NoSQLDatabase for storing rated events.</td>
</tr>
</tbody>
</table>
Configuring ECE to Send Rated Events to BRM

5. Change directory to the ECE_home/oceceserver/bin directory.

6. Start ECC:

   ./ecc

---

Table 10–2 (Cont.) Configuration Parameters for a Rated Event Formatter Instance

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ripeDuration</td>
<td>60</td>
<td>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 <code>ripeDuration</code> time allows time for resolving any duplicate rated events that may have been persisted to the Oracle NoSQL database. The <code>ripeDuration</code> 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.</td>
</tr>
</tbody>
</table>
| 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. |
| threadPoolSize   | 4       | The number of threads used by the Rated Event 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. |
| pluginPath       | n/a     | The path to the JAR library that contains the reader plug-in implementation. A custom plug-in has a modified path to the JAR library. |
| pluginName       | n/a     | The class name with the package path for the formatter plug-in object to be called by Rated Event Formatter. Rated Event Formatter is configured by default to use the BrmCdrPluginDirect plug-in. If you are using the BRM database to process rated events, the `pluginName` is:

   oracle.communication.brm.charging.ratedevent.formatterplugin.BrmCdrPlugin

If you are using a third-party system to process rated events, the `pluginName` is:

   oracle.communication.brm.charging.ratedevent.custom.CUSTOMPlugin

| logFormatterWorker | false   | Whether to enable (true) or disable (false) logging for the worker thread pool. |
| primaryInstanceName | "ratedEventFormatter 1" | The name of the primary Rated Event Formatter instance. |
7. Stop and restart any Rated Event Formatter instances that you configured.
   Each instance reads its configuration information by name.

   For information about stopping and starting Rated Event Formatter instances, see
   the discussion about starting and stopping ECE in BRM System Administrator’s
   Guide.

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.

Configuring the Rated Event Formatter Output
Rated events can be processed in different formats with a BRM database or a
third-party system. To format ECE-rated events into BRM format, configure the
BrmCdrPluginDirect plug-in. To format ECE-rated events in a different format,
configure a custom plug-in. See "About the Custom Plug-in API".

To configure the BrmCdrPluginDirect plug-in:
1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to start CohMgt = true in the
      $ECE_home/oceceserver/config/eceTopology.conf$ file.
      The eceTopology.conf file also contains the host name and port number for
      the node.
   e. In the editor's MBean hierarchy, expand the ECE Configuration node.
2. Expand charging.brmCdrPlugins.Instance_Name, where Instance_Name is the
   name of the instance that you want to configure.
3. Expand Attributes.
4. Specify values for the following attributes as needed:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempDirectoryPath</td>
<td>&quot;/tmp/tmp&quot;</td>
<td>The directory path for the BrmCdrPluginDirect plug-in to store temporary files while processing the rated event objects.</td>
</tr>
<tr>
<td>doneDirectoryPath</td>
<td>&quot;/tmp/done&quot;</td>
<td>The directory path for the BrmCdrPluginDirect plug-in to store completed CDR files from processing the rated event objects.</td>
</tr>
<tr>
<td>doneFileExtension</td>
<td>&quot;.done&quot;</td>
<td>The file extension for the completed CDR files created by the BrmCdrPluginDirect plug-in.</td>
</tr>
</tbody>
</table>

5. Stop and start the Rated Event Formatter instance associated with the
   BrmCdrPluginDirect plug-in instance.
Setting Up Rated Event Loader for ECE

Rated Event (RE) Loader picks up the CDR files from the output directory of the BrmCdrPluginDirect plug-in and loads the events into the BRM database. RE Loader and Rated Event Formatter must be able to access the same file system.

ECE writes all event fields to CDRs. If you want RE Loader to load only specific event fields into the BRM database, you must modify the RE Loader control files to indicate which event fields you want it to load. Control files with ECE control data are included with ECE. You copy and merge these files onto the BRM system. For information about editing control files, see the Oracle database utilities documentation at the following location:

http://docs.oracle.com/cd/B10501_01/server.920/a96652/ch05.htm#1004643

How ECE Gets Historical Data From the BRM Database

Because there is a gap of time between when a call occurs and when it is rated, information about the customer can change during that time. For example, a customer might change the phone number before a call is rated. ECE must look up account data based on the old number.

To retrieve historical information, ECE gets data from audited objects. By default, auditing in BRM is turned off for most objects. After you install the account synchronization components, you must run the object_auditing.pl script to turn on auditing for the objects and fields that ECE needs data about. See BRM Installation Guide.
Generating POIDs for Rated Events

This document describes how to configure Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) to generate POIDs for events created in ECE.

See also:
- List of Charging Features
- Synchronizing Data Between ECE and the BRM Database
- Managing Online Charging Sessions

About Generating POIDs in ECE

You use portal object IDs (POIDs) to track rated events and bill items. For tracking events created in ECE, POIDs are generated as follows:

- For delayed events, ECE generates the POIDs by default.
- For prepaid events, BRM generates the POIDs and sends them to ECE by default. You can configure ECE to generate the POIDs. See "Configuring ECE to Generate POIDs for Prepaid Events".
- For non-usage events, such as subscription events, BRM generates the POIDs and sends them to ECE.

ECE uses Rated Event Formatter to generate the POIDs and persists the last allocated POID ID in the database. This ensures that the POIDs are generated without any duplication even if the ECE system is restarted.

The POID generated in ECE contains the following information:

```
.event_type date cluster_id BRM_schema_id unique_id
```

See Table 11–1 for the description of each entry in the POID.

### Table 11–1 POID Entries in ECE

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_type</td>
<td>A unique 4-bit number assigned to each event type. For example, 0 is assigned to subscription events, 1 is assigned to postpaid events (USAGE_POSTPAID), and 2 to 7 is assigned to prepaid events (USAGE_PREPAID) depending on the prepaidPartitionSet value specified in BRM. The default value for event_type is 0.</td>
</tr>
</tbody>
</table>
You can configure multiple instances of Rated Event Formatter for uninterrupted POID allocation. In case if the primary Rated Event Formatter instance fails, the secondary Rated Event Formatter instance ensures that the POIDs are generated and allocated without any interruption. In a disaster recovery deployment, if the Rated Event Formatter instance in the primary site fails, the Rated Event Formatter instance in the backup site continues the POID allocation for the events.

For tracking the bill items and non-usage events created in ECE, ECE uses the POIDs received from BRM. ECE persists the POID pool received from BRM in the database. This ensures that the reserved POID pool is retained in ECE even after the ECE restart. It allows ECE to continue the POID allocation using the existing POID pool.

**Configuring ECE to Generate POIDs for Prepaid Events**

To configure ECE to generate POIDs for prepaid events, you must perform the following:

1. Enable prepaid-event partitions in BRM. See "Enabling Prepaid Event Partitions in BRM".

2. Ensure that the cluster ID is configured for ECE clusters. The cluster ID must be specified if you have ECE configured for disaster recovery. See "Configuring Cluster ID".

3. Ensure that the name of the primary Rated Event Formatter instance is specified in each Rated Event Formatter instance. See the `primaryInstanceName` MBean attribute in "Configuring Rated Event Formatter".

   The primary Rated Event Formatter instance must be specified if you have ECE configured for disaster recovery.

4. Enable POID generation for prepaid events in ECE. See "Enabling POID Generation for Prepaid Events in ECE".

**Enabling Prepaid Event Partitions in BRM**

To enable prepaid-event partitioning in BRM:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The 16-bit date on which the POID is generated. The date is determined based on ECE virtualTime if it is enabled. For more information on virtualTime, see &quot;Changing Time and Date to Test ECE&quot;.</td>
</tr>
<tr>
<td>cluster_id</td>
<td>A unique 4-bit number assigned to the Coherence cluster to identify ECE in the cluster. The cluster_id is limited to 0 to 15 and the maximum number of ECE clusters allowed in a deployment is 16. The default value for cluster_id is 0. If ECE is configured for disaster recovery, you must specify the cluster ID for each cluster used in the Active-hot standby or Active-cold standby systems.</td>
</tr>
<tr>
<td>BRM_schema_id</td>
<td>A unique 6-bit number assigned to the BRM schema. The BRM_schema_id is limited to 0 to 31.</td>
</tr>
<tr>
<td>unique_id</td>
<td>A unique 34-bit number assigned to each POID.</td>
</tr>
</tbody>
</table>
1. Open the BRM_Home/sys/dm_oracle/pin.conf file in a text editor.

2. Set the prepaid_partition_set entry to a numerical value only between 2 and 7. For example:
   - `dm prepaid_partition_set 2`

3. Set the prepaid_partition_transition_mode entry to 1:
   ```
   - dm prepaid_partition_transition_mode 1
   ```

   **Note:** Setting this entry to 1 enables Data Manager to retrieve the partitions for the existing events. After retrieving all the partitions for the existing events (for example, after 90 days), set this entry to 0 to disable this mode.

4. Save and close the file.

5. Create an editable XML file from the system instance of the /config/business_params object:
   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

6. Set the prepaidPartitionSet parameter to the value you specified in step 2. For example:
   ```
   <prepaidPartitionSet>2</prepaidPartitionSet>
   ```

7. Save the file as bus_params_system.xml.

8. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_system.xml
   ```

9. Stop and restart the CM.

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

11. Go to the BRM_home/apps/partition_utils directory.

12. Enable prepaid-event partitions by running the following command:
    ```
    partition_utils -o enable -t prepaid
    ```

13. Add prepaid-event partitions by running the following command:
    ```
    partition_utils -o add -t prepaid -s start_date -u month|week|day -q quantity
    ```

   where:
   - `start_date` specifies the starting date for the new partitions. The format is MMDDYYYY.
   - `quantity` specifies the number of partitions to add. Enter an integer greater than 0.

**Important:** In multischema systems, perform this task first on the primary BRM installation machine and then on the secondary BRM installation machines.
For more information on enabling and adding partitions, see the discussion about partitioning database tables in *BRM System Administrator’s Guide*.

**Configuring Cluster ID**

To configure the cluster ID for ECE clusters:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/occeserver/config/eceTopology.conf` file.

      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

2. Expand `charging.clusters.Cluster_Name`, where `Cluster_Name` is the name of the ECE cluster that you are configuring.

3. Expand Attributes.

4. Set the `id` attribute to a unique number that indicates the ID of the cluster in the POID generated in ECE.

   Rated Event Formatter uses the cluster ID in the POID to identify the ECE clusters. The cluster ID must be unique for each cluster.

**Enabling POID Generation for Prepaid Events in ECE**

To enable POID generation for prepaid events in ECE:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/occeserver/config/eceTopology.conf` file.

      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.

2. Expand `charging.brmCdrPlugins.Instance_Name`, where `Instance_Name` is the name of the BrmCdrPluginDirect Plug-in instance you are configuring.

3. Expand Attributes.

4. Set the `prepaidPartitionSet` attribute to the value that you specified in the `prepaid_partition_set` entry in the `BRM_Home/sys/dm_oracle/pin.conf` file.

**Note:** To enable POID generation in ECE, you must set this attribute to a number between 2 and 7. If this attribute is set to 0, ECE uses the POIDs received from BRM for tracking events.
Connecting ECE to a Diameter Client

This document provides information about network integration for online charging using Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) Diameter Gateway.

Topics in this document:

■ Overview of Network Integration Using Diameter Gateway
■ Network Integration for Sp and Sy Interface (Policy) Requests
■ Network Integration for Gy Interface Requests
■ Adding Custom AVPs for Usage Requests
■ Configuring Notifications for Diameter Gateway
■ Configuring Alternative Diameter Peers for Notifications
■ Viewing Active Diameter Peers
■ Handling Requests When Charging Servers Are Unavailable

See also:

■ List of Charging Features
■ About Configuring Charging in Elastic Charging Controller
■ Managing Online Charging Sessions

Overview of Network Integration Using Diameter Gateway

The following steps summarize how to set up network integration for online charging using Diameter Gateway, which enables Diameter Gateway to do the following:

■ Receive Gy, Sp, and Sy Diameter requests from Diameter clients and translate them into ECE requests.

■ Submit ECE requests to ECE charging servers for credit-control processing.

■ Receive ECE request responses and translate them into respective Gy, Sp, and Sy Diameter message responses.

■ Send Diameter message responses to Diameter clients.

■ Consume JMS notifications from the ECE notification queue, create Diameter notification messages from them, and send the notification messages to the appropriate Diameter clients.

To implement Diameter Gateway:
1. During the ECE installation, add Diameter Gateway node instances required for your topology and configure each instance.

2. For the all request types you receive from the network, ensure that your credit-control request (CCR) message formats adhere to the attribute value pair (AVP) fields that Diameter Gateway supports and requires.

3. For Gy interface Diameter requests, ensure that you have done the following:
   ■ Defined any custom service/event mappings in PDC.
   ■ Editted your mediation specification file and loaded it into ECE.

   The mediation specification enables Diameter Gateway to associate each Gy interface Diameter request with its respective usage-request builder.

   See "Network Integration for Gy Interface Requests".

4. Configure notifications for Diameter Gateway.

   Diameter Gateway listens for notifications on the ECE (JSM) notification queue (for push notifications from Elastic Charging Server).

   You must enable the notification types to be generated in the ECE system. See "Configuring Notifications for Diameter Gateway".

   You can set tuning parameters for how Diameter Gateway sends notifications to Diameter clients.

   If a Diameter client fails or becomes unavailable before receiving a notification message from a Diameter Gateway instance, Diameter Gateway can route the notification message to another available Diameter peer. For information, see "Configuring Alternative Diameter Peers for Notifications".

   See "About ECE Notifications" for information about configuring the JMS credentials in ECE for the ECE notification queue.

5. Start the Diameter Gateway nodes.

   When the Diameter Gateway nodes start, they automatically join the Coherence cluster gaining access to ECE caches and invocation services that it uses to send requests to ECE. At startup, the Diameter Gateway instances read from the ECE notification queue for notifications.

### Network Integration for Sp and Sy Interface (Policy) Requests

This section provides information about network integration for policy requests using Diameter Gateway.

Given that the technical implementation of Sp has not been defined by the 3GPP standards body, Diameter Gateway uses the Sh interface as the implementation to request and subscribe to policy-related information in the ECE server.

Diameter Gateway retrieves Sp and Sy information from ECE charging servers and sends the information to the Policy and Charging Rule Function (PCRF).

The following Sp (implemented as Sh) and Sy interface policy request types are processed by Diameter Gateway (using the ECE policy-request builders).

**Sy:**

■ Spending Limit Report Request (SLR/SLA)
■ Subscribe Notification Request
Diameter Gateway manages notification subscriptions (when the PCRF subscribes and unsubscribes) for notifications due to Sy and Sp related updates.

Diameter Gateway listens for notifications on the ECE (JSM) notification queue (for push notifications from the Elastic Charging Server). For policy-driven charging, when changes occur to policy counters (balances) or to policy-related subscriber preferences associated with charge offers that have an active policy session, ECE charging servers publish asynchronous notifications to the JMS notification queue. Diameter Gateway receives the policy notifications at startup and processes them as follows:

- From spending-limit JMS notifications, Diameter Gateway creates Sy (Spending-Status-Notification-Request (SNR)) messages for all subscribed sessions and routes them to the appropriate Diameter clients.
- From subscriber-preferences JMS notifications, Diameter Gateway creates Sp/Sh (Push-Notification-Request (PNR)) messages for all subscribed sessions and routes them to the appropriate Diameter clients.

For information about how Diameter Gateway uses the ECE policy management APIs to retrieve Sy-interface and Sp-interface data from the ECE server, see “Configuring Policy-Driven Charging”.

To enable Diameter Gateway to create ECE requests for policy-driven charging, you must configure notifications for Diameter Gateway. See “Configuring Notifications for Diameter Gateway”. You can configure alternative Diameter peers for each peer to which a Diameter Gateway instance connects for routing notifications. See "Configuring Alternative Diameter Peers for Notifications”.

Ensure that your policy CCR message formats adhere to the to the well-known AVP fields of the 3GPP standard for Sh and Sy policy requests.

Network Integration for Gy Interface Requests

This section provides information about network integration for Gy interface online charging requests using Diameter Gateway.

The following Gy interface credit-control request types are processed by Diameter Gateway (using ECE usage-request builders):

- Session-based requests
  - Initiate
  - Update
  - Terminate
  - Cancel
- Price enquiry
- Direct debit
- Refund

For Gy interface credit-control requests, you must do the following for Diameter Gateway to process the requests successfully:
Present Gy interface request types inside of a Multiple-Service Credit Control (MSCC) group.

MSCC AVPs are part of the CCR and Diameter Gateway expects each Gy interface request type to be included in the MSCC group even if the request contains only a single service. Contain the following Gy interface request types in a MSCC group:

- Initiate
- Update
- Terminate
- Cancel
- Price enquiry
- Direct debit
- Refund

For more information about MSCC requests and ECE, see "Configuring Multiple Services Credit Control."

Add network attributes for all event attributes in the event definition that apply to usage-request charging operations.

Diameter Gateway uses the network specification and corresponding network attributes to dynamically populate the event attributes of ECE requests with the CCR AVP data of your incoming Diameter request.

See "How Diameter Gateway Creates Usage Requests".

Edit your mediation specification file and load your mediation specification into ECE.

The mediation specification enables Diameter Gateway to associate each Gy interface Diameter request with its respective usage-request builder.

See "Editing the Mediation Specification File".

Diameter Gateway uses incremental based accounting behavior when processing usage requests.

Diameter Gateway listens for notifications on the ECE (JSM) notification queue (for push notifications from the Elastic Charging Server). From the ECE reauthorization-request JMS notifications it receives, Diameter Gateway creates Gy RAR messages and sends them to Diameter clients running the applicable active Gy sessions.

The Diameter Gateway uses ECE usage-request builders to create request and response messages for Gy interface request types.

**How Diameter Gateway Creates Usage Requests**

Diameter Gateway creates usage requests based on the event definitions sent from PDC to ECE. Diameter Gateway includes a usage-request builder for creating usage requests (as well as different builders for building other requests ECE supports, such as balance query requests, and top-up requests, and so on). When you start Diameter Gateway nodes, the usage-request builder reads the event definition data and sends requests that adhere to the specifications. See "About Usage Request Fixed Attributes" for more information on the attributes.

When you perform network enrichment of the event definition for your events in PDC, you add network attributes for all event attributes in the event definition that
apply to usage-request charging operations. Diameter Gateway uses the network specification and corresponding network attributes to dynamically populate the event attributes of ECE requests with the CCR AVP data of your incoming Diameter request.

You can have Diameter Gateway dynamically populate some fields using the event-attribute to network-attribute you map in PDC and you can have Diameter Gateway explicitly populate other fields using your own custom extension code (for example, when using the Pre-OCS extension, you can explicitly populate the ECE payload for fields using your Pre-OCS extension mechanism).

**About Usage Request Fixed Attributes**

Usage requests contain a set of well known or fixed attributes that must be provided. Fixed attributes are required fields directly exposed by the UsageRequest interface. Fixed attributes are applicable for all the events in ECE.

You cannot pass in null for any of the fixed attributes. For non-duration requests, you can pass the same timestamp for both requestStart and requestEnd.

Fixed attributes within a usage request include the following:

- **userIdentity**
  The userIdentity attribute is the fixed attribute name representing the public user identity of the person or entity using the product (phone number, email address and so on). It is a generic way of identifying who is being charged for the usage.

- **requestId**
  The requestId is an identifier that uniquely identifies the usage interaction. If the usage is session based, the requestId must be the same across different operation types (Initiate, Update and Terminate). The requestId is used to locate the active session associated with the charging customer.

- **requestStart**
  The requestStart is the time at which the usage started.
  
  For session-based usage requests, ECE observes the requestStart value for Initiate operation-type usage requests.

- **requestEnd**
  The requestEnd is the time at which the usage ended.
  
  If the usage interaction has no duration, such as for event-based charging, the requestStart is equal to the requestEnd.

  **Note:** If the payload contains a non-null "DURATION" attribute (either as a top-level attribute or under a Requested Service Units (RSU) and Used Service Units (USU) block, its value will override the value of the requestEnd attribute.

- **requestMode**
  The requestMode defines the mode of the usage request. Valid values are OFFLINE and ONLINE.
  
  For backward compatibility, the default value is ONLINE.

- **sequenceNumber**
The sequenceNumber is the sequential session-centric attribute and is a type of subID you can apply for different types of charging within a session. You cannot change the name of the fixed attributes.

Usage requests also contain configurable (dynamic) attributes. Configurable attributes are defined in the payload blocks of the event definition (request specification data defined in PDC when you enrich event definitions).

**Editing the Mediation Specification File**

The mediation specification enables Diameter Gateway to associate each Diameter request with its respective usage-request builder. Diameter Gateway uses the mediation specification to determine which service and event combination applies to an incoming Diameter request, enabling it to select the event definition that applies to the event to be rated.

You configure Diameter Gateway to base its selection of event definitions on any combination of the following AVPs in the request:

- Service-Context-Id
- Service-Identifier
- Rating-Group
- Event-Timestamp

From the preceding AVP values, Diameter Gateway derives the following fields, which uniquely identify the event definition to use for building the ECE request:

- ProductType (service)
- EventType
- Version

You can configure Diameter Gateway to base its event definition on a custom AVP by using the Diameter Gateway Request-Received extension. You use that extension to modify one of the AVP values in the request so that a different Diameter mediation mapping is produced for a service, event, and version.

To edit the mediation specification:

1. Create a mediation specification file or edit an existing one.
   
   A sample mediation specification file is available at `ECE_home/occeserver/sample_data/config_data/specifications/ece_end2end/diameter_mediation.spec`

   It is recommended to create only one mediation specification file to represent your mediation specification. You can have only one mediation specification loaded in the ECE cluster and the last one loaded takes precedence.

2. In the mediation specification file, add a row (in the table) for each event to be rated that specifies the following information:

   - Rating-Group AVP
     
     The Rating-Group AVP value sent in the Diameter message.
     
     Null is an acceptable value if the field is not expected to be present on the CCR.
   
   - Service-Context-Id AVP
The Service-Context-Id AVP value sent in the Diameter message. Null is an acceptable value if the field is not expected to be present on the CCR.

- Service-Identifier AVP
  The Service-Identifier AVP value sent in the Diameter message. Null is an acceptable value if the field is not expected to be present on the CCR.

- ProductType
  The service you have defined for the event.

- EventType
  The event definition you have defined for the event.

- Version
  The version number of the event definition object that you want to apply to the event.

Define the Service-Identifier, Rating-Group, and Service-Context-Id for each event definition object defined in the mediation table.

For each received Diameter request, Diameter Gateway correlates the Service-Context-Id, Service-Identifier, Rating-Group, and Event-Timestamp AVP values (that you defined in the mediation specification) to the usage-request builder that applies to the event to be rated (for the applicable version, service, and event).

3. (Optional) In the **ValidFrom** field of the table, set a future date and time when you want Diameter Gateway to recognize a newly deployed event definition object.

For example, to have requests processed according to a new specification on April 16, 2015, you would enter:

```plaintext
|   ValidFrom
| "2015-04-16T12:01:01"
```

You can also specify a time zone. For example,

```plaintext
|   ValidFrom
| "2015-04-16T12:01:01 PST"
```

If a time zone is not sent, then the **ValidFrom** field is assumed as UTC.

4. Save the mediation specification file with a **.spec** suffix (for example, **diameter_mediation.spec**) into the directory where you save your configuration data.

5. Verify that the directory specified in the **ECE_home/occeserver/config/management/migration-configuration.xml** file is the directory where you save your configuration data.

6. Run the **configLoader** utility:

```plaintext
start configLoader
```

The utility deploys your mediation specification to the ECE cluster. Any earlier mediation specification that was in the ECE cluster is overwritten.

Any time you deploy a new version of a mediation specification into the repository, Diameter Gateway recreates its in-memory usage-request builder map.
Network Integration for Gy Interface Requests

and begins using the mapping definitions (to send requests that adhere to the specifications) provided that the validFrom date is reached.

7. Perform a rolling restart of Diameter Gateway node instances.

8. Load the mediation specification file into the ECE server by using the configLoader utility.

Network Integration for Gy Balance Query Requests

This section provides information about network integration for balance query requests using Diameter Gateway.

Diameter Gateway uses custom AVPs for querying for remaining-balance customer data; these Oracle AVPs have an ORA- prefix.

For a balance query, the CC-Request-Type AVP in the CCR must be set to 4 (EVENT_REQUEST) and the Requested-Action AVP must be set to 5 (which is an undefined value in the 3GPP standard specification).

For information about the data types for custom balance-query AVP fields, see the ECE_home/oceceserver/config/diameter/dictionary_main.xml file.

Network Integration for Gy Top-Up Requests

This section provides information about network integration for top-up requests using Diameter Gateway.

Diameter Gateway exposes a custom event request for top-up operations that does the following:

■ Credits the specified balances, optionally setting valid-from and valid-to dates
■ Optionally extends the validity of existing balances credited by the top-up
■ Return that the top-up succeeded or failed
■ Return updated balance information in the top-up response

Diameter Gateway uses custom AVPs for processing top-up requests; these Oracle AVPs have an ORA- prefix.

For a top-up, the CC-Request-Type AVP in the CCR must be set to 4 (EVENT_REQUEST) and the Requested-Action AVP must be set to 4 (which is an undefined value in the 3GPP standard specification).

Diameter Gateway uses custom AVPs for processing top-up requests; these Oracle AVPs have an ORA- prefix.

For information about the data types for custom top-up-request AVP fields, see the ECE_home/oceceserver/config/diameter/dictionary_main.xml file.

Sending Multiple-Service Credit Control (MSCC) Requests from Diameter Gateway

Diameter Gateway supports MSCC requests in which a Diameter application performs credit control for multiples services within the same session.

Diameter Gateway only supports Multiple-Service Credit Control (MSCC) requests for usage request processing (all usage-request charging operations must be contained in an MSCC group even if the request contains only a single service).
Configuring Subscriber ID Lookups

When multiple subscriber ID types come in on the CCR message, not all subscription identifiers may be provisioned for your ECE system. For example, you might have separate online charging systems for handling different subscription services. You can configure Diameter Gateway to look up customer public user identity information based only on the subscription identifier types for which you have provisioned your ECE system.

The possible customer subscription IDs that pertain to various customer services are defined by the Subscription-Id grouped AVP in the CCR message. Multiple subscription identifier types can be provided in the group's Subscription-Id-Type AVP field. The customer may have all of the following subscription identifiers for various networks on which the customer uses services: MSISDN, IMSI, SIP, NAI, PRIVATE.

For Diameter Gateway to look up customer public user identity information based on your subscription-identifier-type configuration, do the following:

1. Open your mediation specification file, diameter_mediation.spec.

   *diameter_mediation.spec* is in the directory specified by the configObjectsDataDirectory parameter in the ECE_home/oeceeserver/config/management/migration-configuration.xml file.

2. Where multiple subscription types are expected in the CCR for the event to be rated, locate the row that specifies the rating group, service identifier, and service context ID for the event.

   Your subscription-identifier-type configuration is relevant for the combination of the given Service-Context-Id, Service-Identifier, and Rating-Group AVP values specified in the row for the event to be rated.

3. In the Subscription-Id-Type column for that row, enter the subscription-identifier-type configuration of your choice.

   For each received CCR Diameter message that includes multiple subscriber ID types, Diameter Gateway uses your subscription-identifier-type configuration for looking up the public user identity.

   The subscription-identifier-type configuration options are as follows:

   - For Diameter Gateway to perform a customer lookup by using only one subscription ID type, enter the full string name of that Subscription-Id-Type.

     Enter the name exactly as it is defined in the RFC specification (in capitals) and enclose it with quotation marks.

     The possible values you can enter in the Subscription-Id-Type column for the Subscription-Id-Type are as follows (values in bold):

     - "END_USER_E164"
       The identifier is in international E.164 format (for example, MSISDN), according to the ITU-T E.164 numbering plan defined in [E164] and [CE164].

     - "END_USER_IMSI"
       The identifier is in international IMSI format, according to the ITU-T E.212 numbering plan as defined in [E212] and [CE212].

     - "END_USER_SIP_URI"
       The identifier is in the form of a SIP URI, as defined in [SIP].
For example, if you enter "END_USER_NAI" in the **Subscription-Id-Type** column for that event, Diameter Gateway uses only the subscription identifier type END_USER_NAI to perform a customer public user identity lookup for those events and ignores all other subscription identifier types that may be included in the CCR for those events.

<table>
<thead>
<tr>
<th>DiameterMediationTable</th>
<th>Service-Context-Id</th>
<th>Service-Identifier</th>
<th>Rating-Group</th>
<th>ProductType</th>
<th>EventType</th>
<th>Version</th>
<th>Subscription-Id-Type</th>
<th>ValidFrom</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:gy.service@example.com">gy.service@example.com</a></td>
<td>1</td>
<td>&quot;10&quot;</td>
<td>&quot;VOICE&quot;</td>
<td>&quot;V_USAGE&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;END_USER_NAI&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td>&quot;END_USER_NAI&quot;</td>
<td>&quot;1&quot;</td>
<td>&quot;11&quot;</td>
<td>&quot;DATA&quot;</td>
<td>&quot;D_USAGE&quot;</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>&quot;END_USER_IMSI&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For Diameter Gateway to perform a customer lookup by using a subscription ID type determined by *the order that you list subscription ID types* in the mediation specification, enter a comma-delimited list in the order that Diameter Gateway is to resolve the subscription ID type.

The following example shows a comma-delimited list for which Diameter Gateway first looks up the public user identity of the customer based on the SIP URI subscription identifier, and secondly based on the IMSI. In this case Diameter Gateway ignores all other subscription ID types that may be included in the CCR.

<table>
<thead>
<tr>
<th>DiameterMediationTable</th>
<th>Service-Context-Id</th>
<th>Service-Identifier</th>
<th>Rating-Group</th>
<th>ProductType</th>
<th>EventType</th>
<th>Version</th>
<th>Subscription-Id-Type</th>
<th>ValidFrom</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:gy.service@example.com">gy.service@example.com</a></td>
<td>1</td>
<td>&quot;12&quot;</td>
<td>&quot;DATA&quot;</td>
<td>&quot;D_USAGE&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;END_USER_SIP_URI, END_USER_IMSI&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For Diameter Gateway to perform a customer lookup by using the first subscription ID type that is read in the CCR (all other subscription ID types that may be included in the CCR are ignored), leave the **Subscription-Id-Type** column blank. This type of configuration is shown in the fourth row of the sample mediation specification.

<table>
<thead>
<tr>
<th>DiameterMediationTable</th>
<th>Service-Context-Id</th>
<th>Service-Identifier</th>
<th>Rating-Group</th>
<th>ProductType</th>
<th>EventType</th>
<th>Version</th>
<th>Subscription-Id-Type</th>
<th>ValidFrom</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:gy.service@example.com">gy.service@example.com</a></td>
<td>1</td>
<td>&quot;13&quot;</td>
<td>&quot;SMS&quot;</td>
<td>&quot;S_USAGE&quot;</td>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;2012-12-31T12:01:01 PST&quot;</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Save the mediation specification file.

5. Run the `configLoader` utility to load your mediation specification in the ECE cluster:

   ```
   start configLoader
   ```

   When your mediation specification is loaded, the earlier version of your mediation specification (that was in the ECE cluster) is overwritten and Diameter Gateway uses the configuration of the newly loaded mediation specification.
Your subscription-identifier-type configuration is used by Diameter Gateway for all usage-charging operation types: Initiate, Update, Terminate, PriceEnquiry, BalanceQuery, TopUp, Debit, and Refund.

To troubleshoot issues that may occur with your subscription-identifier-type configuration, note the following points:

- If the subscription IDs cannot be resolved correctly with the values supplied in the `diameter_mediation.spec` file, errors are logged in the Diameter Gateway log files.
- In a DEBUGGING environment, you can enable DEBUG messages in the `log4j.properties` file as shown here:
  ```
  log4j.logger.oracle.communication.brm.charging.ecegateway.diameter.framework=DEBUG
  log4j.logger.oracle.communication.brm.charging.ecegateway.diameter.gy=DEBUG
  ```
- If the subscription IDs cannot be found as configured in the `diameter_mediation.spec` file, you can expect an Errant result-code of DIAMETER_MISSING_AVP (5005) or DIAMETER_INVALID_AVP_VALUE (5004).

**Adding Custom AVPs for Usage Requests**

If you introduce custom AVPs (to introduce new ways for charging for your services), you define your custom AVPs in the `ECE_home/oceceserver/config/diameter/dictionary_custom.xml` file to define their data types.

After modifying the `dictionary_custom.xml` file, perform a rolling restart of Diameter Gateway nodes in your topology.

For AVPs that apply to usage-request processing, you add network attributes for all event attributes in the event definition so that they can be dynamically mapped to ECE payload fields by Diameter Gateway. You also put a path to your AVP field to an MSCC group block.

**Using Incremental or Cumulative Accounting for Usage Requests**

ECE supports incremental and cumulative based accounting behavior when processing usage requests.

Incremental accounting logic is used by the Diameter standard which supports Requested Service Units (RSU) and Used Service Units (USU) concepts. Incremental accounting logic indicates that the creator of the usage request enables the rating engine (ECE) to calculate in the active session the duration based on the units that have been used since the previous session update.

Cumulative accounting logic is used by the Radius standard which indicates that the creator of the usage request always supplies the full quantity (for example: duration, volume, meters, miles and so on) inclusive of all previous session requests.

When you create your usage request builder, specify the accounting behavior by using the `UnitReportingMode` ECE Java enum. When the usage request builder is instantiated, the enum indicates to ECE whether to use incremental or cumulative accounting behavior.
Important: When there are multiple RUMs and attributes ROUND_UP and ROUND_DOWN of quantity in the rate plan, Granted Service Units that are reported on all attributes may be rounded up or down based on the rate plan configuration.

For both incremental and cumulative accounting, you must set attributes for the Requested_Units and Used_Units blocks in the payloads of applicable operation types. For example, the Requested_Units block is defined for the payloads of Initiate and Update operation types and the Used_Units block is defined for the payloads of Update, Update Accounting and Terminate operation types.

When configuring incremental or cumulative quota for usage requests, the metric name (RUMs) must be the same as the attribute name. For example, when sending attribute INPUT_VOLUME on the usage request, the RUMs must be defined with the same name.

Configuring Accounting Mode for Diameter Gateway

You can configure Diameter Gateway to use both incremental-based and cumulative-based accounting logic when processing usage requests. You can perform this by specifying the accounting mode in the mediation specification file. The accounting mode indicates to Diameter Gateway whether to use incremental-based or cumulative-based accounting logic.

To configure the accounting mode for Diameter Gateway:

1. Open the Diameter mediation specification file, `diameter_mediation.spec`.
   For the location of the `diameter_mediation.spec` file, see the `configObjectsDataDirectory` parameter in the `ECE_home/occeserver/config/management/migration-configuration.xml` file.

2. For each event to be rated (in each row), specify the accounting mode in the `UnitReportingMode` column. Valid values are:
   - INCREMENTAL
   - CUMULATIVE

   For example:
   ```
   Service-Context-Id | Service-Identifier | Rating-Group | ProductType | EventType | Version | Subscription-Id-Type | ValidFrom   | UnitReportingMode |
   "gy.service@example.com" | "1" | "10" | "VOICE" | "V_USAGE" | 1.0 | "" | "2014-3-31T12:01:01 PST" | "INCREMENTAL" |
   "gy.service@example.com" | "11" | "DATA" | "D_USAGE" | 1.0 |
   "" | "2014-3-31T12:01:01 PST" | "CUMULATIVE" |
   ```

   The default accounting mode is Incremental. If you specify null or if you do not specify a mode in the `UnitReportingMode` column, Diameter Gateway uses the default accounting mode when processing usage requests. This supports backward compatibility.

   Your accounting mode configuration is applicable for the combination of the given values specified in the row for the event to be rated. You can also configure different accounting modes for the same product and event type combination.

3. Save and close the file.
4. Run the `configLoader` utility to load your mediation specification in the ECE cluster:

   ```
   start configLoader
   ```

   When the mediation specification is loaded, the earlier version of your mediation specification (that was in the ECE cluster) is overwritten and Diameter Gateway uses the configuration of the newly loaded mediation specification.

5. Change directory to the `ECE_home/octeserver/bin` directory.

6. Start ECC:

   ```
   ./ecc
   ```

7. Do one of the following:

   - If the Diameter Gateway instance is *not* running, start it.
     
     The instance reads its configuration information by name at startup.
   
   - If the Diameter Gateway instance is running, stop and restart it.
     
     For information about stopping and starting Diameter Gateway instances, see the discussion about starting and stopping ECE in *BRM System Administrator’s Guide*.

---

### Configuring Notifications for Diameter Gateway

To enable Diameter Gateway to consume notifications from the ECE notification queue:

**Note:** The following steps assume that ECE is installed and that required ECE post-installation tasks are completed.

1. On the Oracle WebLogic server, verify that the ECE event notification queue (a JMS topic) was created.

2. In ECE, verify that JMS credentials were configured correctly so that ECE can publish notifications to the ECE notification queue.

   See "About ECE Notifications" for information about configuring JMS credentials in ECE.

3. Access the ECE configuration MBeans:

   a. Log on to the driver machine.
   
   b. Start the ECE charging servers (if they are not started).
   
   c. 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/octeserver/config/eceTopology.conf` file.
   
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   
   e. In the editor's MBean hierarchy, find the ECE configuration MBeans.

4. Expand the **ECE Configuration** node.

5. Expand **charging.notification**.

6. Expand **Attributes**.
7. Set the appropriate type of notification (such as top-up or advice of charge) to the appropriate value. See "About ECE Notifications".

**Configuring Alternative Diameter Peers for Notifications**

Peer details are configured in Diameter Gateway to filter and route the notifications for the peers to which Diameter Gateway connects. Each Diameter Gateway instance listens to a registered peer. The connection is initiated from the peer to send the respective notifications. If a Diameter Gateway instance sends a notification message to its peer and the peer is unavailable or the peer fails after receiving the notification message, the Diameter Gateway instance retains the notification messages and sends them to another available peer based on your alternative-peer configuration.

To configure alternative Diameter peers for notifications:
1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/ecdTopology.conf` file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE configuration MBeans.
2. Expand the ECE Configuration node.
3. Expand `charging.diameterGatewayPeerConfigurations`.
4. Expand Attributes.
5. For each peer connected to the Diameter Gateway, configure alternative peers by specifying values for the following attributes:
   - `peerName`: Enter the name of the Diameter peer.
   - `alternatePeerName`: Enter the name of the alternative peer for the specified Diameter peer. You can specify two alternative peers for each Diameter peer.
6. Change directory to the `ECE_home/oceceserver/bin` directory.
7. Start the Elastic Charging Controller:
   ```
   ./ecc
   ```
8. Do one of the following:
   - If the Diameter Gateway instance is *not* running, start it.
   - If the Diameter Gateway instance is running, stop and restart it.

**Viewing Active Diameter Peers**

To view all the active diameter peers:
1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
d. Connect to a Diameter Gateway instance node set to start CohMgt = true in the ECE_home/oceceserver/config/eceTopology.conf file.

The eceTopology.conf file also contains the host name and port number for the node.
e. In the editor’s MBean hierarchy, expand the DiameterGateway node.

2. Expand PeerConnectionsTracker.
3. Expand Attributes.
4. Click the peerConnections value.

The diameter peers that are active (which are currently connected to the specific Diameter Gateway instance) are displayed.

**Handling Requests When Charging Servers Are Unavailable**

Diameter Gateway can be configured to use a degraded mode operating mode if the Elastic Charging Server (charging server nodes) become unavailable.

Diameter Gateway actively monitors the health of the Elastic Charging Server. If the Elastic Charging Server becomes unavailable (such as going below the charging-server health threshold), Diameter Gateway sends the DIAMETER_TOO_BUSY result code response to network requests.
Connecting ECE to a RADIUS Client

This document describes how to use Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) RADIUS Gateway for authenticating access requests and processing accounting requests from RADIUS clients, such as terminal servers or network access servers (NAS).

Topics in this document:

- Overview of Authentication and Accounting Using RADIUS Gateway
- About RADIUS Gateway Authentication
- Authenticating Access Requests by Using PAP
- Authenticating Access Requests by Using CHAP
- Authenticating Access Requests by Using EAP
- Loading Data Keys Extracted from BRM into ECE
- Customizing the RADIUS Data Dictionary
- Loading the RADIUS Mediation Specification Data
- About Mapping RADIUS Network Attributes to Event Attributes
- About RADIUS Gateway Accounting
- About Accounting-Start and Accounting-Stop Requests
- About Accounting-On and Accounting-Off Requests
- About Accounting-Interim-Update Requests

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

Overview of Authentication and Accounting Using RADIUS Gateway

You use RADIUS Gateway for authenticating access requests and processing accounting requests for online charging when your customers use your terminal server or NAS to connect to ECE. RADIUS Gateway does the following when it receives a request from the RADIUS client:

1. Translates the request into an ECE request.
About RADIUS Gateway Authentication

2. Submits the ECE request to the ECE server.
3. Receives the ECE response from the ECE server and translates it into a RADIUS message response.
4. Sends the RADIUS message response to the RADIUS client.

The following steps summarize how to set up ECE for authentication and accounting using RADIUS Gateway:

1. Add additional RADIUS Gateway nodes required for your topology and configure each instance.
2. (Optional) Customize the RADIUS data dictionary to include custom vendor-specific attributes.
3. Load your event definitions from PDC into ECE.
4. Load customer data and data keys from BRM into ECE.
5. Load the RADIUS mediation specification data.
6. Map RADIUS network attributes to event attributes.
7. Start the RADIUS Gateway nodes.

When the RADIUS Gateway nodes start, they automatically join the Coherence cluster gaining access to ECE caches and invocation services that it uses to send requests to ECE.

About RADIUS Gateway Authentication

RADIUS Gateway supports the following authentication mechanisms for querying the ECE server and authenticating access requests:

- **Password Authentication Protocol (PAP).** An authentication protocol that uses the user name and password to validate users. See "Authenticating Access Requests by Using PAP" for more information on how RADIUS Gateway performs the PAP authentication.

- **Challenge Handshake Authentication Protocol (CHAP).** An authentication protocol that authenticates a user to a network entity; for example, the Web. This protocol ensures that the server sends a challenge to the RADIUS client after the RADIUS client establishes a network connection to access the Web server. See "Authenticating Access Requests by Using CHAP" for more information on how RADIUS Gateway performs the CHAP authentication.

- **Extensible Authentication Protocol (EAP).** An authentication protocol that supports multiple authentication mechanisms for authenticating network access; for example, EAP-Message Digest 5 (MD5). See "Authenticating Access Requests by Using EAP" for more information on how RADIUS Gateway performs the EAP authentication.

Authenticating Access Requests by Using PAP

You use PAP to authenticate access requests based on the clear-text user name and user password. Only Access-Request requests are considered for PAP authentication; other messages are ignored. The PAP authentication is performed based on the User-Name and User-Password AVP values in the Access-Request request.

The PAP authentication process is as follows:

1. RADIUS Gateway receives the Access-Request request from the RADIUS client.
2. RADIUS Gateway authenticates the RADIUS client using the `sharedsecret` password that you provided during installation.

   **Note:** If RADIUS clients are represented by using an IP address range, ensure that all the RADIUS clients within the IP address range use the same `sharedsecret` password.

3. RADIUS Gateway translates the Access-Request request into an ECE query.

4. RADIUS Gateway sends the query with the User-Name AVP value to the ECE server to validate the user name.

5. If the user name is not found in the ECE server, the ECE server returns a failed response. RADIUS Gateway translates the failed response into the `Access-Reject` message and returns it to the RADIUS client.

6. If a match for the user name is found, RADIUS Gateway sends a query with the User-Password AVP value in the request to the ECE server to validate the user password.

7. The ECE server returns a password. If the password is encrypted, RADIUS Gateway decrypts the password using a data key loaded from BRM before validating the user password. The data key to be used is identified using the key ID in the password returned by the ECE server.

8. If the user password in the ECE server matches the User-Password AVP value in the query, the ECE server returns a success response. RADIUS Gateway translates the success response into the `Access-Accept` message and returns it to the RADIUS client.

9. If the user password does not match, the ECE server returns a failed response. RADIUS Gateway translates the failed response into the `Access-Reject` message and returns it to the RADIUS client.

Sample Access-Request Request for PAP Authentication

```
Code: Access-Request(1)
Identifier: 0
Length: 120
Authenticator: 0x7D564C041FD183A4DBA037E03E3244F3
User-Name: alias#1006
User-Password: 0x41FD183A4335037E03E3244F3123123
NAS-IP-Address: 128.1.2.3
NAS-Port-Type: 1034
Service-Type: 2
```

**Authenticating Access Requests by Using CHAP**

You use CHAP to authenticate access requests by validating the identity of the RADIUS client using Access-Challenge messages. The CHAP authentication is performed based on the CHAP-Password and CHAP-Challenge AVP values in the Access-Request request. RADIUS Gateway uses the State AVP value in the Access-Request request or the `noOfChallenges` value that you configured in ECE to carry out the number of Access-Challenge messages for a given authentication session.

At any time in a given authentication session, RADIUS Gateway can also request the RADIUS client to send an Access-Challenge message. The CHAP authentication uses encrypted passwords for authentication and the Access-Challenge message can be
requested for authentication by RADIUS Gateway at any time. Therefore, the CHAP authentication process is considered more secure than the PAP authentication process.

The CHAP authentication process is as follows:

1. The RADIUS client encrypts a clear-text user password by using the CHAP identifier and CHAP-Challenge AVP value and sends it in the CHAP-Password AVP in an Access-Request request.

2. RADIUS Gateway authenticates the RADIUS client using the sharedsecret password that you provided during installation.

   **Note:** If RADIUS clients are represented by using an IP address range, ensure that all the RADIUS clients within the IP address range use the same sharedsecret password.

3. RADIUS Gateway translates the Access-Request request into an ECE query.

4. RADIUS Gateway sends the query with the User-Name AVP value to the ECE server to validate the user name.

5. If the user name is not found in the ECE server, the ECE server returns a failed response. RADIUS Gateway translates the failed response into the Access-Reject message and returns it to the RADIUS client.

6. If a match for the user name is found, the ECE server returns the password associated with the user name. If the password is encrypted, RADIUS Gateway decrypts the password into a clear-text password using the data key loaded from BRM before validating the password. The data key to be used is identified using the key ID in the password returned by the ECE server.

7. RADIUS Gateway generates an MD5 hash value using the password, CHAP-Challenge AVP, and CHAP identifier (which is the first byte of the CHAP-Password AVP), and compares it with the CHAP-Password AVP value in the Access-Request request.

8. If the values do not match, RADIUS Gateway returns a failed response. RADIUS Gateway returns an Access-Reject message to the RADIUS client.

9. If the MD5 hash value and the CHAP-Password AVP value match, RADIUS Gateway returns a success response.

10. RADIUS Gateway sends an Access-Challenge message to the RADIUS client. RADIUS Gateway uses the State AVP value in the Access-Request request to determine the number of Access-Challenge messages to be sent to the RADIUS client. For example, if the State AVP value is 0, RADIUS Gateway directly returns the Access-Accept message. If the State AVP value is 1, RADIUS Gateway sends only one Access-Challenge message to the RADIUS client.

11. If the State AVP value is null or if the value is not set, RADIUS Gateway calculates a random number between one and the maximum number of challenges configured in ECE and sends the Access-Challenge messages to the RADIUS client.

12. The RADIUS client responds with a value calculated through the MD5 hash function.

13. RADIUS Gateway checks the response against its calculation of the expected hash value.
14. If the values match, RADIUS Gateway repeats the Access-Challenge messages based on the State AVP value or the number calculated by RADIUS Gateway.

15. If the values do not match, RADIUS Gateway returns an Access-Reject message to the RADIUS client.

Sample Access-Request Request for CHAP Authentication

Code: Access-Request(1)
Identifier: 0
Length: 144
Authenticator: 0x7D564C041FD183A4DBA037E03E3244F3
CHAP-Password: 0x423423432412ADA123CC1123124123
Chap-Challenge="0xFBFCE5676F9433682718EF97F8AB24900"
NAS-IP-Address: 127.0.0.8
State: 0
NAS-Port-Type: 1816
Service-Type: 2

Authenticating Access Requests by Using EAP

You use EAP to authenticate users using different authentication mechanisms. EAP includes password-based authentication methods and secure certificate-based authentication methods. The EAP authentication is performed based on the EAP-Message AVP value in the Access-Request request. RADIUS Gateway supports the following EAP authentication methods:

- **EAP-Tunneled Transport Layer Security (TTLS)**. Authentication between RADIUS Gateway and the RADIUS client uses a secured connection in two phases. In the first phase, RADIUS Gateway and the RADIUS client exchange authentication certificates for establishing the secured connection. In the second phase, RADIUS Gateway authenticates the RADIUS client by using different authentication mechanisms, such as EAP-PAP, EAP-CHAP, and EAP-MD5. These authentication mechanisms use the attributes in the Access-Request request to perform the authentication. You can also configure a custom EAP authentication mechanism by using the RADIUS Gateway extension points.

- **EAP-Non-TTLS**. Authentication between RADIUS Gateway and the RADIUS client uses a configured list of EAP authentication mechanisms. The EAP-Non-TTLS authentication process is as follows:

1. RADIUS Gateway performs a standard check on the Access-Request request received from the RADIUS client.

2. RADIUS Gateway sends the EAP-Type AVP in the Access-Challenge message that contains the value corresponding to the first EAP type configured.

3. If the RADIUS client returns NAK, RADIUS Gateway sends the next EAP type in the configured list in the Access-Challenge message. RADIUS Gateway continues this process until the RADIUS client responds with an Access-Accept message or until the end of the configured list is reached. In that case, RADIUS Gateway sends an Access-Reject message.

RADIUS Gateway, by default, supports only the EAP-MD5 authentication mechanism in the EAP-Non-TTLS method. To use a different authentication method, use the CustomAuth and CustomEAPChallenge extension points. The CustomEAPChallenge extension point sends the initial EAP challenge to the RADIUS client. The CustomAuth extension point performs the authentication and returns the authentication result. Based on the result received, RADIUS Gateway sends the appropriate RADIUS response to the RADIUS client.
Loading Data Keys Extracted from BRM into ECE

As part of the initial load of customer data into ECE, Customer Updater loads data keys into ECE. When RADIUS Gateway is started, the data keys are decrypted using the BRM root key in the Oracle wallet file and stored in the memory with a data key ID for each data key. These data keys are used for decrypting the passwords in authentication responses from ECE.

When you add or modify a data key in BRM, you must load the newly added or modified data key extracted from BRM into ECE.

To load data keys extracted from BRM into ECE:

1. Access the Customer Updater MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started) and Customer Updater.
      See "Starting and Stopping ECE" in BRM System Administrator’s Guide.
   c. Connect to the Customer Updater 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.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, expand the UpdateEventhandler node.

2. Expand Update.
4. Select updateDataKeys.
5. Click the updateDataKeys button.

The newly added or modified data keys extracted from BRM are loaded into ECE.

Customizing the RADIUS Data Dictionary

This section covers customizing the RADIUS data dictionary.

About the RADIUS Data Dictionary

The data dictionary includes a list of AVPs that are used by RADIUS Gateway to perform authentication and accounting operations. The RADIUS data dictionary
contains the standard AVPs that are prescribed in RADIUS Request for Comments (RFC) 2865, 2866, and 2869, and also some sample vendor-specific attributes. You can use the sample vendor-specific attributes as a template for adding custom vendor-specific attributes. The default location of the RADIUS data dictionary file is \texttt{ECE\_home/config/radius/radiusDictionary.xml}.

\textbf{Important:} Do not remove, rename, or move the RADIUS data dictionary file to a different location.

\section*{Creating a Custom Data Dictionary}

You can create a custom data dictionary file by using the \texttt{ECE\_home/config/radius/radiusDictionary.xml} file as a template. The default location for your custom data dictionary file is \texttt{ECE\_home/config/radius/custom/dictionary\_file}, where \texttt{dictionary\_file} is the name of your custom data dictionary file. You can add new vendor-specific attributes to your custom data dictionary file. See "Adding Custom Vendor-Specific Attributes".

\section*{Selecting a RADIUS Data Dictionary When Using Different NAS Vendors}

If you must use NAS servers from multiple vendors, you have the following options:

- If your NAS is RFC 2865 compliant, you can use the RFC2865 data dictionary. This is the preferred solution. Update the dictionary file with any vendor-specific attributes associated with the NAS.

- If your NAS is not RFC 2865 compliant, you can use the RADIUS data dictionary files for adding vendor-specific attributes. See "Adding Custom Vendor-Specific Attributes" for more information.

\section*{Adding Custom Vendor-Specific Attributes}

In special cases, where you are using NAS servers from multiple vendors, you must add the vendor attribute and code in your custom data dictionary file.

The syntax for adding a vendor-specific attribute is:

\begin{verbatim}
<?xml version="1.0" encoding="UTF-8"?>
<dictionary
   schemaLocation="radiusDictionary.xsd"
   <vendor value="vendor\_ID" name="vendor\_name"/>
   </attribute
   name="attribute\_name"
   vendor="vendor\_name"
   syntax="data\_type"
   code="attribute\_ID"/>
</dictionary>
\end{verbatim}

Table 13–1 lists the vendor-specific attribute values and descriptions.

\begin{table}[h!]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Parameters} & \textbf{Description} \\
\hline
\texttt{vendor\_ID} & Number used to identify the NAS or gateway vendor. These numbers are assigned by the Internet Advisory Board (IAB). See your vendor’s documentation for details. Some common vendor identification numbers are: \\
& \hspace{1cm} 9 (Cisco) \\
& \hspace{1cm} 10415 (3GPP) \\
& \hspace{1cm} 2636 (Juniper) \\
\hline
\end{tabular}
\caption{Vendor-specific Attribute Values}
\end{table}
To load the RADIUS mediation specification data:

1. Create a mediation specification file or open the sample RADIUS mediation specification file.

   A sample mediation specification file (`ECE_home/oceceserver/sample_data/config_data/specifications/ece_simple`) is available.
2. Load the pricing data from PDC into ECE.
   For every event definition, which contains charging operation types (for example, Initiate) loaded into ECE from PDC, ECE generates network mapping files.

3. Add a row (in the table) for each new product to be rated that specifies the following information:
   - **Service-Identifier AVP**
     A unique identifier of the service. The Service-Identifier AVP value is sent by the RADIUS request. “null” is valid if the field is not expected to be present in the request.
   - **ProductType**
     The product type that you have defined for the event in its associated request specification.
   - **EventType**
     The event type that you have defined for the event in its associated request specification.
   - **Version**
     The version number of the request specification that you want to apply to the event.
   - **ValidFrom**
     A future date and time when you want RADIUS Gateway to recognize a newly deployed request specification.
     To have requests processed according to a new specification, you would enter:
     \[ yyyy-mm-ddThh:mm:ss [timezone] \]
     If timezone is not specified, it defaults to UTC.
   - **Network-Mapping-FileName**
     The name of the network mapping file generated for the product and event combination.

See Example 13–1 for a sample entry in the RADIUS mediation specification file.

4. Open the \texttt{ECE_home/oceceserver/config/management/migration-configuration.xml} file.

5. Search the `configObjectsDataDirectory` parameter and copy the value. For example:
   \[
   \texttt{configObjectsDataDirectory} = \texttt{ECE_home/oceceserver/sample_data/config\_data}
   \]

6. Save the mediation specification file to that same directory.

7. Load the file into the ECE server by running the following command:
   \[
   \texttt{start configLoader}
   \]

---

**Important:** Create only one RADIUS mediation specification file to represent the mediation specification for RADIUS Gateway.
The utility loads the RADIUS mediation specification data to the ECE cluster. The `configLoader` utility uses the location in the `configdata` parameter for loading the data. As mediation specification files have same names, so any existing RADIUS mediation specification data in the ECE cluster is overwritten.

Example 13–1 Sample RADIUS Mediation Specification Entry

```plaintext
RadiusMediationTable {
  Service-Identifier | ProductType | EventType | Version | ValidFrom | Network-Mapping-FileName |
  "1" | "TelcoGprs" | "EventDelayedSessionTelcoGprs" | 2.0 | "2010-12-31T12:01:01 PST" | "EventDelayedSessionTelcoGprs_TelcoGprs.xml" |
}
```

When you load the RADIUS mediation specification data into the ECE cluster, RADIUS Gateway re-creates its in-memory usage-request builder map and uses the mapping definitions to send requests to ECE.

About Mapping RADIUS Network Attributes to Event Attributes

To process requests from RADIUS clients, you map network attributes from RADIUS clients to the corresponding event attributes in ECE. You do this by editing the network mapping file. When you load the pricing data from PDC into ECE, ECE generates the network mapping file for each product and event combination. Some default network mappings are already pre-configured in the files generated by ECE. You can update the default values in these files.

RADIUS Gateway uses this mapping in ECE to process requests by dynamically mapping the values of the network attributes in the RADIUS request to the corresponding event attributes in ECE.

Mapping RADIUS Network Attributes to Event Attributes

If you add or remove an event attribute from the event definition in PDC, you have to add or remove the corresponding network attributes in ECE. You do this by editing the network mapping file in ECE.

Before you map the attributes, load the RADIUS mediation specification file. See "Loading the RADIUS Mediation Specification Data" for more information.

To map network attributes to event attributes:

1. Load the pricing data from PDC into ECE.

   Mapping files will be automatically generated when the pricing data is published from PDC to ECE.

   For every event definition, which contains charging operation types (for example, `Initiate`) loaded into ECE from PDC, ECE generates the network mapping files. The network mapping files are stored in the directory specified by the `configObjectsDataDirectory` parameter in the `ECE_Home/occeserver/config/management/migration-configuration.xml` file.

   A sample network mapping file is available in the `(ECE_Home/occeserver/sample_data/config_data/specifications/ece_end2end/network_mapping)` directory. You can use this as a reference for mapping the attributes.

2. Open a network mapping file in a text editor.

3. Ensure that the ORIGIN_NETWORK event attribute is added as a top-level attribute in the network mapping file.
4. Map the network attributes to the event attributes by doing the following:
   a. Search for the event attribute that you want to map to the network attribute.
   b. Add the following entry:

   `<networkField>NetworkAttribute</networkField>`

   where NetworkAttribute is the attribute of the requests received from RADIUS clients.
   For example:

   ```xml
   <attributeMapping type="RadiusMediationEntries">
     <attribute>
       <name>TERMINATE_CAUSE</name>
       <networkField>Acct-Terminate-Cause</networkField>
     </attribute>
   </attributeMapping>
   ```

5. Save and close the file.

   **Important:** Verify that the name of this network mapping file is specified in the RADIUS mediation specification file.

6. Load the network mapping data by doing one of the following:
   - If RADIUS Gateway is running, run the following command:
     ```
     start configLoader loadNetworkMapping
     ```
   - If RADIUS Gateway is *not* running, run the following commands:
     ```
     start customerUpdater
     start radiusGateway
     ```

   The network mapping data is loaded into the ECE cluster. Any existing network mapping data available for the product and event specification in the ECE cluster is overwritten. ECE is now in a usage-processing state, where it can accept requests from RADIUS Gateway.

   When you load the network mapping into the ECE cluster, RADIUS Gateway re-creates its in-memory usage-request builder map and begins using the latest mapping definitions to send requests to ECE.

**About RADIUS Gateway Accounting**

RADIUS Gateway processes the accounting requests to track information about customer usage. For example, RADIUS Gateway tracks when customers log in to a network for using the services and when customers log out of the network. The information tracked by RADIUS Gateway is used for statistical purposes, network monitoring, and billing the customers based on the duration of the sessions or the type of services used.

To track customer usage information, RADIUS Gateway uses the network mapping definitions in ECE and maps the accounting requests received from the RADIUS clients to the usage requests with the corresponding operation types configured in ECE.
See the following topics for information on the different types of accounting requests received from the RADIUS clients:

- About Accounting-Start and Accounting-Stop Requests
- About Accounting-On and Accounting-Off Requests
- About Accounting-Interim-Update Requests

The RADIUS Gateway accounting process is as follows:

1. At the start of accounting or the start of a user session, the RADIUS client sends an accounting request to RADIUS Gateway. The Acct-Status-Type AVP value in the request indicates the start of accounting or start of a session for the user.

2. RADIUS Gateway processes the request and records the information as either an accounting-on record or an accounting-start record in ECE, based on the accounting request received.

3. RADIUS Gateway returns an Accounting-Response message to the RADIUS client to acknowledge the accounting-start or accounting-on request.

4. While the session is active, the RADIUS client sends periodic updates on the data usage to RADIUS Gateway through accounting requests with the Acct-Status-Type AVP set to Interim-Update.

5. RADIUS Gateway processes the requests and records the information as accounting-interim-update records in ECE.

6. RADIUS Gateway returns Accounting-Response messages to the RADIUS client to acknowledge the interim-update requests.

7. At the end of accounting or the end of the user session, the RADIUS client sends an accounting request that contains the Acct-Status-Type AVP value indicating the end of accounting or the end of the user session.

8. RADIUS Gateway processes the request and records the information as either an accounting-off record or an accounting-stop record in ECE, based on the accounting request received.

9. RADIUS Gateway returns an Accounting-Response message to the RADIUS client to acknowledge the accounting-off or accounting-stop request. At any time, if the RADIUS client does not receive an Accounting-Response message, it continues to send accounting requests until it receives a response.

### About Accounting-Start and Accounting-Stop Requests

When a client is configured to use RADIUS accounting, the RADIUS client sends an Accounting-Start request, which specifies the start of a session for delivering a service, and an Accounting-Stop request, which specifies the end of the session that was started for delivering a service, to RADIUS Gateway. The Accounting-Start request describes the type of service being delivered and the user who is using that service. The Accounting-Stop request describes the type of service that was delivered. The Accounting-Stop request might also contain statistics, such as elapsed time, input and output octets, or input and output messages. The RADIUS client uses the Acct-Status-Type AVP to specify the start of a session and to specify the end of a session.

The following AVPs must be present in an Accounting-Start or Accounting-Stop request:

- Acct-Session-Id
Note: The Accounting-Start and Accounting-Stop requests for a given session must have the same Acct-Session-Id AVP.

- Acct-Status-Type
- NAS-IP-Address or NAS-Identifier
- User-Name
- The AVP that you configured to derive the service in ECE by using the avpName and vendorId parameters.

For an Accounting-Start request, the Acct-Status-Type AVP must be set to 1. When a RADIUS client sends the Accounting-Start request, the RADIUS client indicates that the user service session has started. When RADIUS Gateway receives the Accounting-Start request, RADIUS Gateway records the information contained in the request for billing purpose and returns the Accounting-Response message to the RADIUS client.

For an Accounting-Stop request, the Acct-Status-Type AVP must be set to 2. When a RADIUS client sends the Accounting-Stop request, the RADIUS client indicates that the user service session has ended. When RADIUS Gateway receives the Accounting-Stop request, RADIUS Gateway records the information contained in the request for billing purposes and returns the Accounting-Response message to the RADIUS client.

The RADIUS client continues to send the Accounting-Start or Accounting-Stop requests until it receives the Accounting-Response message.

Sample Accounting-Start Request for Accounting

```
[Code: Accounting-Request(4)
 Identifier: 0
 Length: 94
 Authenticator: 0x30303030303030303030303030303030
 Acct-Session-Id: 123456
 Acct-Status-Type: 1
 NAS-Identifier: telco.org
 User-Name: alias#5000
 Service-Type: 1]
```

Sample Accounting-Stop Request for Accounting

```
[Code: Accounting-Request(4)
 Identifier: 1
 Length: 87
 Authenticator: 0x30303030303030303030303030303030
 Acct-Session-Id: 123456
 Acct-Status-Type: 2
 Acct-Input-Octets: 10
 Acct-Output-Octets: 18
 Acct-Session-Time: 200
 NAS-Identifier: telco.org
 User-Name: alias#5000
 Service-Type: 1]
```
About Accounting-On and Accounting-Off Requests

When a client is configured to use RADIUS accounting, the RADIUS client sends an Accounting-On request, which specifies the start of accounting, and an Accounting-Off request, which specifies the end of accounting, to RADIUS Gateway. The RADIUS client uses the Acct-Status-Type AVP to specify the start of accounting and to specify the end of accounting.

The following AVPs must be present in an Accounting-On or Accounting-Off request:

- Acct-Status-Type
- NAS-IP-Address or NAS-Identifier
- The AVP that you configured to derive the service in ECE by using the avpName and vendorId parameters.

For an Accounting-On request, the Acct-Status-Type AVP must be set to 7. When a RADIUS client sends the Accounting-On request, the RADIUS client indicates that it is ready for service. When RADIUS Gateway receives the Accounting-On request, RADIUS Gateway closes or terminates any open accounting session associated with that RADIUS client before the RADIUS client indicated it was ready for service.

For an Accounting-Off request, the Acct-Status-Type AVP must be set to 8. When a RADIUS client sends the Accounting-Off request, the RADIUS client indicates that it is going out of service. When RADIUS Gateway receives the Accounting-Off request, RADIUS Gateway closes or terminates all the open accounting sessions associated with that RADIUS client.

Sample Accounting-On Request for Accounting

```
[Code: Accounting-Request(4)
 Identifier: 4
 Length: 68
 Authenticator: 0x30303030303030303030303030303030
 Acct-Session-Id: 131
 Acct-Status-Type: 7
 NAS-Identifier: telco.org
 User-Name: alias#5000
 Service-Type: 1

[Code: Accounting-Response(5)
 Identifier: 4
 Length: 20
 Authenticator: 0x00000000000000000000000000000000]
```

Sample Accounting-Off Request for Accounting

```
[Code: Accounting-Request(4)
 Identifier: 5
 Length: 68
 Authenticator: 0x30303030303030303030303030303030
 Acct-Session-Id: 131
 Acct-Status-Type: 8
 NAS-Identifier: telco.org
 User-Name: alias#5000
```
About Accounting-Interim-Update Requests

During a session, the RADIUS client periodically sends Accounting-Interim-Update requests, which specify the current session duration and current data usage, to RADIUS Gateway. The RADIUS client uses the Acct-Status-Type AVP to specify the interim update.

The following AVPs must be present in an Accounting-Interim-Update request:

- Acct-Session-Id
- Acct-Status-Type
- NAS-IP-Address or NAS-Identifier
- User-Name
- The AVP that you configured to derive the service in ECE by using the `avpName` and `vendorId` parameters.

When periodic Accounting-Interim-Update requests are sent for the same active session, the identifier in each Accounting-Interim-Update request must be unique. If the identifier is the same, RADIUS Gateway considers only the first request received with that identifier and ignores other requests.

For an Accounting-Interim-Update request, the Acct-Status-Type AVP must be set to 3. When a RADIUS client sends the Accounting-Interim-Update request, the RADIUS client indicates that the session is active. When RADIUS Gateway receives the Accounting-Interim-Update request, RADIUS Gateway records the information contained in the request for billing purposes and returns the Accounting-Response message to the RADIUS client.

The RADIUS client continues to send Accounting-Interim-Update requests until it receives the Accounting-Response message.

Sample Accounting-Interim-Update Request for Accounting

[Code: Accounting-Request(4)
Identifier: 0
Length: 95
Authenticator: 0x30303030303030303030303030303030
Acct-Session-Id: 123456
Acct-Status-Type: 3
Acct-Input-Octets: 6
Acct-Output-Octets: 10
NAS-Identifier: telco.org
User-Name: alias#5000
Service-Type: 1]
Radius Response Packet
[Code: Accounting-Response(5)
Identifier: 0
Length: 20
Authenticator: 0x00000000000000000000000000000000
This document describes how to implement policy-driven charging in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:
- About Policy-Driven Charging
- Configuring Policy-Driven Charging
- Configuring Breach Tolerance for Policy-Tier Thresholds
- About Integrating Policy Clients with ECE
- About the ECE Sy and Sp Interface
- About Calculating Maximum Authorization for Policy-Driven Charging Sessions
- About the Policy Management API

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

About Policy-Driven Charging

Policy-driven charging enables you to track a subscriber’s service usage and, based on that usage, change the customer’s quality of service (QoS) during online charging.

For example, a subscriber purchases a package for a certain QoS to download video content. The subscriber makes his choice from one of many packages that you have configured with gradations in the QoS based on usage amounts in megabytes, such as 100-150, 150-200, and 200-250 megabytes. When the subscriber starts downloading video content from the network, you can track the quantity of megabytes that the subscriber downloads during that session. When you find that the downloaded quantity crosses the upper threshold set for the selected QoS (for example 150 megabytes), you can use Oracle Communications Billing and Revenue Management’s (BRM’s) policy-driven charging to make a seamless change in the policy set for the subscriber’s (video downloading) session on the network and allow a change to the QoS from the current to the next level.

ECE supports policy-driven charging. Policy-driven charging implements network, customer, and service policies that can be used by service providers for improving customer experience and for making efficient use of network resources. Service
providers can use policies for various reasons, such as to control data usage, set quality of service (QoS), allocate amounts of bandwidth to each service, enforce parental controls, implement charging rules, and so on.

When Policy and Charging Rules Function (PCRF) policy clients are integrated with ECE, ECE acts as the Subscriber Profile Repository (SPR) because it stores the customer profile information used by the PCRF. ECE offers a combined Sp and Sy interface, which the PCRF uses to retrieve customer preferences and policy counter information.

Policies can be service and network aware. Network aware policies can be created for specific access technologies where the network condition can dynamically alter prices. Service aware policies can be created to provide control over how a customer consumes network resources.

To support policy-driven charging, ECE exposes the following information in its in-memory data grid to policy clients (such as Diameter Gateway or your third party network mediation software for online charging). Policy clients use the ECE policy management APIs to retrieve the information and send it to the PCRF:

- **Policy label information**

  Policy enforcement programs on the PCRF use policy labels such as status labels. For example, a quality of service label might be defined as **normal-QoS** or **low-QoS** as shown below:

  ```xml
  <policy_label>
  <label>Basic Subscription</label>
  <resource_code>MBU</resource_name>
  <resource_id>100012</resource_id>
  <unit>megabyte</unit>
  <tiers>
    <tier>
      <range_start>0</range_start>
      <range_end>300</range_end>
      <status_label>normal-QoS</status_label>
    </tier>
    <tier>
      <range_start>301</range_start>
      <status_label>low-QoS</status_label>
    </tier>
  </tiers>
  </policy_label>
  </policy_labels>
  ``

  Policy label information is stored in the policy specification (offer profiles in BRM) in PDC. ECE loads this information into its data grid when it loads pricing data from PDC.

- **Policy counter information**

  The Sy interface of the ECE Java policy API transfers policy counter information from ECE to the policy client; it provides policy counter status reporting and policy counter status change notifications.

  Policy counters track a customer’s usage of a service. For example, ECE tracks how many megabytes of data a subscriber has downloaded. The policy client retrieves the policy counters from ECE and sends them to the PCRF for evaluation.

- **Subscriber preferences information**

  ECE stores subscriber preferences associated with how the customer would like to receive policy notifications. Policy clients can retrieve this data from ECE using the Sp interface of the ECE Java policy API. Subscriber preferences such as the following can be sent to the PCRF:
About Policy-Driven Charging

Configuring Policy-Driven Charging

– Customer’s charging related information (for example, if the customer purchased a Gold, Platinum, or Bronze package)
– Customer’s preferred channel for receiving notifications (for example, email or SMS)
– Customer’s language

To support policy-driven charging, ECE publishes policy notifications. Policy specifications can store threshold definitions for specific balances. ECE can use the threshold definitions to publish notifications when thresholds are breached (SpendingLimit notifications). Also, when the subscriber preferences of a customer changes, ECE publishes notifications with the new or changed preference information (SubscriberPreference notifications). ECE sends notifications to the JMS notification queue. The policy client listens on the queue and uses the data in the notifications to send Sy and Sp messages to the PCRF.

ECE publishes policy notifications only for charge offers that have active policy sessions. When the policy client (such as Diameter Gateway) initiates policy sessions, it subscribes for receiving the policy notifications on behalf of the PCRF.

When a customer purchases a new charge offer, the PCRF re-queries the policy label and policy counter (Sy data) to subscribe to the additional counters associated with the new charge offer.

For more information about policy notifications, see “ECE Notification Reference”.

About Group-Based Policy-Driven Charging

ECE supports group-based policy-driven charging where a policy counter is shared by a group of users, enabling the PCRF to define rules for a group of users.

Group-based policy-driven charging in ECE works as follows:

■ The owner of a discount sharing group shares a policy counter.
■ A shared discount offer is used to impact the shared policy counter.
■ The shared discount is associated with a policy specification that defines policy counter thresholds.
■ When a policy threshold is breached, ECE generates a notification for all users in the group.

Policy-Driven Charging Example

The following is an example of how ECE supports policy-driven charging:

A service provider allows a customer to download 300 megabytes of data per month at high QoS.

The customer’s counter for data downloaded resets at the beginning of each month.

The service provider defines policy thresholds in a policy specification in PDC with the label names normal-QoS and low-QoS; these policy threshold labels are also stored in ECE.

The service provider configures the PCRF with a policy rule that defines what action to take based on the labels defined in the policy specification. The rule determines what action to take when the customer reaches 300 megabytes of data before the end of the month.

The PCRF rule uses the label names normal-QoS and low-QoS as follows:
The customer reaches the 300 megabyte data quota.
Using the rule shown above, the PCRF makes a policy decision to configure the Policy and Charging Enforcement Function (PCEF) so that the data transfer speed is set to 128 kilobits per second, downgraded from 10 megabits per second.
The PCEF enforces this decision by changing the data transfer speed on the network switch.

Configuring Policy-Driven Charging

ECE supports in-session notifications for policy-driven charging by publishing asynchronous external notifications during a policy session. Policy clients, such as Diameter Gateway, consume the data in these notifications for sending in-session notifications to the PCRF.

About ECE and Policy Clients

To support policy-driven charging, ECE offers a policy management API. Policy clients can use the API to retrieve data relevant for policy enforcement from its data grid.

Policy-driven charging in ECE is based on the Policy Charging and Rules Function (PCRF), defined in the 3GPP TS 23.203 v9.9.0 (2011-06) specification. The PCRF integrates with ECE through your online network mediation software.

ECE exposes its cached in-memory data so that your online network mediation software can retrieve policy counter information and policy-related subscriber preference information. ECE publishes notifications containing the policy information and your online network mediation software uses the notifications to send the information to the PCRF for evaluation.

Figure 14–1 illustrates how ECE fits into a charging system that implements policy-driven charging.

If (status_label=normal-QoS) (Bandwidth=10 Mbps)
If (status_label=low-QoS) (Bandwidth=128 kbps)
How ECE Processes Policy Requests for Online Network Mediation System

The following procedure describes how ECE processes requests for policy-driven charging from your online network mediation software (or from Diameter Gateway).

1. A customer starts to use a service which starts a network session.
   For example, the customer turns on a mobile phone which connects to a wireless network.

2. At the start of the network session, the Policy and Charging Enforcement Function (PCEF) obtains from the Policy and Charging and Rule Function (PCRF) a policy configuration.
   The PCEF uses the Gx interface to obtain the policy configuration for the network session.

3. The PCRF requests the policy counters and subscriber preferences from your online network mediation software (or from Diameter Gateway).
   The PCRF uses the Diameter Sy/Sp interface.

4. Your online network mediation software (or Diameter Gateway) initiates a policy session with ECE that does the following:
Requests policy counter and status label information.
Requests the policy counters for a specific charge offer and subscribes for receiving notifications when the values of the policy counter information changes.

Requests policy-related subscriber preferences information by doing one of the following:
- Retrieves the value for a specified set of subscriber preferences and subscribes for receiving notifications when the values of the preferences change during the policy session.
- Retrieves only the values for a specified set of subscriber preferences and does not subscribe for receiving notifications when the values of the preferences change during the policy session.

Your online network mediation software (or Diameter Gateway) uses the PolicySessionRequest ECE Java combined Sy/Sp (implemented as Sh) interface which uses the SubscribeNotificationRequest procedure and the UserDataRequest procedure.

5. ECE sends a policy response to your online network mediation software (or to Diameter Gateway), which does the following:
- Indicates whether the request succeeded or failed and provides a list of reasons supporting the response.
- Sends the status of the policy counters as follows for the specified service. If the service is not specified, returns the information for all services:
  - Sends the policy specification (offer profile) name configured for the service.
  - Sends the status label associated with the policy counter.
  - Sends an effective time for the values of the policy counters. After the effective time expires, the PCRF is expected to send another request for policy counter and status label information (send another SpendingLimitReportRequest).
  - Sends the label name of the next probable status that applies after the effective time expires. For example, Medium_QoS.
  - Sends the delay interval. The PCRF can use the delay interval and the effective time to determine when to query for the policy counters again.

ECE uses the SpendingLimitReportResponse procedure of the ECE Java Sy interface.

- Sends the subscriber preferences for the specified set of subscriber preferences.
  ECE uses the SubscribeNotificationResponse procedure of the ECE Java Sp interface.

6. The PCRF rules engine interprets the information and installs a policy on the PCEF which the PCEF enforces.

7. A charging session is established and the PCEF sends a Ro message to your online network mediation software (or to Diameter Gateway).

8. Your online network mediation software (or Diameter Gateway) initiates a charging session with ECE.

9. ECE publishes policy notifications for the following:
Changes to the policy counter status for the policy counters the PCRF subscribed for (Sy data) at the beginning of the policy session.

Changes to the subscriber preferences the PCRF subscribed for (Sp data), if any, at the beginning of the policy session.

For information about policy notifications, see "ECE Notification Reference".

10. Your online network mediation software (or Diameter Gateway) consumes the policy notifications and sends the data to the PCRF.

11. As the charging session continues, ECE performs credit control functions: rates events, authorizes usage events only if adequate balance is available, administers threshold checks based on the current balance and consumed reservation of the customer balance.

12. When ECE detects a policy threshold breach during the charging session, it publishes a policy notification to the JMS notification queue containing the new status of the policy counter. Your online network mediation software (or Diameter Gateway) sends the data to the PCRF.

The customer balance change that causes the policy threshold breach could occur as a result of any of the following:

- Usage requests coming from the network mediation system
- Update requests coming from BRM (a subscription activity in the customer management system)
- Top-ups coming from top-up systems

13. The PCRF evaluates the new policy counter values and the associated policy status labels and installs a new policy configuration on the PCEF.

The new policy is established dynamically during the charging session.

14. The customer stops using his service which ends the network session.

15. Your online network mediation software (or Diameter Gateway) terminates the charging session with ECE.

16. Your online network mediation software (or Diameter Gateway) terminates the policy session with ECE.

### Configuring Breach Tolerance for Policy-Tier Thresholds

In case of policy-driven charging, policy-tier thresholds must be crossed to trigger implementation of business rules, such as reduced QoS for subscribers who download an excessive amount of data.

For policy tier thresholds, BRM cannot authorize an amount above the threshold, even if the subscriber's credit balances are sufficient to cover the charges. Instead, BRM authorizes the remaining balance up to the policy threshold but does not send an FUI. Therefore, only about 80 percent of the remaining balance is made available for use. The session ends when the remaining balance becomes so small that the service can no longer be supported.

To enable subscribers to continue using a service as they near a policy tier threshold, you must configure a breach tolerance for the threshold. When the threshold is crossed, the service continues under a new business rule, such as lower QoS for larger download totals.
For example, suppose the network sends a usage request for 200 MB, but adding that to a subscriber’s current 1.9 GB policy counter balance will cause the balance to breach a 2 GB policy tier threshold. In this case, BRM does one of the following:

- **Without Breach Tolerance:** If a breach tolerance is not configured, BRM makes only about 80 MB available to prevent the usage from exceeding the policy tier threshold. When usage reduces the 80 MB balance to the point that the remaining balance cannot support the service, the session ends.

- **With Breach Tolerance:** If a breach tolerance of 100 or more MB is configured, BRM authorizes the entire 200 MB request. This enables the subscriber’s usage to cross the 2 GB policy tier threshold by 100 MB. As soon as the policy tier threshold is crossed, a change in the quality of service is triggered, and the service continues under the new policy.

You can set a breach tolerance for each balance element used in a policy counter. You decide what tolerance value is appropriate for your business needs.

To configure a tolerance for policy-tier threshold breaches:

1. Before charging servers are started, open `ECE_home/oceceserver/config/management/charging-settings.xml` and uncomment the following lines:

   ```xml
   <toleranceConfigMappingGroup  config-class="java.util.ArrayList">
   <toleranceConfig
       config-class="oracle.communication.brm.charging.appconfiguration.
       beans.policy.ToleranceConfig"
       balanceElementId="12345"  tolerance='1.25'/>
   <toleranceConfig
       config-class="oracle.communication.brm.charging.appconfiguration.
       beans.policy.ToleranceConfig"
       balanceElementId="34567"  tolerance='3'/>
   </toleranceConfigMappingGroup>
   ```

2. Save the file.
3. On the driver machine, change directory to the `ECE_home/oceceserver/bin` directory.
4. Start Elastic Charging Controller (ECC):
   ```
   ./ecc
   ```
5. Start your charging servers:
   ```
   start server
   ```
6. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Ensure that the ECE charging servers are started.
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceceserver/config/eeCTopology.conf` file.
      
      The `eeCTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the ECE Configuration node.
7. Expand charging.policyConfig.
10. For each balance element (policy counter) to which policy specifications apply in your system, do the following:
   a. Specify values for the following parameters:
      - beid: Enter the balance element ID of the balance element.
      - tolerance: Enter the RUM units allowed to exceed the authorized usage quantity that ECE returns to the network for a specified charging session. The value must be greater than 0. Base it on your business needs.
      - Your customers can use all balances and exceed their policy-tier threshold limits by the specified number of RUM units.
   b. Click the setPolicyTolerance button.

About Integrating Policy Clients with ECE

Policy clients such as Diameter Gateway integrate with ECE by using the ECE policy APIs.

The policy client uses the ECE policy Sy interface to retrieve policy counter information from ECE. The policy client, in turn, sends the policy counter information to the PCRF using its Diameter Sy interface. As part of initiating a policy Sy session with ECE, the policy client subscribes for receiving notifications that contain the policy counter information.

The policy client uses the ECE policy Sp interface to retrieve customer preferences information from ECE. The policy client, in turn, sends the customer preferences information to the PCRF using its Diameter Sp interface. As part of initiating a policy Sp session with ECE, the policy client subscribes for receiving notifications that contain the customer preferences information.

About the ECE Sy and Sp Interface

To support policy-driven charging, ECE offers policy management APIs. The ECE Sy interface enables policy clients to subscribe for and retrieve spending limit information about policy counters from ECE. The ECE Sp interface enables policy clients to subscribe for and retrieve customer preference information relevant to policy enforcement from ECE.

The following sections describe each interface:

- About the ECE Sy Interface
- About the ECE Sp Interface

ECE also supports a combined ECE Sy and Sp interface that enables policy clients to retrieve and subscribe for both types of information in one policy session. A combined ECE Sy and Sp interface reduces the number of messages between ECE and policy clients. See "About a Combined ECE Sy and Sp Interface" for information.
About the ECE Sy Interface

ECE supports the Sy interface which is used by the PCRF to retrieve policy counter information. To support the Sy interface, ECE offers the following ECE Sy procedures and notification:

- **Spending Limit Report Request**
  
  Policy clients such as Diameter Gateway use this procedure to request the status of policy counters available in ECE and to subscribe and unsubscribe (for the PCRF) to updates of ECE policy counters.

- **Spending Limit Notification**
  
  ECE uses this notification to report statuses of requested policy counters for one or more services and also report the results of request processing.
  
  The policy client transfers the status information to the PCRF.

About the ECE Sp Interface

ECE supports the Sp interface which is used by the PCRF to query customer preferences. To support the Sp interface, ECE offers the following ECE Sp procedures:

- **Subscribe Notification Request**
  
  Policy clients such as Diameter Gateway use this procedure to retrieve customer preferences and to subscribe and unsubscribe (for the PCRF) to updates of customer preference data changes.
  
  The customer preferences can include the following:
  - Customer’s allowed services
  - Customer’s allowed Quality of Service (QoS)
  - Customer’s preferred channel for receiving notifications (such as receiving an SMS or email)
  - Customer’s preferred language

- **Subscribe Notification Response**
  
  ECE uses this procedure to report customer-preference data updates to the policy client subscribed for the notification.

- **User Data Request**
  
  Policy clients use the User Data Request procedure only to retrieve subscriber preferences without subscribing for receiving notifications when the preferences change.

- **User Data Response**
  
  ECE uses this procedure to send subscriber-preference data to the policy client.
  
  The policy client transfers customer preference data to the PCRF.

**Querying for Extended Subscriber Preference Information in Sp Query**

The PCRF can also query extended information about customers and services. The policy client, such as Diameter Gateway, uses the ECE policy Sp query procedure to retrieve extended customer and service information.

To retrieve extended information from ECE using the policy Sp query request, you must configure the extended service and customer information in ECE.
To configure the query for extended service and customer information:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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/occeserver/config/eeetopology.conf file.
   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor's MBean hierarchy, find the ECE configuration MBeans.
2. Expand the ECE Configuration node.
3. Expand charging.policyConfig.
4. Expand Operations.
5. Select setDsl.
6. Do the following for each type of service or customer you want the policy client to query:
   a. For the alias parameter, replace String with the alias for the extended information to use in the policy query request.
      Configured aliases are included in the policy query request.
   b. For the dsl parameter, replace String with the DSL to use to retrieve the information from ECE in the following format:
      gettype([product | customer]/attribute with arguments)
      For example:
      getObject(product/lifeCycleStateName)
     c. Click the setDsl button.
       This creates a mapping between the extended information alias with the DSL used to retrieve the extended information from customers and services.

About a Combined ECE Sy and Sp Interface

ECE supports combining its ECE Sp and Sy interfaces by offering the following procedures:

- Policy Session Request
  Policy clients, such as Diameter Gateway, use this procedure for retrieving Sp and Sy information and subscribing or unsubscribing (for the PCRF) to receiving updates to Sp and Sy data. This request is a combination of the Spending Limit Report Request and the Subscribe Notification Request.

- Policy Session Response
  ECE uses this procedure to report the information requested by the Policy Session Request and provide results of request processing.
  The policy client transfers the information to the PCRF.
About Calculating Maximum Authorization for Policy-Driven Charging Sessions

For policy-driven charging sessions, ECE readjusts the requested quota based on the following data:

- Current balance
- Used reservation across all parallel sessions
- Nearest threshold in the policy specification

For example, consider this situation:

- Current balance: 80 MB
- Used reservation across all parallel sessions (iPhone, video, computer): 35 MB
- Nearest threshold in the policy specification: 140 MB

Under those conditions, if ECE receives an authorization request for an additional 30 MB, that request exceeds the 140 MB threshold by 5 MB (80 MB + 35 MB + 30 MB = 145 MB). Therefore, unless a breach tolerance of 5 MB or more is configured, ECE authorizes only 25 MG.

Configuring ECE to Reject Spending Limit Requests Without Counters

For Sy subscriptions, you can configure ECE to reject a Spending Limit Request (SLR) if there are no policy counters available for the subscriber.

To configure ECE to reject SLRs when no policy counters are available:

1. Access the ECE MBeans:
   a. Log on to the driver machine, which is the machine on which you installed ECE.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/occeserver/config/eceTopology.conf` file.
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor's MBean hierarchy, expand the ECE Configuration node.

2. Expand `charging.policyConfig`.
3. Expand `Attributes`.
4. Set the `syRejectNoCounters` attribute to `true`.

About the Policy Management API

To use the policy management API, clients call the submitPolicy API with PolicyRequest.

For details about the policy management API, see the documentation for `oracle.communication.brm.charging.brs`, `oracle.communication.brm.charging.messages.policy`, and
oracle.communication.brm.charging.messages.query (for user data request/response information) in BRM ECE Java API Reference.
This document describes testing scenarios for testing an Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) implementation.

Topics in this document:

- About ECE Testing Utilities
- About Loading Sample Data
- About Performance MBean
- Changing Time and Date to Test ECE
- Using the query Utility to Test ECE
- Verifying that Usage Requests Can Be Processed
- Verifying That ECE Notifications Are Published to the JMS Topic
- Verifying that Friends and Family Calls Are Processed
- Verifying That Closed User Group Calls Are Processed
- Verifying That Balance Impacts Are Assigned to Bill Items
- Verifying That payloads Are Correctly Formed

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

About ECE Testing Utilities

ECE offers the following testing utilities you can use when implementing ECE in a charging system:

- Simulator. The simulator emulates the role of a client application, such as a network mediation software client application, sending requests to ECE. The simulator enables you to send usage requests, query requests, update requests, or policy requests to ECE for processing. You can run sample workloads for testing latency and throughput of your system.

- Loader. The loader utility loads sample pricing-related configuration data, sample pricing data, and sample customer data.
About Loading Sample Data

- **query** utility. The **query** utility enables you to execute queries on ECE data for development or debugging purposes.

  **Important:** The ECE data model within the Coherence cache is subject to change. Oracle does not recommend that client applications directly use the Coherence API or the **query** utility for accessing ECE cache data. For querying ECE cache data, write your client applications to use the ECE APIs such as the balance query and authentication query APIs.

- **customerGenerator** utility. The **customerGenerator** utility creates XML files that represent sample customer data. After creating the files, use **customerLoader** utility to load data into ECE.

  **Caution:** Do not run the **customerLoader** utility without the **-incremental** parameter in a production environment.

An ECE customer XML data file must conform to the format of the ECE customer XML schema file (*ECE_home/oceceserver/odi_transformation/ECE_Schema.xsd*).

- **customerLoader** utility. The **customerLoader** utility loads customer data into from XML files. In addition to using this utility in a development system, you can use it in a production system to correct data migration errors. To do so, run the utility with the **-incremental** parameter.

  **Caution:** Do not run the **customerLoader** utility without the **-incremental** parameter in a production environment.

- **Performance MBean.** You can use the **PerformanceMonitor MBean** to monitor the performance of your ECE deployment during testing. See “About Performance MBean”.

### About Loading Sample Data

After installing ECE, you can load sample data. Sample data is in the *ECE_home/oceceserver/sample_data* directory which includes:

- Sample data for integrating with BRM
- Sample data for integrating with PDC
- Sample data for integrating with clients that send policy requests (used for policy testing)

Sample data includes sample event definitions, sample configuration data, sample product offering cross-reference data, and sample customer data. Subsets of sample data geared for ECE implementations for policy-related charging is also available.

To use sample data, you configure your data-loading utilities to load data from sample data directories.
Loading Pricing Data On a Development System

When you install ECE, ECE includes no pricing data. You use the `pricingLoader` utility to load sample pricing data in an XML file to test charging.

---

**Caution:** Do not use the `pricingLoader` utility on a production system. Use `pricingLoader` only on a development system when PDC is not used.

---

You can use the following as sample data:

- The sample data installed with ECE.
- Data exported from PDC by using the PDC `ImportExportPricing` utility.

By default, the `pricingLoader` utility reads data stored in `ECE_home/oceceserver/config/management/sample_data/pricing_data`. To change the directory where the sample data is stored:

1. Open the `ECE_home/oceceserver/config/management/migration-configuration.xml` file.
2. Set the `pricingDataDirectory` parameter to the path of the directory that contains the ECE pricing component XML data file.

To load sample pricing data, run the following command:

```bash
> start pricingLoader
```

---

**Note:** ECE must be running when you load pricing data.

---

About Performance MBean

You can use the `PerformanceMonitor` MBean to monitor the performance of your ECE deployment. You can monitor the CPU usage of server nodes and client nodes, such as the simulator, during your testing.

For example, when building charging extensions, you can run ECE without your extensions and use the methods to see how much CPU time is used. You can then run ECE with your extensions, and use the methods again to see how much CPU time is used. By comparing the CPU times, you can derive the additional time spent by your extension.

The following `PerformanceMonitor` MBean methods are available:

- `startTrackingCPU()`
  
  The `startTrackingCPU()` method starts tracking CPU usage for the running process.

- `stopTrackingCPU()`
  
  Use the `stopTrackingCPU()` method to stop tracking CPU usage for the running process. This method returns CPU utilization between 0 and 1 where 0 means 0% CPU usage and 1 means 100% CPU usage.

- `getTrackedCPU()`
Use the getTrackedCPU() method to get the last tracked CPU usage between \([0, 1]\) if a previously tracked CPU usage is available. If a previously tracked CPU usage is not available, \(-1\) is returned.

The simulator MBean exposes the throughput information through the getlastThroughput() method. The getlastThroughput() method gets the throughput number from the last successfully completed simulation run. If completed simulation runs do not exist or if a simulation run is in-progress, \(-1\) is returned.

**Changing Time and Date to Test ECE**

You can change ECE’s current time and date, without affecting the operating system time and date, to test time-sensitive functions associated with Rated Event Formatter and Diameter Gateway in ECE.

---

**Caution:** Changing the time and date introduces the possibility of corrupting data. Do not change the time and date in a production database.

---

For example, you can change ECE’s current time and date to test the following:

- Whether accounts are billed correctly. If you advance the date in BRM to the next billing cycle to test if the accounts are billed correctly, you must advance the date in ECE to the same date as set in BRM. This ensures that the events rated by ECE on that day are sent to BRM with the same date as set in BRM so that the events can be billed for the next billing cycle.

- Whether customer’s spending limit is reported correctly. If a charge offer includes a conditional balance impact and the conditional balance impact is valid only for a day, you can advance the date by a day to ensure that when the Spending-Limit-Report-Request (SLR/SLA) request is received, the spending limit for the next day is reported.

- Whether events are rerated correctly. You can advance the date in ECE to store rated events in the Oracle NoSQL database data store with the future date to ensure that they are rerated when rerating is run in BRM for the future date.

To change the time and date to test ECE:

1. Access the ECE MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. Start a JMX editor, such as JConsole, that enables you to edit MBean attributes.
   d. Connect to the ECE charging server node set to `start CohMgt = true` in the `ECE_home/oceceserver/config/eceTopology.conf` file.
      
      The `eceTopology.conf` file also contains the host name and port number for the node.
   e. In the editor’s MBean hierarchy, expand the **ECE configuration** node.

2. Expand `charging.server`.

3. Expand `Attributes`.

4. Specify the values for the following attributes:
   - `virtualTimeMode`. Enter one of the following values:
Using the query Utility to Test ECE

The query utility provides access to ECE cache content, enabling you to execute queries on ECE Coherence caches. The query utility is meant to be used for debugging purposes only.

You can use the query utility to write scripts that interact with the ECE domain objects, creating CohQL queries. The query utility supports all ECE caches and objects.

**Important:** Oracle does not recommend that client applications directly use the Coherence API or the query utility for accessing ECE cache data. For querying ECE cache data, write your client applications to use the ECE APIs such as the balance query and authentication query APIs.

The query utility is included with the ECE Server software in \texttt{ECE\_home/oceceserver/bin}.

To learn about query utility options, use the help command:

\$ ./query.sh -h

The following shows non-interactive use of the query utility:

\$ ./query.sh -s -c -l "select sum(getAvailableBalance('USD',null).getQuantity()) from Balance"

The following shows interactive use of the query utility:

\$ ./query.sh
Coherence Command Line Tool
CohQL> select count () from Customer;
Results
1000
CohQL> select key(), value().getCode().toString() from BalanceElement
Results
840, 'USD'

The query utility log file is \texttt{ECE\_home/oceceserver/logs/query\_out.log}.
The query statement history is contained in `ECE_home/oecceserver/bin/.cohql-history`. You can use the up and down arrows to move through the command history.

**Important:** Oracle does not recommend that client applications directly use the Coherence API or the `query` utility for accessing ECE cache data. For querying ECE cache data, write your client applications to use the ECE APIs such as the balance query and authentication query APIs.

---

**Example: Query the Subscriber Base Balance Summary**

Here is an example of how to summarize balance amounts across the entire subscriber base (total balance) in the grid:

```
$ ./query.sh -s -c -l "select sum(getAvailableBalance('USD',null).getQuantity())
from Balance"
```

---

**Example: Query a Customer Balance**

Following is an example of how to query a customer's balance. You first query a specific customer to find the balance ID, and then you query a specific balance to find the balance element and balance amount.

**Tip:** You can use the same model for querying a customer's active session object.

---

**Step 1: Query the customer to find the balance ID**

To query the customer and find the balance ID:

```
$ ./query.sh
Coherence Command Line Tool
CohQl> select key(), value() from Customer where key() = "Cust#6500000001"
Results
"Cust#6500000001",
####################################### Customer Begin
####
########################################
CustomerImpl
{ customerId='Cust#6500000001
, inTransaction='null
, defaultBalanceId='Bal#6500000001
, externalReference='1
, version=0
, profiles={Birthday=[RatingProfileValueImpl{name=NUMBER, value=2013-08-21,
validFrom=1970-01-01T00:00:00.000Z, validTo='292278994-08-17T07:07:55.807Z}]}
, subscriberPreferences={}
, subscribedPreferences=null
, AlterationSharingAgreements ={()}
, DistributionSharingAgreements ={()}
, productMap={Pro#6500000001=ProductImpl{
...}
, balanceId = 'Bal#6500000001'
, profiles = {FriendsAndFamily=[RatingProfileValueImpl{name=NUMBER,
value=6501234567, validFrom=1970-01-01T00:00:00.000Z,
validTo='292278994-08-17T07:12:55.807Z}]
, subscriberPreferences = ()
, subscribedSpendingLimitCounters = ()
, Life cycle state = 102
```
Verifying that Usage Requests Can Be Processed

You use the ECE simulator to send requests to ECE for processing. The simulator emulates network traffic coming from a network mediation system. You use the ECE query utility to verify that the usage has impacted the customer balance.

Step 2: Query the balance to find the balance element

To query the balance to find the balance element, you specify two components of the associated key (composite key) that links the customer to the balance.

$ ./query.sh

Coherence Command Line Tool

CohQl> select value() from Balance where key().getId() = "Bal#6500000001" and key().getAssociatedKey() = "Cust#6500000001"

Results

[BalanceImpl{BalanceId=Bal#6500000001}
 {externalRevision=0}{OwnerId=1}
 {BillingUnitId=BillingUnit#6500000001}
 {BillingUnit=null}ActiveReservationMap{}}
 balanceItemSpecs{{USD=BalanceItemSpecImpl{beCode='USD', unit=Money{cur=USD}
 , creditProfile=oracle.communication.brm.charging.config.creditprofile.internal.CreditProfileReference@1dc79d4
 , consumptionRule=EARLIEST_START}}}
 balanceItems{{(BalanceItemImpl{balanceItemId=0}{currentBalance=-10000}
 {balanceItemSpec=BalanceItemSpecImpl{beCode='USD', unit=Money{cur=USD}
 , creditProfile=oracle.communication.brm.charging.config.creditprofile.internal.CreditProfileReference@1dc79d4
 , consumptionRule=EARLIEST_START}{validity=null}{validityRule=null})}},
Note: If you installed online or offline network mediation software, you can use that software instead of the ECE simulator to send usage requests for online or offline charging. This section describes how to use the simulator only.

The simulator enables you to control the types of usage requests sent and the number and type of subscribers sending the usage requests.

To verify that usage requests can be processed, perform the steps described in these sections:

- Starting ECE Nodes in the Cluster
- Running the Simulator to Send Usage Requests
- Verifying that Balances Are Impacted in ECE

Starting ECE Nodes in the Cluster

To start all ECE nodes in the cluster:

1. Log on to the driver machine.
2. Change directory to the `bin` directory:

   ```
   cd ECE_home/oceceserver/bin
   ```

3. Start the Elastic Charging Controller:

   ```
   ./ecc
   ```

4. Start the ECE nodes:

   ```
   start
   ```

To verify that the ECE nodes are running:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
   b. Start the ECE charging servers (if they are not started).
   c. 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.

   d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
   e. In the editor’s MBean hierarchy, find the ECE state machine MBeans.

2. Expand the ECE Configuration node.
3. Expand ECE State Machine.
4. Expand StateManager.
5. Expand Attributes.
6. Verify that the `stateName` attribute is set to `UsageProcessing`.

   This means the ECE nodes are running.
Running the Simulator to Send Usage Requests

To run the simulator and send usage requests:

1. Start the ECE simulator:
   ```
   start simulator
   ```

2. Initialize the simulator:
   ```
   init simulator
   ```

3. Run the sample workload:
   ```
   simulate simulator
   ```
   The simulator takes a few seconds to finish processing the workload.

4. Open the `invocation.log` file located in `ECE_home/oceseserver`. You should see statistics for the sample workload.

Verifying that Balances Are Impacted in ECE

To verify that the usage requests impacted customer’s balances, use the ECE `query` utility.

**Query for customer balances in the Customer cache**

Here are two examples of how to query the customer cache to return the customer balances:

```
$ ./query.sh
Coherence Command Line Tool
CohQl> "select value() from Balance where ownerId='cccc'"
```

```
$ ./query.sh
Coherence Command Line Tool
CohQl> "select value() from Balance where balanceId='xxxx'"
```

In the results of the query returned, locate the following string:
```
{currentBalance=UnitValue{quantity=amount, unit=Money{cur=USD}}
```

where `amount` is the quantity amount of the balance impact.

Verifying That ECE Notifications Are Published to the JMS Topic

To verify that ECE external asynchronous notifications are being published to the JMS topic, you can use the following sample SDK notification programs:

- `sample_jms_client.sh`
- `sample_jms_server.sh`

Use these sample programs to check the correctness of the JMS topic.

You can also use the `sample_jms_client.sh` sample program to check the messages produced from the ECE side to the JMS topic.

Disabling the Publishing of ECE Notifications to the JMS Topic

Some types of testing may not require publishing ECE external notifications to the JMS topic.
To disable external notifications:

1. Open the $ECE\_home/oceceserver/config/charging-cache-config.xml$ file.

2. For the ServiceContext module, change the cache-store configuration entry by replacing the following:

   $$\begin{align*}
   &<init-param> \\
   &<param-name>cache-store</param-name> \\
   &<param-value>oracle.communication.brm.charging.notification.internal.coherence.AsynchronousNotificationPublisher</param-value> \\
   \end{align*}$$

   with this:

   $$\begin{align*}
   &<init-param> \\
   &<param-name>cache-store</param-name> \\
   &<param-value>oracle.communication.brm.charging.util.coherence.internal.NoPersistenceCacheStore</param-value> \\
   \end{align*}$$

   ECE external notifications are disabled.

3. Save the file.

Verifying that Friends and Family Calls Are Processed

To verify that your ECE deployment is processing friends and family calls, perform prerequisite tasks in BRM and PDC and then generate usage for the friends and family call for the customer.

To verify that friends and family calls are processed:

1. Ensure the appropriate provisioning tag is available in BRM as follows:
   a. Ensure you define a provisioning tag that includes the **Friends&Family** extended rating attribute (ERA).
   b. Ensure the provisioning tag in BRM contains the same profile specification labels provided in PDC.

   The profile specification labels that come ready-to-use in the PDC installation are **MYFRIENDS** and **MYFAMILY**. Specify these labels in the provisioning tag when using the ready-to-use profile specification labels in PDC.
   c. (Optional) If you create a new provisioning tag in BRM, rather than using the ready-to-use sample provisioning tag, run the **SyncPDC** utility to synchronize the provisioning tag name to PDC.

2. If not already loaded, load the sample profile attribute specification for friends and family into PDC.

   The sample XML file is available at $PDC\_home/apps/Samples/Examples/PDCSampleProfileSpec.xml$ where $PDC\_home$ is the directory in which you installed PDC.

   Use the PDC **ImportExportPricing** utility to load the XML file into the PDC database.

3. If not already loaded, load the sample custom analyzer rule for friends and family into PDC.
Verifying That Closed User Group Calls Are Processed

1. Ensure the appropriate provisioning tag is available in BRM by doing the following:
   a. Ensure you define a provisioning tag that includes the ClosedUserGroup extended rating attribute (ERA).
   b. Ensure the provisioning tag in BRM contains the same profile specification labels that are provided in PDC.
Verifying That Closed User Group Calls Are Processed

The profile specification label that comes in the PDC installation is CLOSEDUSERGROUP. Specify this label in the provisioning tag when using the profile specification labels in PDC.

c. (Optional) If you create a new provisioning tag in BRM, rather than using the sample provisioning tag, run the SyncPDC utility to synchronize the provisioning tag name to PDC.

2. If not already loaded, load the sample profile attribute specification for closed user group into PDC.

The sample XML file is available at the following:

- For service-based closed user group samples:
  \[PDC\_home/apps/Samples/Examples/Sample\_ServiceCUG\_ProfileSpecification.xml\]
- For customer-based closed user groups that work with sample data:
  \[PDC\_home/apps/Samples/Examples/OOB\_ProfileSpecifications.xml\]

Use the PDC ImportExportPricing utility to load the XML file into the PDC database.

**Tip:** Closed user group profiles are rating profiles (known as extended rating attributes in BRM) that have a closed-user-group affiliation. The closed-user-group affiliation is enabled by setting the useDynamicIdentifier field to true in the PDC profile attribute specification.

3. If not already loaded, load the sample custom analyzer rule for closed user group into PDC.

The sample XML file is available at the following:

- For service-based closed user group samples:
  \[PDC\_home/apps/Samples/Examples/Sample\_ServiceCUG\_CR.xml\]
- For customer-based closed user groups that work with sample data:
  \[PDC\_home/apps/Samples/Examples/OOB\_CRs.xml\]

OOB\_CRs.xml contains three custom rules: Friends&Family and ClosedUserGroup and SpecialDay. These custom rules are designed to be used specifically with generic selectors.

Use the PDC ImportExportPricing utility to load the XML file into the PDC database.

4. In PDC, configure the charge offer for closed user group calls by doing the following:

a. Create a generic selector with the ClosedUserGroup custom analyzer rule.

b. Create the charge offer for the closed user group calling service.

c. For the charge offer, select the provisioning tag that specifies ClosedUserGroup.

d. Create the charge for the charge offer.

e. For the charge, include the generic selector with the ClosedUserGroup rule.
Tip: You associate the closed user group rule in the generic selector with a result: a string value that maps to the rule, such as ClosedUserGroup. At runtime, ECE uses this result in the charge to apply different rates for calls to the closed user group.

5. Verify that Pricing Updater is started.

6. Publish the PDC pricing data to the ECE rating engine.

Pricing Updater synchronizes the pricing data to ECE.

7. Verify that EM Gateway is started.

8. In BRM, for both the calling customer and the called customer, create the customer account, purchase the charge offer, and configure the required closed-user-group-profile information for the customer.

For example, if the closed user group profile is at the customer level, specify the closed user group phone number. If the closed user group profile is at the service level, specify the closed user group name.

The BRM customer data updates are incorporated into the ECE cache in real time through EM Gateway.

9. Generate usage for a closed user group call for the calling customer.

Use the ECE SDK sample programs to generate usage.

10. Verify that balances are impacted as expected.

Once you verify that closed user group calls are processed as expected using the closed user group sample data in PDC and BRM, create your own closed user group configurations.

Verifying That Balance Impacts Are Assigned to Bill Items

To verify that balance impacts are assigned to bill items according to your business rules:

Important: Before loading item type selectors into PDC, make a backup copy of the customized config_item_tags.xml and config_item_types.xml files in BRM.

1. Ensure that a storable class for each bill item type is available in BRM.

If you are verifying that ECE can apply balance impacts to a custom bill item, ensure the custom storable class is available in BRM. For example, /item/custom.

2. If not already loaded, load the item type selector into PDC.

The item type selector contains item specifications and item type selector rules.

You associate item type selector rules with an item tag: a string value that maps to the item type. At runtime, ECE evaluates your item-type-selector rule. The result of the rule evaluation is a unique item type. ECE assigns balance impacts to the bill item associated to the item type.

Item-type-selector XML files are available at PDC_home/apps/Samples/Examples.

Use the PDC ImportExportPricing utility to load the item-type-selector XML file into the PDC database.
3. Verify that Pricing Updater is started.

4. Verify that EM Gateway is started.

5. In BRM, create the customer account and purchase the charge offer for the service associated with the bill items for which you are verifying bill-item assignment.
   The BRM customer data updates are incorporated into the ECE cache in real time through EM Gateway.

6. Generate usage for the customer that impacts the bill items for which you are verifying bill-item assignment.
   Use the ECE SDK sample programs to generate usage.

7. Run billing.

8. Verify that balance impacts are assigned to bill items as expected.

**Verifying That Payloads Are Correctly Formed**

To debug rating errors, you may need to verify that payloads in usage requests are correctly formed. You can view payloads in the RequestSpecification cache by using the following CohQL command and piping the contents to a file:

```sql
select toSpecFormat() from RequestSpecification
```

The RequestSpecification cache contains read-only information.

If you identify an issue with a payload, correct the issue in PDC as follows:

1. Export the event object.

2. Update the XML with the corrections.

3. Re-import the event object.

   After the event object is re-imported, PDC re-publishes the event object, and Pricing Updater updates the event definition in ECE in the RequestSpecification cache accordingly.
This document describes how to customize the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) rating.

Use the ECE extensions to customize Diameter Gateway, RADIUS Gateway, pre-rating, post-rating, post-charging, and post-update processes. ECE extensions include sample implementations that guide you in implementing your custom business logic.

Topics in this document:
- Operational Considerations
- Extension Points
- Implementing the Extensions Logic
- Sample Extensions

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

**Operational Considerations**

All pre-rating, post-rating, post-charging, and post-update extensions must be implemented in a single class respectively. This class can delegate to additional implementations if multiple extensions are being implemented.

Extensions data is loaded into a replicate cache in Coherence and the amount of data loaded into the cache must be taken into consideration when sizing for Java.

**Configuration**

You configure implementation classes for the diameter-request processing and usage-request processing extension points through JMX management by using a JMX editor.

To configure the implementation classes for the diameter-request processing and usage-request processing extension points:

1. Access the ECE configuration MBeans:
   a. Log on to the driver machine.
b. Start the ECE charging servers (if they are not started).

---

**Note:** Ensure that the extension code is provided in Classpath, typically under the $ECE_HOME/lib directory, when the ECE Server running the JMX Management console is started. See "Sample Extensions".

---

c. 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/occeserver/config/eceTopology.conf` file.

d. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
e. In the editor's MBean hierarchy, find the ECE configuration MBeans.

2. Expand the ECE Configuration node.

3. Expand `charging.extensions`.

4. Expand Attributes.

5. Specify values for the following attributes as needed:

   - `diameterGyExtension`
   - `postRatingECEExtension`
   - `preRatingECEExtension`
   - `postChargingECEExtension`
   - `radiusAuthExtension`
   - `radiusAccountingExtension`
   - `postUpdateExtension`

---

**Performance**

If extensions are activated, they are called for during every usage request. Always consider performance for the code you execute in the extensions.

The extensions framework provides an extensions cache mechanism that provides the lowest latency access to the extensions data. It is recommended that you use the extensions cache mechanism rather than external data sources.

You can use the PerformanceMonitor MBean to monitor CPU usage of server nodes and client nodes. When building your charging extensions, the methods of the PerformanceMonitor MBean enable you to monitor the performance impacts of your extensions. For example, you can run ECE without your extensions and use the methods to see how much CPU time is used. You can then run ECE with your extensions, and use the methods again to see how much CPU time is used. By comparing the CPU times, you can derive the additional time spent by your extension.

---

**Logging**

Logging is available in the extensions using log4j logger to server node log file, for example:

```java
extensionContext.getLogger().debug('Hello World!' + extensionContext);
```
Exceptions

If ECE needs to reject the usage request, an **ExtensionsException** error can be thrown to reject the usage request and report a “CUSTOM_EXTENSION_ERROR” reason code in the response.

For details on **ExtensionsException**, see BRM ECE Java API Reference.

Security

To ensure security for the extension, follow these best practices:

- Enable JMX security
- Enable ECE cluster node security
- Ensure strict governance of OS accounts
- Follow secure Java coding practices
- Implement string code review process
- Run latency-sensitive performance tests on the extensions hooks
- Use Jar signing

Extension Points

The following sections describe the extension points to customize Diameter Gateway, RADIUS Gateway, pre-rating, rating, post-rating, post-charging, and post-update processes:

- Diameter-Request Processing Extension Points
- RADIUS-Request Processing Extension Points
- Usage-Request Processing Extension Points
- Update-Request Processing Extension Points

Diameter-Request Processing Extension Points

Diameter Gateway provides extension points for Credit Control Request (CCR) and Credit Control Answer (CCA) flows:

- **RequestReceived extension.** The role of the RequestReceived extension is to manipulate the CCR attribute-value pair (AVP) before the usage request is processed by Diameter Gateway and to provide an immediate response that bypasses the online charging system (OCS) completely.

- **PreOCS extension.** The role of the PreOCS extension is to manipulate the mapped ECE usage request payload to perform enrichments that are not possible in the RequestReceived extension.

- **PostOCS extension.** The role of the PostOCS extension is to manipulate the CCA AVPs before the diameter response is returned to the diameter client.

*Figure 16–1* shows the diameter-request processing extension points.
RADIUS-Request Processing Extension Points

RADIUS Gateway provides extension points for authentication and accounting flows.

**Authentication Extension Points**

RADIUS Gateway provides extension points for the authentication flow:

- **RequestReceived extension.** The role of the RequestReceived extension is to add or update a custom AVP before the authentication request is processed by RADIUS Gateway and to provide an immediate response that bypasses the OCS completely.

- **CustomEAPChallenge extension.** The role of the CustomEAPChallenge extension is to send custom access-challenge request to the RADIUS client when the Extensible Authentication Protocol (EAP) is used for authentication.

- **PreOCS extension.** The role of the PreOCS extension is to perform any actions related to authentication that are required before the RADIUS request is sent to ECE.

- **CustomAuth extension.** The role of the CustomAuth extension is to implement the custom EAP authentication methods.

- **CustomEncode extension.** The role of the CustomEncode extension is to implement the custom hashing algorithm that is used on passwords during authentication when the Password Authentication Protocol (PAP) is used for authentication.

- **PostOCS extension.** The role of the PostOCS extension is to add or update a custom AVP before the authentication response is returned to the RADIUS client.
Figure 16–2 shows the RADIUS-request processing extension points for EAP authentication.

Figure 16–2 Extension Points for EAP Authentication

Figure 16–3 shows the RADIUS-request processing extension points for PAP and Challenge-Handshake Authentication Protocol (CHAP) authentication. The solid line depicts PAP authentication and the dotted line depicts CHAP authentication in this figure.
Accounting Extension Points

RADIUS Gateway provides extension points for accounting flow:

- **RequestReceived extension.** The role of the RequestReceived extension is to add or update a custom AVP before the accounting request is processed by RADIUS Gateway and to provide an immediate response that bypasses the OCS completely.

- **PreOCS extension.** The role of the PreOCS extension is to enrich the usage request before the usage request is sent to ECE for accounting purposes.

- **PostOCS extension.** The role of the PostOCS extension is to add or update a custom AVP before the accounting response is returned to the RADIUS client.

*Figure 16–4* shows the RADIUS-request processing extension points for accounting.
Usage-Request Processing Extension Points

ECE provides extension points in the rating flow: before charge calculation, after charge calculation (prior to making a balance impact), and after charging (after applying a balance impact). The role of the pre-rating extension is to alter the usage request. The role of the post-rating extension is to alter the rated result. The role of the rating extension is to alter rated results after each of the following processes: rating, alteration, sharing, and taxation. The role of the post-charging extension is to enrich the usage response. You cannot customize rating during the rating, alteration, and tax calculation processes, only before and after. Access is provided to a custom data store that provides low-latency access to data required for the extensions; for example, customer data and balance data.

*Figure 16–5* shows the usage-request processing extension points.
Update-Request Processing Extension Points

ECE provides an extension point for post-update extensions in the updates-processing flow. The role of the post-update extension is to enrich and filter external notifications. This extension is called after receiving update requests and before publishing the external notifications.

Implementing the Extensions Logic

The GyExtension, PreRatingExtension, PostRatingExtension, PostChargingExtension, RadiusRequest, and RadiusResponse interfaces expose initialize() and shutdown() methods that are called by the hook framework when the server starts up and when it shuts down. Use these methods to configure your own internal data structures related to the extensions business logic.

For diameter-request processing extension points, a different method is called for each extension point. The handleRequestReceived() method is called for every CCR that is processed by the charging flow. The handlePreOCS() method is called for every CCR and usage request that is processed by the charging flow. The handlePostOCS() method is called for every CCA and usage response that is processed by the charging flow. All methods expose relevant ExtensionContext data for accessing the
ExtensionsDataRepository, AppConfigRepository, and other extensions-related contexts.

Figure 16–6 shows the data used in the diameter-request processing extension points.

Figure 16–6  Data Used in Diameter-Request Processing Extension Points

For extension points that process requests from RADIUS clients, the RadiusRequest and RadiusReply interfaces are exposed to the extension points through the ExtensionContext methods.

For authentication-related extension points, the following methods are called by the authentication flow:

- handleRequestReceived(). Called for every authentication request that is processed by the authentication flow.
- handlePreOCS(). Called to perform any actions related to authentication that are required in the authentication flow.
- handlePostOCS(). Called for each authentication response that is processed by the authentication flow.
- handleCustomEAPChallenge(). Called to send custom access-challenge requests to the RADIUS client in the EAP authentication flow.
- handleCustomAuth(). Called to implement a custom EAP authentication method in the authentication flow.
- handleCustomEncode(). Called to implement the custom hashing algorithm that is used on passwords in the PAP authentication flow.

Figure 16–7 shows the data used in the RADIUS-request processing extension points for authentication.
Implementing the Extensions Logic

For accounting-related extension points, the following methods are called by the accounting flow:

- **handleRequestReceived()**. Called for every accounting request that is processed by the accounting flow.
- **handlePreOCS()**. Called for every accounting request and usage request that is processed by the accounting flow.
- **handlePostOCS()**. Called for every accounting response and usage response that is processed by the accounting flow.

*Figure 16–8 shows the data used in the RADIUS-request processing extension points for accounting.*
For usage-request processing extension points, the `execute()` method is called for every usage request, rated result, usage response, and notification that is processed by the charging flow.

For the rating extension point, the following methods are called by the charging flow:

- **handlePostApplyCharge()**. Called to alter rated results after calculating charges (rating).
- **handlePostApplyAlteration()**. Called to alter rated results after calculating discounts (alteration).
- **handlePostApplyDistribution()**. Called to alter rated results after calculating charge distribution (sharing).
- **handlePostApplyTaxation()**. Called to alter rated results after calculating taxes (taxation).

All methods expose relevant ExtensionContext data for accessing the ExtensionsDataRepository, AppConfigRepository, and other extensions-related contexts.

*Figure 16–9* shows the data used in the pre-rating extension point.
### Figure 16–9 Data Used in Pre-Rating Extension Point

![Diagram](image)

Figure 16–9 shows the data used in the pre-rating extension point.

### Figure 16–10 Data Used in Rating Extension Point

![Diagram](image)

Figure 16–10 shows the data used in the rating extension point.
Figure 16–11 shows the data used in the post-rating extension point.

**Figure 16–11 Data Used in Post-Rating Extension Point**

![Diagram showing data used in post-rating extension point]

Figure 16–12 shows the data used in the post-charging extension point.

**Figure 16–12 Data Used in Post-Charging Extension Point**

![Diagram showing data used in post-charging extension point]

ECE provides build and deployment capabilities in the form of shell scripts. If any third-party libraries need to be used inside the custom extensions logic, copy the third-party JAR files to the `ECE_Home/lib` directory, where `ECE_Home` is the directory in which ECE is installed. After the JAR files have been copied, they need to synchronize across to the other servers in the cluster. Synchronization is done by running the `sync` command in Elastic Charging Controller (ECC).
Custom extensions logic implementation classes that implement the `GyExtension`, `PreRatingExtension`, `RatingExtension`, `PostRatingExtension`, `PostChargingExtension`, `RadiusRequest`, and `RadiusResponse` interfaces and their dependencies must be packaged in JAR format. Ensure the packaged extensions JAR files are available to the ECE runtime environment in the `ECE_Home/lib` directory.

**RequestReceived Extension**

The RequestReceived extension manipulates the CCR, authentication, or accounting request so that the CCR, authentication, or accounting request can match the business requirement and provides an immediate response that bypasses the OCS completely. This extension is called before any rating, discounting, or alteration logic has been invoked.

**Accessible Data**

- Credit Control Request
- Authentication Request
- Accounting Request
- System configuration
- Extensions data

**Modifiable Data**

You can modify the CCR, authentication, or accounting request. For example, you can manipulate AVPs to adapt to non-standard diameter implementations. Certain CCR, authentication, and accounting request types may not be supported by ECE, Diameter Gateway, or RADIUS Gateway, so a response can be created in this extension and returned immediately, bypassing the OCS.

**CustomAuth Extension**

The CustomAuth extension implements custom EAP authentication methods; for example, EAP-POTP, EAP-PSK etc.

**Accessible Data**

- EAP-Authentication-Request
- System configuration

**Modifiable Data**

You can use a custom EAP authentication method if the RADIUS client does not support EAP-TTLS or EAP-MD5.

**CustomEAPChallenge Extension**

The CustomEAPChallenge extension sends a custom access-challenge request to the RADIUS client when custom EAP authentication mechanisms are used for authentication.

**Accessible Data**

- Access-Challenge-Request
- System configuration
Implementing the Extensions Logic

Modifiable Data
You use the extension point to send the custom access-challenge request to the RADIUS client when EAP is used for authentication.

PreOCS Extension
The PreOCS extension manipulates usage request payloads before the usage request is sent to ECE, so that the request can match the business requirement. And, the PreOCS extension performs any actions related to authentication that are required before the RADIUS request is sent to ECE. This extension is called before any rating, discounting, or alteration logic has been invoked.

Accessible Data
- Credit Control Request
- Authentication Request
- Accounting Request
- ECE Usage Request
- System configuration
- Extensions data

Modifiable Data
You can modify the ECE usage request payload. For example, certain usage request manipulations can be made only when the ECE usage request payload is accessible. The usage request manipulations are done in this extension.

PostOCS Extension
The PostOCS extension manipulates CCA, accounting, or authentication responses to match the business requirement before returning the CCA, accounting, or authentication responses to the diameter client or the RADIUS client. This extension is called after charging, authentication, and accounting has been completed and recorded.

Accessible Data
- Credit Control Request
- Accounting Response
- Authentication Response
- ECE Usage Response
- Diameter Credit Control Answer
- System configuration
- Extensions data
Modifiable Data
You can modify the CCA, accounting response, and authentication response. For example, you can manipulate AVPs to adapt to non-standard diameter and RADIUS implementations.

CustomEncode Extension
The CustomEncode extension implements the custom hashing algorithm that is used on passwords for authentication.

Accessible Data
- Encoded Password
- System configuration
- Extensions data

Modifiable Data
You can use the custom hashing algorithm on passwords for authentication. For example, typically the password from the RADIUS client is hashed (stored in the hash format) for PAP authentication. However, if the password is hashed in any other format, you implement the CustomEncode extension point to hash the incoming password.

Pre-Rating Extension
The pre-rating extension enhances the usage request based on the customer, service, and balance data so that the usage request can match the business requirement. This extension is called before any rating, discounting, or alteration logic has been invoked.

Accessible Data
- ECE Usage Request
- Customer (including profile data)
- Service (including profile data)
- Balance information
- System configuration
- Extensions data

Modifiable Data
- You can modify usage requests. For example, you modify usage requests to:
  - Alter the requested quota. This is implemented in the sample extensions provided.
  - Apply special rates or discounts (such as birthday discounts) for calls based on the extended rating attributes of both calling customers and called customers.
- You can modify the values of the pricing attributes with custom logic. This enables you to override a product price.
Rating Extension

The rating extension modifies the rated results after each of the following processes: rating, alteration, sharing, and taxation.

Accessible Data

- Customer (including profile)
- Shared customer (if part of a sharing relationship)
- Product (including profile)
- Balance information
- System configuration
- Extensions
- Rated result

Modifiable Data

- You can alter rated results to modify charges, discounts, charge sharing, taxes, and item assignments. For example:
  - After rating, you can alter charges based on the zones, such as standard and geographic zones.
  - After taxation, you can alter custom item types for the rating impacts generated from ECE, such as charge, alteration, and distribution rating impacts.

Post-Rating Extension

The post-rating extension modifies the rated event.

Accessible Data

- Customer (including profile data)
- Shared customers (if part of a sharing relationship)
- Service (including profile data)
- Balance information
- System configuration
- Extensions data
- Rated result

Modifiable Data

- Modify the balance impact amount, GL code, tax code, balance element or invoice data for rating impacts generated from ECE; for example, charge, alteration, or distribution rating impacts.
- Create new tax rating impacts; for example, implement tax on tax.
Post-Charging Extension

The post-charging extension enriches the usage responses and external notifications; excluding the Advice of Charge (AOC) notifications. This extension is called after charging is completed but before the usage response is generated.

Accessible Data
- Customer (including profile data and subscriber preferences)
- Shared customers (if part of a sharing relationship, subscriber preferences)
- Service (including profile data, subscriber preferences, life cycle state)
- Balance information (including current request impacts)
- Business profile
- System configuration
- Extension data
- Rated result

Modifiable Data
You can modify the usage responses and external notifications. You can use the post-charging extension point to:
- Enrich usage responses and external notifications. You can add custom data as AVPs to the response and notification. For example, you can add a custom language preference to a customer’s subscriber preferences. The custom values will be available as diameter hooks for further propagation.
- Filter out external notifications that you do not want to be published to external systems.

You can also configure the post-charging extension point to reject the current response without impacting balances.

Post-Update Extension

The post-update extension enriches the external notifications; excluding the AoC notifications. This extension is called after receiving update requests and before publishing the external notifications.

Accessible Data
- Customer (including profile data and subscriber preferences)
- Shared customers (if part of a sharing relationship, subscriber preferences)
- Service (including profile data, subscriber preferences, life cycle state)
- Balance information (including current request impacts)
- Business profile
- System configuration
- Extension data
- Rated result
Modifiable Data

You can modify the external notifications with custom logic. You can use the post-update extension point to:

- Enrich external notifications. For example, you can add custom data to any external notification that is generated to provide additional data, such as spending limit notifications.
- Filter out external notifications that you do not want to be published to external systems. For example, when the billing is run, ECE generates subscribe-notification-request (SNR) notifications for all impacted resources. You can filter out unneeded SNR notifications and publish only required notifications to external systems.

To use the post-update extension, you must define the post-update extension’s fully qualified class name in the ECE_home/config/management/charging-settings.xml file.

Extensions Cache

The extensions framework provides a generic repository from which data required for the pre-rating, post-rating, and post-charging extensions can be uploaded and used. The data format is described in a specifications file that describes the format of the data. The extensions specification allows a DataLoader to load the data into the ECE extensions cache. Example 16–1 is an example of a specifications file for the post-rating extension:

Example 16–1  Sample Tax Table

```java
/*
 * Sample tax table
 */
ExtensionDataSpecification
   Info {
      Name 'tax_table_0001'
   }
   Payload {
      Block 'TAX_ROW' {
         String 'TAXCODE'
         String 'PKG'
         Decimal 'RATE'
         DateTime 'START'
         DateTime 'END'
         String 'LEVEL'
         String 'LIST'
         String 'DESCRIPTION'
         String 'RULE'
      }
   }
```

Example 16–2 shows the associated data to load into the cache using the specification file above:

Example 16–2  Example Data File

```text
# This is a sample csv file containing typical tax configuration data.
#
#TaxCode |Pkg |Rate    |Start      |End        |Level |List  |Description |Rule
usage   |U   |0.05    |01/01/2013 |12/31/2014 |Fed   |US    |USF         |Std
usage   |U   |0.08    |01/01/2013 |12/31/2014 |Sta   |CA    |USTA        |Std
```
### Extensions Cache API

The extensions repository provides the following APIs for managing extensions data:

- **putExtensionsData().** Takes a single key-value pair of string as a key and value being an ExtensionsData object.

- **putExtensionsDataCollection().** Takes a map of key-value pairs of string keys and value being ExtensionsData objects.

- **findExtensionsData().** Returns an ExtensionsData object for a given key.

- **getAllExtensionsData().** Returns a read-only collection of all extensions data from the repository.

### Extensions Repository Constraints

- You must generate a unique key as a string for one ExtensionsData object (entry in the extensions cache) at the time of retrieval of the extensions data from the cache.

- Because the extensions data is replicated across the whole cluster, the amount and size of data is limited to what a given Java heap can manage; you can also adjust the Java heap size. Refer to the Java provisioning guidelines.

- Changes made to the extensions data after it is loaded are expensive to make due to its cache topology. Avoid frequent updates to the extensions data, especially in a larger cluster.

- The framework does not dictate the type of data source that extensions data are loaded from. The provided **SampleExtensionsDataLoader** SDK demonstrates loading the data from a comma-separated-value (CSV) file using extensions domain-specific language APIs. This sample is a recommended design, but it should not be used as a reference about how to store data.

### Sample Extensions

This section documents the sample extensions.

#### Diameter Gateway Extension - Service

The sample program **SampleDiameterGyExtension** shows how to use the immediate-response feature based on an incoming AVP value.

**Logic:**

If Service-Context-Id is OFFLINE:

Then respond with Diameter Code DIAMETER_REDIRECT_INDICATION and set the Redirect-Host AVP value

#### Pre-Rating Extension - Dynamic Quota Management

The sample program **SamplePreRatingExtension** shows pre-rating custom logic. It illustrates sample logic for the pre-rating scenarios.
Dynamic Quota Management - Modifying Quota Based on Network Type
The SamplePreRatingExtension program shows how to use the ECE extensions API to modify the input request quantity based on the input network type where the customer balance is greater than a predefined amount.

Logic:
If ORIGIN_NETWORK network field is:
"3G_UTRAN" and USD balance greater than 50 then set quota to 10 MB
or
"4G_UTRAN" and USD balance greater than 50 then set quota to 100 MB

Dynamic Quota Management - Modifying Requested Quota
The SamplePreRatingExtension program shows how to use the ECE extensions API to update the input request to modify the requested quota. You can use this sample program to access the ECE cache to derive at a quota and then update the requested quota in the input request. ECE then uses the derived quota for allocation.

Dynamic Quota Management - Modifying Default Quota Configuration
The SampleDynamicQuotaExtension program shows how to use the ECE extensions API to update the following attributes in the input request to modify the quota configuration based on your requirements:
- Quota holding time. Specifies how long a granted quota can be idle before the reservation is released.
- Volume quota threshold. Specifies how much of the granted quota must be consumed before a subscriber can request additional quota. This attribute is configured per service, event, and number of granted units.
- Validity time. Specifies whether the validity time can be set to a fixed value per service-event combination at runtime. This attribute is independent of the number of units in the granted quota.

Pre-Rating Extension - Retrieving Function Values for Discount Expressions
The SamplePreRatingExtension program shows how to use the ECE extensions API to retrieve the value referenced by the function in a discount expression. You create a custom function in ECE that defines an event profile attribute. You can use the SamplePreRatingExtension program to call the custom function. ECE then adds the defined event profile attribute and its value to the usage request.

Logic:
If the PDC pricing specifies a 10% discount for all accounts active less than 12 months, then the logic is the following:
If customerActiveMonths value is:
< 12 then apply a discount of 10%
or
> 12 then apply a discount of 0%
Pre-Rating Extension - Overriding Price in Product Offerings

The SamplePreRatingExtension program shows how to use the ECE extensions API to override the price specified in product offerings. You create a custom function in ECE that overrides the default value of the pricing attributes in dynamic tags, which are the XML elements configured in PDC. You can use the SamplePreRatingExtension program to call the custom function. The overridden values are then populated in the event profile map in the request specification data. ECE uses the overridden values to determine the price when processing usage requests.

Rating Extension - Custom Item Assignment

The sample program SampleRatingExtension shows how to use the ECE extensions API to alter the custom item type for rating impacts. It alters custom item types for the rated results based on the data accessible through the rating extension. The default configuration for the custom item type used in the extension must exist in the ECE configuration.

Logic:
1. After taxation, determine the custom item type to be used based on the data accessible through the rating extension.
2. Assign the rating impacts to the custom bill items based on the new custom item type.

Post-rating Extension - Complex Taxation

The sample program SamplePostRatingComplexTaxationExtension shows how to use the ECE extensions API to override or augment post-rating results using complex taxation as an example. The program iterates over the tax rating periods and overrides tax impacts by modifying the rating periods for federal tax and then generates new tax periods for the state tax.

It applies the tax rate based on the pre-loaded tax configuration data in the extensions cache. The tax rate is determined based on tax code, tax time, and validity, which are all based on the request start time. The default configuration for the tax code used in the extension must exist in the ECE configuration.

Logic:
1. Determine the federal tax rate from the tax configuration table using the tax code, request start time.
2. Calculate the federal tax based on this tax rate.
3. Modify the original impact in the tax rating period based on the taxable impact from the linked charge, alteration, or distribution rating period.
4. Determine the state tax rate from the tax configuration table using the tax code, request start time.
5. Calculate state tax based on this tax rate.
6. Create new tax rating period for the state tax and link it to the original charge/alteration/distribution rating period.

This program also shows how to use the extensions API to override the invoice data in the rating result. The overridden value is persisted into the CDR output file.
Rating/Charging Extension - Triggering RAR Notifications

The following sample programs show how to use the ECE extensions API to trigger server-initiated reauthorization request (RAR) notifications in the rating and charging flow:

- SampleRarPreRatingExtension
- SampleRarPostRatingExtension
- SampleRarPostChargingExtension

These programs access the ECE cache data and trigger RAR notifications to retrieve the exact reservation balance for performing any business operation.

In the custom logic, if the `SendGenericRARNotification` is set to `true`, ECE generates generic RAR notifications for all Diameter sessions for the client and the Rating-Group and Service-Identifier are not set in those notifications. If `SendGenericRARNotification` is set to `false`, ECE generates service-specific RAR notifications with Rating-Group and Service-Identifier set in the notifications.

Post-Rating Extension - Adding or Deleting Rating Periods

The `PostRatingConsolidateRatingPeriods` sample program shows how to use the ECE extensions API to:

- Add a single rating period with the consolidated charge for all the rating periods of type CHARGE.
- Delete all the existing rating periods of type CHARGE.

You can use this sample program to access the ECE cache and override the rating periods in the final rated results by adding or deleting rating periods.

Post-Charging Extension - Adding Custom Data to Usage Responses and Notifications

The sample program `SamplePostChargingExtension` shows how to use the ECE extensions API to add custom data to the following:

- Usage responses. You add the data as AVPs. For example, you can add a custom language preference to a customer’s subscriber preferences. The custom values are available as diameter hooks for further propagation.
- External notifications. You add the data as key-value pairs. For example, you can add information such as calling number, called number, event type, and balance group to these notifications, such as credit threshold notifications.

Post-Charging Extension - Overriding Dynamic Quota

The `SamplePostChargingExtension` program shows how to use the ECE extensions API to override the quota attributes, such as quota holding time and volume quota threshold, in the usage response. You provide the data as name-value pairs. ECE then accesses the data and updates the usage response.

Post-Charging Extension - Adding or Modifying Redirection Rules

The `SamplePostChargingExtension` program shows how to use the ECE extensions API to add or modify rules for redirecting a subscriber session to a service portal applicable to the business scenario. You can add or modify them based on the customer conditions, such as whether the customer has insufficient funds or whether the customer has an inactive account.
Post-Update Extension - Enriching External Notifications

The sample program `SampleUpdateNotificationExtension` shows how to use the ECE extensions API to add custom data to external notifications that are generated to provide additional data. You add the data as name-value pairs. ECE then accesses the data and updates the external notifications.

Extensions Data Load Sample

The sample program `SampleExtensionsDataLoader` demonstrates how the extensions data repository can be used and how to load data into the repository.

The data loader used for extensions is located at `ECE_home/occesdk/source/oracle/communication/brm/charging/sdk/extensions`.

The following SDK artifacts are provided:

- **tax_configuration.spec**
  - This is a specification for tax codes. The specification expects a single block with a cardinality of 1 per ExtensionsData.
  - Contains the following attributes:
    * Tax code (String)
    * Pkg (String)
    * Rate (Decimal)
    * Start (DateTime)
    * End (DateTime)
    * Level (String)
    * List (String) Description (String)
    * Description (String)
    * Rule (String)

- **tax_configuration_data.csv**
  - A pipe-delimited CSV file. This file acts as a data source for tax codes.

- **SampleExtensionsDataLoader**
  - A class that reads the CSV file, prepares the payload as per tax specification, and uses the extensions repository to put a collection of ExtensionsData.
  - Also asserts if the number of ExtensionsData put in the Repository are the same as the total being read.

How To Use the Sample Extensions

The following procedure shows how to use the sample extensions:

1. ECE SDK is installed under `$SDK_HOME`. The directory listing is shown below:

   ```
   $ ls -l
   total 124
   drwxr-xr-x 2 ecsuser ecsuser  4096 Jun 21 10:47 bin
   drwxr-xr-x 2 ecsuser ecsuser  4096 Jun 21 10:47 bin
   drwxr-xr-x 3 ecsuser ecsuser  4096 Jun 21 10:47 bin
   drwxr-xr-x 3 ecsuser ecsuser  4096 Jun 21 10:47 config
   -rw-r--r-- 1 ecsuser ecsuser  5 Jun 21 10:47 VERSION
   ```
2. Under the source directory, create a pre-extensions or post-extensions Java Class using the Extensions API and other libraries (samples are provided as a part of the ECE SDK.)

```bash
$ cd source
$ cd oracle/communication/brm/charging/sdk/extensions
$ ls -l
```

```
total 28
-rw-r--r-- 1 ecsuser ecsuser  6427 Jun 21 10:47 SampleExtensionsDataLoader.java
-rw-r--r-- 1 ecsuser ecsuser 12194 Jun 21 10:47 SamplePostRatingComplexTaxation
-rw-r--r-- 1 ecsuser ecsuser  6066 Jun 21 10:47 SamplePreRatingExtension.java
-rw-r--r-- 1 ecsuser ecsuser XXXXX Jun 21 10:47 SamplePostChargingExtension.java
```

3. Write custom logic in Java and copy it under the directory. The Java source is under the package `oracle.communication.brn.charging.sdk.extensions`:

```
$SDK_HOME/source/oracle/communication/brm/charging/sdk/extensions
```

4. Change ECE_HOME in the script `build_deploy_extension.sh` file under `$SDK_HOME/bin/extensions`:

```bash
### configuration begin
ECE_HOME=$ECE_HOME
### configuration end
```

5. Compile the extensions class using the shell script: `build_deploy_extension.sh`.

   a. Each extensions file has to be compiled individually (similar to SDK programs).

   b. Any additional ECE or third-party library required for the extensions needs to be added to the CLASSPATH in the `build_deploy_extension.sh` script

   ```bash
   $sh $SDK_HOME/bin/extensions/build_deploy_extension.sh build SampleDiameterGyExtension
   $sh $SDK_HOME/bin/extensions/build_deploy_extension.sh build SamplePostRatingComplexTaxationExtension
   $sh $SDK_HOME/bin/extensions/build_deploy_extension.sh build SamplePreRatingExtension
   $sh $SDK_HOME/bin/extensions/build_deploy_extension.sh build SamplePostChargingExtension
   
   Do the following optional step if external data needs to be loaded. To compile the sample extensions loader use the `sample_extensions_loader.sh` shell script:

   ```bash
   $sh $SDK_HOME/bin/extensions/sample_extensions_loader.sh build SampleExtensionsDataLoader
   $sh $SDK_HOME/bin/extensions/sample_extensions_loader.sh run
   
   6. Deploy creates a single JAR file (`ece.extensions-VERSION-SNAPSHOT.jar`) with all the extensions classes and copies the JAR file under `$ECE_HOME/lib`. The JAR file is copied only to the driver node. It has to be propagated to other ECE nodes in the grid manually or use a rolling upgrade.

   ```bash
   $sh $SDK_HOME/bin/extensions/build_deploy_extension.sh deploy
   
   7. Define the pre-rating, rating, post-rating, post-charging, and post-update extensions fully-qualified class names by configuring the `charging.extensions` MBean in the `ECE Configuration` node using a JMX editor. For instructions, see...
"Configuration".

preRatingExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePreRatingExtension"
RatingExtension="oracle.communication.brm.charging.sdk.extensions.
SampleRatingExtension"
postRatingExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePostRatingComplexTaxationExtension"
potChargingExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePostChargingExtension"
diameterGyExtension="oracle.communication.brm.charging.sdk.extensions.
SampleDiameterGyExtension"
potUpdateExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePostUpdateExtension"

8. Start or restart the ECE server nodes and enable logging for the extensions by setting `oracle.communication.brm.charging.extensions.client` to DEBUG via JMX and verify that the custom extensions are executed as a part of rating logic. You can also turn on debug logging for the RATING module using the JMX console.

Validating Sample Extensions

After the server nodes are bought up initially or by using a rolling upgrade, send a sample SDK usage request. Enable debug for the RATING module and verify the server log contains the "SamplePreRatingExtension invoked" and "PostRatingComplexTaxationSampleExtension executed" messages.
This document describes how to use the sample programs included in the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) SDK for learning how to call the ECE APIs.

Topics in this document:

- About the Sample Programs
- Compiling and Running the Sample Programs
- Example of SampleDebitRefundSession

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

### About the Sample Programs

The sample programs are in the `ECE_home/ocecesdk/source/oracle/communication/brm/charging/sdk` directory.

The samples are supported on the Linux and Oracle Solaris platforms.

Each sample program includes these supporting files:

- Source files to view or modify for your own applications
- Shell scripts to compile and run the sample programs

The sample programs use the generic `.ecc` script `sdk_production_loader.ecc`.

To determine which parameter values you must use for running a sample program, you can use the help option of the sample script. All of the sample scripts have a help option which prints the usage of the sample program.

For descriptions of the methods the sample programs use, see `oracle.communication.brn.charging.sdk` in BRM ECE Java API Reference.

All of the sample programs can work with the ready-to-use sample data included with the ECE Server software installation.

**Note:** The ECE sample programs do not work well with data you load using the simulator `loader` utility.
For a list of each sample program, their descriptions, the shell scripts used to compile and run them, and the applicable .ecc script, see the following tables:

- ECE Sample Programs for Usage Requests
- ECE Sample Programs for Update Requests
- ECE Sample Programs for Policy Requests
- ECE Sample Programs for Query Requests
- ECE Sample Programs for Extension Implementations
- ECE Sample Programs for Notifications

For descriptions of the methods the sample programs use, see `oracle.communication.brm.charging.sdk` in `BRM ECE Java API Reference`.

Table 17–1 lists the usage sample programs, their descriptions, the shell scripts used to compile and run them, and the applicable .ecc script.

### Table 17–1  ECE Sample Programs for Usage Requests

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleAccountingOnOff</td>
<td>sdk_production_loader.ecc</td>
<td>sample_accounting_on_off.sh</td>
<td>Simulates an accounting on/off request being sent from the mediation client.</td>
</tr>
<tr>
<td>SampleDataSession</td>
<td>sdk_production_loader.ecc</td>
<td>sample_data_session.sh</td>
<td>Simulates a simple data session, including an INITIATE, an UPDATE and a TERMINATE request.</td>
</tr>
<tr>
<td>SampleDebitRefundSession</td>
<td></td>
<td>sample_debit_refund_session.sh</td>
<td>Shows how to send debit and refund requests with multiple values in unit-based and amount-based mode. See &quot;Example of SampleDebitRefundSession&quot;.</td>
</tr>
<tr>
<td>SampleGenericSession</td>
<td></td>
<td>sample_generic_session.sh</td>
<td>Simulates any kind of voice or data session.</td>
</tr>
<tr>
<td>SampleGprsSession</td>
<td></td>
<td>sample_gprs_session.sh</td>
<td>Simulates a GPRS session.</td>
</tr>
<tr>
<td>SampleIncrementalUsageRequestLauncher</td>
<td>sdk_production_loader.ecc</td>
<td>sample_incremental_usage_request.sh</td>
<td>Simulates a voice session with incremental mode.</td>
</tr>
<tr>
<td>SampleMultipleServicesLauncher</td>
<td>sdk_production_loader.ecc</td>
<td>sample_multiple_service.sh</td>
<td>Shows how to send usage requests for the Multiple Services Credit Control (MSCC) case (multiple subrequests are sent in a single usage request).</td>
</tr>
<tr>
<td>SamplePriceEnquiry</td>
<td>sdk_production_loader.ecc</td>
<td>sample_price_enquiry.sh</td>
<td>Sends a price enquiry request.</td>
</tr>
<tr>
<td>SampleReAuthRequest</td>
<td>sdk_production_loader.ecc</td>
<td>sample_RAR.sh</td>
<td>Sample program that shows the generation of a reauthorization request (RAR) message. Also shows how to consume notification messages. This portion of the code is for illustration only and is disabled.</td>
</tr>
</tbody>
</table>
Table 17–2  ECE Sample Programs for Update Requests

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleExternalTopUpReqLauncher</td>
<td>-</td>
<td>sample_external_topup_notification_request.sh</td>
<td>Shows how third party systems can perform direct top-ups in ECE.</td>
</tr>
</tbody>
</table>

Table 17–3 lists the policy sample programs.
### Table 17–3  ECE Sample Programs for Policy Requests

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SamplePolicySessionRequestLauncher</td>
<td>-</td>
<td>sample_policy_session_request.sh</td>
<td>Simulates a policy session. Shows how to send a policy request to ECE that requests both Sp and Sy information.</td>
</tr>
<tr>
<td>SampleSpendingLimitReportRequestLauncher</td>
<td>-</td>
<td>sample_spending_limit_report_request.sh</td>
<td>Simulates a policy Sy query request. Shows how to send a request to retrieve policy counter status information.</td>
</tr>
<tr>
<td>SampleSubscribeNotificationRequestLauncher</td>
<td>-</td>
<td>sample_subscribe_notification_request.sh</td>
<td>Simulates a policy Sp query request. Shows how to send a request to retrieve the value for a specified set of subscriber preferences and subscribe for receiving notifications when the values of the preferences change. For example, shows how to retrieve the channel a subscriber prefers for receiving policy-related notifications (SMS or email) or the language in which the subscriber prefers the notification to be written (French, English).</td>
</tr>
<tr>
<td>SampleSubscriberPreferenceUpdateRequestLauncher</td>
<td>-</td>
<td>sample_subscriber_preference_update_request.sh</td>
<td>Simulates a policy-related update request. Shows how to update the subscriber preferences in ECE.</td>
</tr>
<tr>
<td>SampleUserDataRequestLauncher</td>
<td>-</td>
<td>sample_user_data_request.sh</td>
<td>Simulates a policy Sp query request without subscription. Shows how to send a request to retrieve the values for subscriber preferences configured for a customer’s service.</td>
</tr>
</tbody>
</table>

Table 17–4 lists the query sample programs.

### Table 17–4  ECE Sample Programs for Query Requests

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleAuthenticationQuery</td>
<td>sdk_production_loader.ecc</td>
<td>sample_auth_query_request.sh</td>
<td>Sends an authentication query request.</td>
</tr>
<tr>
<td>SampleBalanceQueryRequestLauncher</td>
<td>sdk_production_loader.ecc</td>
<td>sample_balance_query_request.sh</td>
<td>Sends a balance query request.</td>
</tr>
</tbody>
</table>
Table 17–5 lists the extension sample programs.

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>sample_extensions_loader</td>
<td>Data loader for extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implementations</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>build_deploy_extension</td>
<td>Sample extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implementation</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>tax_configuration.spec</td>
<td>Sample extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implementation</td>
</tr>
</tbody>
</table>

Table 17–6 lists the notifications sample programs.

<table>
<thead>
<tr>
<th>Sample Program</th>
<th>ECC Script</th>
<th>Shell Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleDurableJmsClient</td>
<td>-</td>
<td>sample_durable_jms_client.sh</td>
<td>Simulates a durable JMS client</td>
</tr>
<tr>
<td>SampleJmsClient</td>
<td>-</td>
<td>sample_jms_client.sh</td>
<td>Simulates a JMS client</td>
</tr>
<tr>
<td>SampleJmsServer</td>
<td>-</td>
<td>sample_jms_server.sh</td>
<td>Simulates a JMS server</td>
</tr>
</tbody>
</table>

Compiling and Running the Sample Programs

You can compile and run a sample program with the shell script provided for that specific sample program as shown in Table 17–1.

To compile and run a sample program:

1. Open the ECE_home/oceceserver/config/eceTopology.conf file.
2. Un-comment the line where the sdkCustomerLoader node is defined.
   You are required to uncomment this line to be able to run the SDK sample programs.

   **Caution:** Do not run the customerLoader utility without the -incremental parameter in a production environment.

3. Change directory to the ECE server bin directory:
   
   $ cd ECE_home/oceceserver/bin

   where ECE_home is the directory in which you installed the ECE Server software.

4. Load the ECE runtime environment:
   
   $ ./ecc 'load sdk_production_loader.ecc'

5. Change directory to the ECE SDK bin subdirectory that contains the shell script for compiling and running the sample program you want to run:
   
   $ cd ECE_home/ocecesdk/bin/sample_program_directory
where `sample_program_directory` is one of the following directories

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ECE_home/occesdk/bin/extensions</code></td>
<td>Shell scripts for extension-implementation sample programs.</td>
</tr>
<tr>
<td><code>ECE_home/occesdk/bin/notification</code></td>
<td>Shell scripts for notification sample programs.</td>
</tr>
<tr>
<td><code>ECE_home/occesdk/bin/policy</code></td>
<td>Shell scripts for policy sample programs.</td>
</tr>
<tr>
<td><code>ECE_home/occesdk/bin/query</code></td>
<td>Shell scripts for query sample programs.</td>
</tr>
<tr>
<td><code>ECE_home/occesdk/bin/update</code></td>
<td>Shell scripts for update sample programs.</td>
</tr>
<tr>
<td><code>ECE_home/occesdk/bin/usage</code></td>
<td>Shell scripts for usage sample programs.</td>
</tr>
</tbody>
</table>

For example, to compile and run the `sample_voice_session.sh` sample program (the sample program for sending a voice session usage request to ECE), you must be in the `ECE_home/occesdk/bin/usage` directory.

6. Compile the sample program:

   ```shell
   $ sh ./scriptname.sh build
   ```

   You must compile the sample program once.

   For example, to compile the `sample_voice_session.sh` sample program, enter:

   ```shell
   $ sh ./sample_voice_session.sh build
   ```

7. Run the sample program.

   ```shell
   $ sh ./scriptname.sh run
   ```

   Some programs require that you enter parameters. The `run` command output gives you information about what parameters are required.

   You can also run the `run` command with no parameters. The command uses default parameter values from the SDK scripts when you run it with no parameters.

8. When you are done with the sample program, shut down the ECE runtime environment:

   ```shell
   $ ecc stop server
   ```

**Example of SampleDebitRefundSession**

Below is a usage example for running the SampleDebitRefundSession sample program.

To see the usage of any sample program, type no arguments at the command line.

Usage:

```
sample_debit_refund_session.sh build | run | defaultrun | userId requestType correlationId [BALANCE_ELEMENT_ID,AMOUNT BALANCE_ELEMENT_ID,AMOUNT ...] [TOTAL,IN,OUT TOTAL,IN,OUT ...]
```
Example of SampleDebitRefundSession

where:

- **build** compiles the related SDK source files
- **run** runs the SDK program (debit refund) according to the parameters you provide.

  You must supply all parameters in the command line.

  Alternatively, you can provide no parameters. If you provide no parameters, the SDK script invokes the ECE sample program with default parameter values.

- **defaultrun** builds and runs the SDK program.

  No parameters are required. The program uses the default parameter specified inside the shell script.

- Order of parameters are fixed and if one optional parameter is provided then all values of other optional parameters must be supplied.

- requestType is either DEBIT_AMOUNT, REFUND_AMOUNT, DEBIT_UNIT, or REFUND_UNIT.

- In case of DEBIT_AMOUNT or REFUND_AMOUNT, BALANCE_ELEMENT_ID is the well-known ISO code for balance elements, such as 840 for US Dollars or 95 for Included Minutes.

- In case of DEBIT_UNIT or REFUND_UNIT, TOTAL,IN and OUT must be specified with numbers in MB (megabytes).

Example:

```bash
# debit $10 for USD and 25 seconds Included Minutes
sample_debit_refund_session.sh run 650999777 DEBIT_AMOUNT CORR_ID 840,10 95,25

# refund $50 for USD and 5 seconds Included Minutes
sample_debit_refund_session.sh run 650999777 REFUND_AMOUNT CORR_ID 840,50 95,5
```
This document provides reference information for Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) charging utilities.

Topics in this document:

- *pin_update_items_journals*
- *query*
Use the **pin_update_items_journals** utility to process the temporary item and journal data and update the main item and journal tables. This utility is run when the value of the **StagedBillingFeeProcessing** business parameter specifies that the service charges should be aggregated to a single account-level item.

**Location**

`BRM_home/bin`

**Syntax**

```
pin_update_items_journals [-verbose] [-help]
```

**Parameters**

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

- **-help**
  Displays the syntax and parameters for this utility.

**Results**

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

**Error Handling**

When the **pin_update_items_journals** utility encounters an error while processing items and journal data in the temporary tables, it sets the PIN_FLD_BILLING_STATUS field of the bill unit (**/billinfo** object) to PIN_BILL_ERROR. In addition, it sets the appropriate bit of the PIN_FLD_BILLING_STATUS_FLAGS field of the **/billinfo** object as follows:

- Updating journal objects: PIN_BILL_FLAGS_UPDATE_JOURNALS_ERROR (bit value 0x2000)
- Updating item objects: PIN_BILL_FLAGS_UPDATE_ITEMS_ERROR (bit value 0x4000)

**Important:** If a nonpaying child bill unit caused the failure, **pin_update_items_journals** updates the billing statuses set in the PIN_FLD_BILLING_STATUS and PIN_FLD_BILLING_STATUS_FLAGS fields of the **/billinfo** object for both nonpaying child and parent bill units.

After you have resolved the processing errors, you can reprocess the items and journals data by running the **pin_update_items_journals** utility again.
query

The `query` utility provides access to ECE cache content, enabling you to execute queries on ECE Coherence caches. For information about using the `query` utility, see "Using the query Utility to Test ECE".

**Location**

```plaintext
ECE_home/occeserver/bin
```

**Syntax**

```plaintext
./query.sh
```

**Parameters**

- `-help`
  Displays the syntax for this utility.

**Results**

Look in the `ECE_home/occeserver/logs/query_out.log` file for errors.
Part I provides an overview of configuring charging in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

- ECE Notification Reference
- Sample Notification Payloads
- Specifications and Standards Compliance in ECE
This appendix lists the notifications used in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

To configure notifications, use a JMX editor such as JConsole. See “About ECE Notifications”.

Topics in this document:
- Charging Notifications
- BRM Notifications
- About the Notifications API

See also “Sample Notification Payloads”.

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

### Charging Notifications

Table A–1 lists the notifications used in online charging. These notifications are sent to the network from ECE.

<table>
<thead>
<tr>
<th>Notification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold breach</td>
<td>Sent when a customer’s balance breaches a credit threshold value, such as the threshold amount or threshold percentage that was breached.</td>
</tr>
<tr>
<td></td>
<td>You can send threshold breach notifications as asynchronous notifications, in-session notifications, or both.</td>
</tr>
<tr>
<td></td>
<td>Configure the <code>thresholdBreachNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Credit ceiling breach</td>
<td>Sent when a customer’s balance breaches a credit ceiling, such as the balance element code and balance identifier corresponding to the breach.</td>
</tr>
<tr>
<td></td>
<td>Configure the <code>creditCeilingBreachNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Credit floor breach</td>
<td>Sent when a customer’s balance breaches a credit floor, such as the balance element code and balance identifier corresponding to the breach.</td>
</tr>
<tr>
<td></td>
<td>Configure the <code>creditFloorBreachNotificationMode</code> MBean attribute.</td>
</tr>
</tbody>
</table>
**Table A–2**  Notifications Sent to the BRM Server

<table>
<thead>
<tr>
<th>Notification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advice of Charge (AoC)</td>
<td>Sent to support the Advice of Charge (AoC) supplementary service. You can send AoC notifications as asynchronous notifications, in-session notifications, or both. Configure the <code>adviceOfChargeNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Server-Initiated Reauthorization Requests (RAR)</td>
<td>Sent to update a session in response to changes that occur to a customer’s product offerings or balance (for example, a change to a charge offer or to a Friends and Family promotion). See &quot;Enabling Server-Initiated Reauthorization Requests&quot;. Configure the <code>rarNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Policy-driven charging</td>
<td>Sent to a policy client when a threshold for a policy-driven charging rule is reached. Configure the <code>spendingLimitNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Top-up</td>
<td>Sent to support server-initiated reauthorization requests (RAR). See &quot;Enabling Server-Initiated Reauthorization Requests&quot; for information. Configure the <code>topUpNotificationMode</code> MBean attribute.</td>
</tr>
</tbody>
</table>

---

**BRM Notifications**

*Table A–2* lists the notifications sent to the BRM server from ECE.

**Table A–2**  Notifications Sent to the BRM Server

<table>
<thead>
<tr>
<th>Notification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger billing</td>
<td>Sent when a subscriber starts a charging session near the time billing is set to run in BRM. This ensures that billing generates new recurring grants and charges on time. Configure the <code>billingNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Replenish POID IDs</td>
<td>Sent to obtain POID IDs. For tracking a customer’s balance impacts for a given item type, ECE maps the service and event combinations in a usage request to its relevant item type. ECE sends information to BRM so that it can obtain the POID IDs when it does not have enough that can be used (ReplenishPoidIDNotificationEvent). Configure the <code>replenishPoidNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>Life-cycle transitions</td>
<td>Sent when a customer’s life-cycle state has changed. Configure the <code>lifeCycleTransitionNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>First usage validity initialization</td>
<td>Sent to synchronize validity of first usage balance elements from ECE to BRM and from ECE to NCC Notification Gateway. Configure the <code>firstUsageValidityInitNotificationMode</code> MBean attribute.</td>
</tr>
<tr>
<td>External top-ups</td>
<td>Updates balances when a top-up made through external systems.</td>
</tr>
</tbody>
</table>

---

**About the Notifications API**

You can use notifications in your custom applications. For example, a client application could use the notification to send an email to the customer informing the customer that a credit threshold was breached.

For Java API descriptions of the service event objects published as external notifications, see `oracle.communication.brm.charging.servicecontext` in *BRM ECE Java API Reference*.

The notification framework transforms the Java payload of the ECE service event into XML. The payloads published into the JMS notification queue (topic) are XML strings.
The following are examples of the XML structure for external notification messages that can be published.

**THRESHOLD_BREACH_EVENT (Breach Direction Up)**
The payload published for a threshold breach (breach direction up) uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <CreditThresholdBreachNotification>
    <NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
    <PublicUserIdentities>
      <PublicUserIdentity>123</PublicUserIdentity>
    </PublicUserIdentities>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>-4.00</CurrentBalance>
    <ThresholdAmount>-4.5</ThresholdAmount>
    <ThresholdPercent>55.0</ThresholdPercent>
    <BreachDirection>THRESHOLD_BREACH_UP</BreachDirection>
    <AlertType>2</AlertType>
    <Reason>0x01</Reason>
    <OperationType>USAGE</OperationType>
    <SubOperationType>INITIATE</SubOperationType>
  </CreditThresholdBreachNotification>
</Notification>
```

**THRESHOLD_BREACH_EVENT (Breach Direction Down)**
The payload published for a threshold breach (breach direction down) uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <CreditThresholdBreachNotification>
    <NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
    <PublicUserIdentities>
      <PublicUserIdentity>123</PublicUserIdentity>
    </PublicUserIdentities>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>-9.00</CurrentBalance>
    <ThresholdAmount>-8.0</ThresholdAmount>
    <ThresholdPercent>20.0</ThresholdPercent>
    <BreachDirection>THRESHOLD_BREACH_DOWN</BreachDirection>
    <AlertType>2</AlertType>
    <Reason>0x01</Reason>
    <OperationType>USAGE</OperationType>
    <SubOperationType>INITIATE</SubOperationType>
  </CreditThresholdBreachNotification>
</Notification>
```
This document provides samples of XML strings that Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) publishes into the JSM notification queue for different notification types.

See also "ECE Notification Reference".

Topics in this document:
- Aggregated Threshold Breach Event (Aggregated Based on Balance Element ID)
- Billing Event
- Credit Ceiling Breach Event
- Credit Floor Breach Event
- External Top-up Event
- First Usage Validity
- Life-Cycle Transition
- Replenish POID ID Event
- Spending Limit
- Subscriber Preference Event
- Threshold Breach Event (Breach Direction Down)
- Threshold Breach Event (Breach Direction Up)
- Top-up Event

See also:
- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

### Aggregated Threshold Breach Event (Aggregated Based on Balance Element ID)

The payload published for an aggregated threshold breach (aggregated based on Balance Element ID) uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
```
<AggregatedCreditThresholdBreachNotification>
  <NotificationType>AGGREGATED_THRESHOLD_BREACH_EVENT</NotificationType>
  <PublicUserIdentities>
    <PublicUserIdentity>123</PublicUserIdentity>
  </PublicUserIdentities>
  <BalanceElementId>840</BalanceElementId>
  <BalanceElementCode>USD</BalanceElementCode>
  <CurrentBalance>-3.00</CurrentBalance>
  <ThresholdAmount>[-4.5, -3.5]</ThresholdAmount>
  <ThresholdPercent>[55.0, 65.0]</ThresholdPercent>
  <BreachDirection>THRESHOLD_BREACH_UP</BreachDirection>
  <OperationType>USAGE</OperationType>
  <SubOperationType>INITIATE</SubOperationType>
</AggregatedCreditThresholdBreachNotification>
First Usage Validity

Sample Notification Payloads

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <FirstUsageValidityNotification>
    <NotificationType>FIRST_USAGE_VALIDITY_INIT_NOTIFICATION_
Life-Cycle Transition

The payload published for a life cycle transition notification uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <LifeCycleTransitionNotification>
    <NotificationType>LIFECYCLE_TRANSITION_NOTIFICATION_EVENT</NotificationType>
    <PublicUserIdentities>
      <PublicUserIdentity>0049100120</PublicUserIdentity>
    </PublicUserIdentities>
    <CustomerId>3135579</CustomerId>
    <ExternalReference>1</ExternalReference>
    <ProductId>3134811</ProductId>
    <ProductType>TelcoGsmTelephony</ProductType>
    <LifecycleState>103</LifecycleState>
    <ExpirationTime>1439653419867</ExpirationTime>
    <SubscriberPreferences>
      <SubscriberPreference PublicUserIdentity="316-20150813-143831-0-21484--153892112-slc06bui:TelcoGsmTelephony, 0049100120:TelcoGsmTelephony">
        <SubscriberPreferencesInfo>
          <PreferenceName>Language</PreferenceName>
          <PreferenceValue>French</PreferenceValue>
        </SubscriberPreferencesInfo>
      </SubscriberPreference>
    </SubscriberPreferences>
  </LifeCycleTransitionNotification>
</Notification>
```

Replenish POID ID Event

The payload published for a replenish POID ID notification uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <ReplenishPoidIdNotification>
    <NotificationType>REPLENISH_POID_ID_NOTIFICATION_EVENT</NotificationType>
  </ReplenishPoidIdNotification>
</Notification>
```
<SchemaName>1</SchemaName>
<Quantity>10000</Quantity>
</ReplenishPoidIdNotification>
</Notification>

**Spending Limit**

The payload published for a spending limit notification uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <SpendingLimitNotification>
    <NotificationType>SPENDING_LIMIT_NOTIFICATION</NotificationType>
    <CustomerId>340876</CustomerId>
    <PublicUserIdentities>
      <PublicUserIdentity>9986068473</PublicUserIdentity>
      <PublicUserIdentity>login123</PublicUserIdentity>
    </PublicUserIdentities>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>2</CurrentBalance>
    <ConsumedReservation>3</ConsumedReservation>
    <Unit>MegaBytes</Unit>
    <Breaches>
      <OfferProfileName>Offer1</OfferProfileName>
      <LabelName>Fair Usage</LabelName>
      <StatusLabel>low qos</StatusLabel>
      <DeltaToNextThreshold>8</DeltaToNextThreshold>
    </Breaches>
    <DuplicateEvent>True</DuplicateEvent>
  </SpendingLimitNotification>
</Notification>
```

**Subscriber Preference Event**

The payload published for creating a subscriber preference notification uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <CreateSubscriberPreference>
    <NotificationType>SUBSCRIBER_PREFERENCE_NOTIFICATION_EVENT</NotificationType>
    <CustomerId>340876</CustomerId>
    <ProductInfo>
      <ProductId>12345</ProductId>
      <PublicUserIdentities>
        <PublicUserIdentity>9886753556</PublicUserIdentity>
        <PublicUserIdentity>login</PublicUserIdentity>
      </PublicUserIdentities>
    </ProductInfo>
    <SubscriberPreferencesInfo>
      <PreferenceName>Language</PreferenceName>
      <PreferenceValue>English</PreferenceValue>
    </SubscriberPreferencesInfo>
    <SubscriberPreferencesInfo>
      <PreferenceName>Channel</PreferenceName>
      <PreferenceValue>Email</PreferenceValue>
    </SubscriberPreferencesInfo>
  </CreateSubscriberPreference>
</Notification>
```
The payload published for modifying a subscriber preference notification uses the following format:

```xml
<ModifySubscriberPreference>
  <CustomerId>customer1</CustomerId>
  <ProductInfo>
    <ProductId>12345</ProductId>
  </ProductInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Language</PreferenceName>
    <PreferenceValue>English</PreferenceValue>
  </SubscriberPreferencesInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Channel</PreferenceName>
    <PreferenceValue>Email</PreferenceValue>
  </SubscriberPreferencesInfo>
</ModifySubscriberPreference>
```

The payload published for deleting a subscriber preference notification uses the following format:

```xml
<DeleteSubscriberPreference>
  <CustomerId>customer1</CustomerId>
  <ProductInfo>
    <ProductId>12345</ProductId>
  </ProductInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Language</PreferenceName>
    <PreferenceValue>English</PreferenceValue>
  </SubscriberPreferencesInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Channel</PreferenceName>
    <PreferenceValue>Email</PreferenceValue>
  </SubscriberPreferencesInfo>
</DeleteSubscriberPreference>
```

The payload published for a threshold breach (breach direction up) uses the following format:

```xml
<CreateSubscriberPreference>
  <CustomerId>customer1</CustomerId>
  <ProductInfo>
    <ProductId>12345</ProductId>
  </ProductInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Language</PreferenceName>
    <PreferenceValue>English</PreferenceValue>
  </SubscriberPreferencesInfo>
  <SubscriberPreferencesInfo>
    <PreferenceName>Channel</PreferenceName>
    <PreferenceValue>Email</PreferenceValue>
  </SubscriberPreferencesInfo>
</CreateSubscriberPreference>
```
Threshold Breach Event (Breach Direction Up)

The payload published for a threshold breach (breach direction up) uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <CreditThresholdBreachNotification>
    <NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
    <PublicUserIdentities>
      <PublicUserIdentity>123</PublicUserIdentity>
    </PublicUserIdentities>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>-4.00</CurrentBalance>
    <ThresholdAmount>-4.5</ThresholdAmount>
    <ThresholdPercent>55.0</ThresholdPercent>
    <BreachDirection>THRESHOLD_BREACH_UP</BreachDirection>
    <AlertType>2</AlertType>
    <Reason>0x01</Reason>
    <OperationType>USAGE</OperationType>
    <SubOperationType>INITIATE</SubOperationType>
  </CreditThresholdBreachNotification>
</Notification>
```

Top-up Event

The payload published for a top-up uses the following format:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
  <RARNotification>
    <NotificationType>TOP_UP_NOTIFICATION_EVENT</NotificationType>
    <PublicUserIdentities>
      <PublicUserIdentity>123</PublicUserIdentity>
      <PublicUserIdentity>456</PublicUserIdentity>
    </PublicUserIdentities>
    <ActiveSessions>
      <ActiveSessionId>SESSION1</ActiveSessionId>
      <ActiveSessionId>SESSION2</ActiveSessionId>
    </ActiveSessions>
    <ProductType>VOICE</ProductType>
    <ProductId>test</ProductId>
    <CustomerId>12345</CustomerId>
    <DuplicateEvent>True</DuplicateEvent>
  </RARNotification>
</Notification>
```
</RARNotification>
</Notification>
This appendix describes the specifications and standards used in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

See also:

- List of Charging Features
- About Configuring Charging in Elastic Charging Controller
- Managing Online Charging Sessions
- Connecting ECE to a Diameter Client

About Specifications and Standards Compliance

The ECE charging API aligns with the Remote Authentication Dial In User Service (RADIUS) Accounting Request for Comments (RFC) specifications and with the standards described in the 3rd Generation Partnership Project (3GPP) Technical Specifications (TS). ECE charging supports any 3GPP sub-domain; some are listed here as examples:

- PS (Packet Switched) connections
- CS (Circuit Switched) connections
- WLAN (Wireless Local Area Network)
- IMS (IP-Multimedia Subsystem)
- PCRF (Policy and Charging Rules Function) and Sy/Sp (Sh) interfaces

The ECE charging API is extensible; it can accommodate proprietary extensions of the standards.

The ECE Java API aligns with the Diameter Ro, Diameter CCA, Diameter Rf, and RADIUS message formats. Network mediation software programs (client applications) that support these protocols can send usage requests to ECE.

The following 3GPP Technical Specifications (TS) relate to ECE charging functionality.

- "3GPP TS 32.240 Telecommunication management; Charging management; Charging architecture and principles"

For online charging, ECE exposes a Java API based on Diameter Ro, which is extensible for supporting any extension or variation.

ECE implements the following functionality for online charging:
– Online Charging Function modules:
  * Session Based Charging Function (SBCF)
  * Event Based Charging Function (EBCF)
– Rating Function (RF)
– Account Balance Management Function (ABMF)

■ "3GPP TS 32.260 Telecommunication management; Charging management; IP Multimedia Subsystem (IMS) charging"
■ "3GPP TS 32.299 Telecommunication management; Charging management; Diameter charging applications"

For offline charging, ECE exposes a Java API based on DIAMETER Rf, which can be called from the offline mediation system. The Java interface has functionality close to that of the Rf interface described in 3GPP 32.299 and is extensible for supporting any extension or variation.

Oracle Communications Offline Mediation Controller uses this interface to load CDRs into ECE for charging.

■ GB922 TM Forum Information Framework (SID).

The following RADIUS RFCs relate to ECE charging functionality.

■ RFC 2866, RFC 2867, RFC 2868, RFC 2869, RFC 3579

ECE aligns with the Diameter Credit-Control Application charging functionality described in Internet Engineering Task Force (IETF) Network Working Group RFC 4006.

The following 3GPP Technical Specifications (TS) relate to the Policy and Charging Rules Function (PCRF) and ECE:

■ "3GPP TS 29.219 Policy and charging control: Spending limit reporting over Sy reference point"
  The Sy interface is located between the PCRF and Online Network Mediation Controller. It enables the transfer of customer spending information.
■ "3GPP TS 29.329 Sh interface based on the Diameter protocol"
  The Sp (implemented as Sh) interface is located between the SPR (Subscription Profile Repository) and PCRF. It enables the retrieval of customer identities and profile information.