Oracle® Communications Billing and Revenue Management

ECE Implementing Charging



Release 16.26.0
G54463-01
October 2026

Oracle Communications Billing and Revenue Management ECE Implementing Charging, Release 16.26.0

G54463-01

Copyright © 2019, 2026, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

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.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

About This Content

1 About Configuring Charging in Elastic Charging Engine

About Usage Charging in ECE 1
About Integrating Applications with ECE 2
About Customizing ECE 2
About Balance Management in a Prepaid Session 3

Part I Using the ECE Java API

2 About the ECE API

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

3 Configuring Multiple Services Credit Control

About Multiple Services Credit Control 1

4 Advice of Charge and Advice of Promotion

About Advice of Charge 1
About Advice of Promotion 1

5 Configuring Top-Ups

Integrating Top-Up Clients 1
Detecting Duplicate Top-Up Requests 1

Copyright © 2019, 2026, Oracle and/or its affiliates.

Using the Top-Up API 2

6 Configuring Balance Queries

Integrating Balance Query Clients 1
About Sending Authentication Queries 1
About Sending Balance Queries 1
Configuring Debit Request History 2
About Balance Query Requests 2

Part II Working with BRM

7 Synchronizing Data Between ECE and the BRM Database

About Synchronizing Data Between BRM and ECE 1
Setting Up Synchronization Between BRM and ECE 1
Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates 2
Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE 2
Configuring Post-Commit BRM-to-ECE Synchronization Behavior 3
About Processing Customer Updates When the ECE Persistence Database Is Unavailable 4
Configuring Write-Behind Persistence for Customer Updates 4
How ECE Gets Historical Data from the BRM Database 6
Determining the Discount Sharing Group Owner for Rating 6
Configuring Discount Sharing Group Owner Determination at Installation or Before Restart 7
Configuring Discount Sharing Group Owner Determination During Runtime 7

8 Loading ECE Rated Events into BRM

About Sending Rated Events to the BRM Database 1
Adding a Rated Event Publisher Instance 1
Configuring Rated Event Publisher 2
Configuring Item Assignment for Rated Events 3
Configuring Item Assignment During Rerating 4
Updating the Payload Configuration for Item Assignment During Rerating 5
Enabling Item Assignment During Rerating at Installation or Before Restart 5
Enabling Item Assignment in Rerating During Runtime 5
Configuring Lifecycle States in ECE for BRM 6
Configuring Lifecycle States During Runtime 6
Configuring Lifecycle States At Installation 7
Loading Modified Custom Service Life Cycles into ECE 8

Copyright © 2019, 2026, Oracle and/or its affiliates.

Including or Excluding a Customer's Remaining Balance in Rated Events 8
Accessing ECE Configuration MBeans 9

9 Generating POIDs for Rated Events

About Generating POIDs in ECE 1
Configuring ECE to Generate POIDs for Prepaid Events 2
Enabling Prepaid Event Partitions in BRM 2
Configuring Cluster ID 3
Enabling POID Generation for Prepaid Events in ECE 4

Part III Managing ECE Notifications

10 Configuring Notifications in ECE

About ECE Notifications 1
Enabling ECE to Publish Notifications to External Applications 2
Enabling Specific Notification Types 2
Replicating Notifications to Kafka Partitions Dedicated to External Applications 4
Replicating Notifications to Kafka Topics Dedicated to External Applications 6
Publishing Asynchronous Notifications to a Separate Kafka Topic 7
Publishing Asynchronous Notifications to Multiple External Topics 9
Enabling In-Session Group Notifications in ECE 9
Including Rollover Balances in Notifications 11
Enriching Notifications Using ECE Extensions 11
About Configuring BRM Gateway to Process ECE Notifications 11
Configuring BRM Gateway to Process ECE Notifications 12
Configuring the BRM Gateway 13
Configuring Multiple BRM Gateway for Multi-Schema Deployments 14
Connecting BRM Gateway to Kafka Topics and BRM 16
Configuring WebLogic Queues for BRM Gateway 18
Setting the Transaction Timeout for the connectionFactory 18
Considerations for Using a Non-WebLogic Server JMS Provider 19
Modifying JMS Credentials for Publishing External Notifications 19

11 Configuring Subscriber Preferences

How BRM Manages Subscriber Preferences 1
Tasks for Including Subscriber Preference Information in Notifications 1
Configuring ECE to Enrich External Notifications with Subscriber Preference Information 2
Configuring Which Subscriber Preferences to Collect 3
Configuring Group Notifications 5

Copyright © 2019, 2026, Oracle and/or its affiliates.

Collecting Subscriber Preferences Using Client Tools 6
Granting CSR Access to Subscriber Preferences 6

Part IV Managing Charging Sessions

12 Understanding Linear and Non-Linear Rating in ECE

How ECE Applies Linear Rating 1
How Non-Linear Rating Manages Tariff Boundaries 2
How ECE Reserves Resources for Non-Linear Rating 2
How ECE Creates Separate Rated Events with Nonlinear Rating and TTC 3
Using RAR for In-Session Quota and Policy Updates 3
How ECE Applies Non-Linear Rating at Time-Model Boundaries 3
Configuring Linear or Non-Linear Rating 3
Configuring ECE to Use Linear Rating for All Usage Events 4
Configuring Linear or Non-Linear Rating for Specific Products 4
Configuring Linear or Non-Linear Rating for Product and Event Combinations 5

13 Managing Midsession-Rated Events

About Midsession-Rated Events 1
Configuring ECE to Generate Midsession-Rated Events 2
Generating Midsession-Rated Events for Charge Offer Changes 3
Generating Midsession-Rated Events When the USU is Block Missing 5
Configuring Threshold-Based Tier Selection for Split Rating Periods 5
Customizing the Worst-Case Charging Reservation 6
Viewing the Reason for a Midsession-Rated Event 6

14 Managing Online Charging Sessions

Managing Dynamic Charging Overrides for Online Sessions 1
Enabling Server-Initiated Reauthorization Requests 2
Customizing Server-Initiated Reauthorization for Sharing Groups 3
Configuring ECE to Return Remaining-Balance Information in Usage Responses 3
Configuring Default Taxes in ECE 5
Logging Prepaid Usage Overage 6
Managing Direct Debit Data in ECE Cache 7
Configuring ECE to Redirect Subscriber Sessions to a Service Portal 7
Enabling Match Factor in ECE 10
Customizing Consumption Order of Loan and Default Balances 11
Location-Based Charging 11
About Processing Conditional Balance Impacts 12

Copyright © 2019, 2026, Oracle and/or its affiliates.

Customizing ECE to Apply Fixed Charge for Zero Usage 12
Accessing ECE Configuration MBeans 14
About Overload Protection in ECE 15
Redirecting Customer Sessions to a Service Portal 15
Returning FUI in Initiate Responses 16
Configuring Redirection Rules 16
Redirection-Rule Syntax 17

15 Managing Session Start and End Times

Using Session Connect Time for Charging 1
Optimizing Network Signaling 1
Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers 2

16 Managing Reservations for Online Sessions

Configuring Reservation Expiration and Validity 1
Configuring a Minimum Quantity for Reservation 2
Configuring Reservation Quota for Services 3
Managing Dynamic Quotas for Online Sessions 4
Triggering RAR Notifications for Ongoing Sessions 5

17 Managing Rounding and Consumption Rules

Configuring Rounding for a Resource 1
Configuring Rounding for Reverse Rating on Multiple RUMs 2
Configuring Systemwide Consumption Rules for Balances 3
Configuring the Consumption Order for Offers with the Same Priority 4
Configuring Balance Impact Rounding in ECE 4
Configuring Rounding for a Currency Balance Impact 5
Configuring Rounding for a Noncurrency Balance Impact 6
Configuring Rounding When Authorizing Multiple RUMs 6

18 Managing Granted Allowances

Processing Granted Allowances Before Applying Usage Charges 1
About Automatically Renewing Granted Allowances 1
About Allowance Renewals 2
About Allowance Renewals During Online Charging 2
About Allowance Renewals During Offline Charging 3
Configuring Automatic Renewals 3
Setting Up Automatic Renewals for Offline Charging 3

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring Granted Allowances 4
Enabling Notifications for Automatically Renewed Granted Allowances 4

19 Managing Charging During System Downtimes

Configuring Diameter Gateway to Bypass Rating During ECE Downtime 1
Managing the Persistence of Usage Requests During ECE Downtime 1
Replaying Persisted Requests into ECE 2
Configuring How ECE Manages Active Sessions When Network Elements Fail 3

20 Using Impact-Based Candidate Selection

How Impact-Based Candidate Selection Works 1
Selecting Candidates Across Rating Periods 1
Selecting a Charge Selector Candidate 2
Selecting a Generic Selector Candidate 2

Part V Integrating with External Systems

21 Creating Kafka Topics for ECE Integration

Setting Up ECE and Gateways for Apache Kafka Using Topic-Based Routing 1
Enabling the Topic-Based Kafka Architecture 2
Generating Kafka Topics with Topic-Based Routing 2
Migrating Existing ECE Systems Gateways to Apache Kafka Using Topic-Based Routing 3
Setting Up ECE and Gateways for Apache Kafka Using Partition-Based Routing 4
Configuring Kafka Partition Assignments for Gateways 5
Determining the Number of Partitions Required 6
Connecting ECE and Gateways to Kafka Topics 6
Recording Failed ECE Usage Requests 10
Understanding and Configuring ECE Prepaid Usage Overage 10

22 Connecting ECE to a 5G Client

About the HTTP Gateway 1
About Determining the Charging Type 2
About Sending Notifications to HTTP Gateway 2
Integrating HTTP Gateway with 5G Networks 2
Configuring Registration Details for the HTTP Gateway Server 3
Configuring Multiple Primary and Secondary NRF Registration Servers 8
Example NRF JSON Payload File 9
Configuring NF Services 11

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring HTTP Gateway for Convergent Charging 14
Editing the HTTP Gateway Mediation Specification File 17
Mapping nCHF Request Fields to ECE Payload Attributes 19
Retaining IPv4 and IPv6 PDP Address Information in the ECE Payload 19
Checking for Client-Side Timeouts in HTTP Gateway 21
Configuring ECE to Publish Notifications to HTTP Gateway 22
Securing Communication Between HTTP Gateway and the NRF, PCF, and SMF 23
Configuring Communication Through SCP 24
Starting the HTTP Gateway 26
Using the ECE REST API 26

23 Generating CDRs for External Systems

About Using the HTTP Gateway 1
About Generating CDRs 1
About Saving CDR Files to Disk 2
About the CDR Generation Process 2
Setting Up ECE to Generate CDRs 3
Accessing ECE Configuration MBeans 4
Configuring HTTP Gateway for CDR Generation 4
Configuring the CDR Gateway 5
Configuring the CDR Formatter 6
Configuring the CDR Formatter Plug-in 8
About Trigger Types 9
Triggers for Convergent Charging Events 9
Triggers for Roaming Events 11

24 Connecting ECE to a Diameter Client

Overview of Network Integration Using Diameter Gateway 1
Network Integration for Sp and Sy Interface (Policy) Requests 2
Network Integration for Gy Interface Requests 3
How Diameter Gateway Creates Usage Requests 5
About Usage Request Fixed Attributes 5
Editing the Mediation Specification File 6
Network Integration for Gy Balance Query Requests 8
Network Integration for Gy Top-Up Requests 8
Sending Multiple-Service Credit Control (MSCC) Requests from Diameter Gateway 9
Configuring Subscriber ID Lookups 9
Configuring Diameter Gateway Nodes for Online Charging 11
Checking for Client-Side Timeouts in Diameter Gateway 15
Configuring ECE to Publish Notifications to Diameter Gateway 16

Copyright © 2019, 2026, Oracle and/or its affiliates.

Adding Custom AVPs for Usage Requests 17
Using Incremental or Cumulative Accounting for Usage Requests 17
Configuring Accounting Mode for Diameter Gateway 18
Configuring WebLogic Queues for Notifications 19
Configuring Alternative Diameter Peers for Notifications 20
Viewing Active Diameter Peers 20
Including Loan Sub-Balance in Balance Queries 20
Enriching Balance Query Responses with Subscriber Preferences 21
About Handling Requests When Charging Servers Are Unavailable 21
Cleaning Sy Sessions from the Cache 22

25 Connecting ECE to a RADIUS Client

Overview of Authentication, Accounting, and Disconnection Using RADIUS Gateway 1
About RADIUS Gateway Authentication 1
Authenticating Access Requests by Using PAP 2
Authenticating Access Requests by Using CHAP 3
Authenticating Access Requests by Using EAP 4
About RADIUS Gateway Accounting 5
About Accounting-Start and Accounting-Stop Requests 6
About Accounting-On and Accounting-Off Requests 8
About Accounting-Interim-Update Requests 9
About RADIUS Gateway Disconnection 10
Setting Up Authentication, Accounting, and Disconnection Using RADIUS Gateway 11
Loading Data Keys Extracted from BRM into ECE 11
Customizing the RADIUS Data Dictionary 12
About the RADIUS Data Dictionary 12
Creating a Custom Data Dictionary 12
Selecting a RADIUS Data Dictionary When Using Different NAS Vendors 12
Adding Custom Vendor-Specific Attributes 12
Loading the RADIUS Mediation Specification Data 13
About Mapping RADIUS Network Attributes to Event Attributes 15
Mapping RADIUS Network Attributes to Event Attributes 15
Configuring ECE and RADIUS Gateway to Manage Disconnect Requests 16
Configuring RADIUS Gateway to Send Disconnect Messages to Your RADIUS Client 17
Configuring ECE to Publish Notifications to RADIUS Gateway 17

26 Configuring Policy-Driven Charging

About Policy-Driven Charging 1
About Group-Based Policy-Driven Charging 3
Policy-Driven Charging Example 3

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring Policy-Driven Charging 3
About ECE and Policy Clients 4
How ECE Processes Policy Requests for Online Network Mediation System 4
Configuring Breach Tolerance for Policy-Tier Thresholds 6
About Integrating Policy Clients with ECE 8
About the ECE Sy and Sp Interface 8
About the ECE Sy Interface 8
About the ECE Sp Interface 9
Querying for Extended Subscriber Preference Information in Sp Query 9
About a Combined ECE Sy and Sp Interface 10
About Calculating Maximum Authorization for Policy-Driven Charging Sessions 10
Configuring ECE to Reject Spending Limit Requests Without Counters 11
About the Policy Management API 11
Configuring ECE to Treat Uninitialized Policy Counters as Zero 11

Part VI Customizing ECE

27 Customizing Rating

Operational Considerations 1
Configuring Extensions 1
About Performance with Extensions 2
Using Value Map Results in Rating Extensions 2
About Logging in Extensions 3
About Extension Exceptions 4
About Extension Security 4
Extension Points 4
BRM Gateway Request Processing Extension Points 4
Diameter-Request Processing Extension Points 5
HTTP Gateway Request Processing Extension Points 6
RADIUS-Request Processing Extension Points 7
Authentication Extension Points 7
Accounting Extension Points 9
Update-Request Processing Extension Points 10
Usage-Request Processing Extension Points 10
Implementing the Extensions Logic 12
Customizing a DSL Selector Extension for Impact-Based Selection 20
BRMCustomOpCodeCall Extension 20
CustomAuth Extension 20
CustomEAPChallenge Extension 21
CustomEncode Extension 21

Copyright © 2019, 2026, Oracle and/or its affiliates.

OCSBypass Extension 21
PreOCS Extension 21
PreProcessor Extension 22
PostOCS Extension 22
PostOCSBalanceQuery Extension 23
Pre-Rating Extension 23
Post-Rating Extension 23
Rating Extension 24
RequestReceived Extension 25
Post-Charging Extension 25
Post-Update Extension 26
Extensions Cache 26
Extensions Cache API 27
Sample Extensions 28
How To Use the Sample Extensions 28
Validating Sample Extensions 30
BRM Gateway Extension – Creating Opcode Flist 30
Diameter Gateway Extension – Gy Service 30
Diameter Gateway Extension – Sy Service 30
HTTP Gateway Extension – Service 31
OCSBypass Extension – Bypassing Rating 31
Pre-Rating Extension – Dynamic Quota Management 31
Dynamic Quota Management – Modifying Quota Based on Network Type 31
Dynamic Quota Management – Modifying Requested Quota 32
Dynamic Quota Management – Modifying Default Quota Configuration 32
Pre-Rating Extension – Retrieving Function Values for Discount Expressions 32
Pre-Rating Extension – Generating Midsession-Rated Event 33
Pre-Rating Extension – Overriding Price in Product Offerings 33
Post-Rating Extension – Complex Taxation 33
Post-Rating Extension – Generating Midsession-Rated Events 34
Post-Rating Extension – Adding or Deleting Rating Periods 34
Post-Charging Extension – Adding Custom Data to Usage Responses and Notifications 35
Post-Charging Extension – Overriding Dynamic Quota 35
Post-Charging Extension – Adding or Modifying Redirection Rules 35
Post-Charging Extension - Enriching Notifications 35
Post-Charging Extension – Creating Custom Notifications for Top Ups 35
Post-Charging Extension – Modifying the Renewal Message 36
Post-Update Extension – Enriching External Notifications 36
Rating/Charging Extension – Triggering RAR Notifications 36
Rating Extension – Custom Item Assignment 37
Extensions Data Load Sample 37

Copyright © 2019, 2026, Oracle and/or its affiliates.

28 ECE Sample Programs

About the ECE Sample Programs 1
Finding the Sample Programs 1
Descriptions of the Sample Programs 2
Compiling and Running the Sample Programs 6
Example of SampleDebitRefundSession 7
Compiling and Deploying SampleRatedEventFormatterCustomPlugin 8

29 Testing ECE

About ECE Testing Utilities 1
About Loading Sample Data 2
About Performance MBean 2
Changing Time and Date to Test ECE 3
Using the query Utility to Test ECE 4
- Example: Query the Subscriber Base Balance Summary 5
- Example: Query a Customer Balance 5

Verifying that Usage Requests Can Be Processed 6
- Starting ECE Nodes in the Cluster 7
- Running the Simulator to Send Usage Requests 7
- Verifying that Balances Are Impacted in ECE 8

Verifying That ECE Notifications Are Published to the JMS Topic 8
- Disabling the Publishing of ECE Notifications to the JMS Topic 8

Verifying that Friends and Family Calls Are Processed 9
Verifying That Closed User Group Calls Are Processed 10
Verifying That Balance Impacts Are Assigned to Bill Items 12
Verifying That Payloads Are Correctly Formed 13

Part VII ECE Utilities

30 Charging Utilities

query 1

A Sample Notification Payloads

Aggregated Threshold Breach Event (Aggregated Based on Balance Element ID) A-1
Billing Event A-2
Credit Ceiling Breach Event A-2
Credit Floor Breach Event A-2
Custom Notification for BRM Gateway A-3

Copyright © 2019, 2026, Oracle and/or its affiliates.

External Top-up Event A-3
First Usage Validity A-4
Life-Cycle Transition A-4
Replenish POID ID Event A-5
Spending Limit A-5
Subscriber Preference Event A-5
Threshold Breach Event (Breach Direction Down) A-7
Threshold Breach Event (Breach Direction Up) A-7
Top-up Event A-7
Enriched Notification A-8

B Specifications and Standards Compliance in ECE

About Specifications and Standards Compliance B-1

Copyright © 2019, 2026, Oracle and/or its affiliates.

About This Content

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

Audience

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

1

About Configuring Charging in Elastic Charging Engine

Oracle Communications Elastic Charging Engine (ECE) is a powerful tool for implementing charging in various scenarios.

Topics in this document:

- About Usage Charging in ECE
- About Integrating Applications with ECE
- About Customizing ECE
- About Balance Management in a Prepaid Session

For an overview of ECE, see "About Billing and Revenue Management" and "ECE System Architecture" in *BRM Concepts*.

For information about administering ECE, see "ECE System Administration" in *BRM System Administrator's Guide*.

About Usage Charging in ECE

ECE offers a versatile charging solution, enabling the following types of usage charging:

- **Online Charging**: Real-time rating of events during a session, such as in prepaid calls.
- **Offline Charging**: Batch rating of events, typically for post-paid telephone usage.

ECE receives usage requests containing event data required for rating. For instance, to rate a phone call, ECE needs the calling number and the start and end times. Upon receiving a usage request, ECE identifies the customer and their associated charge offer for rating the event.

Each type of service and event needs to be rated differently. For example, some events are rated by measuring duration, and some by measuring volume. When you configure events and services, you create event definitions that specify the data needed for charging the event. The event definitions are sent to ECE, and are stored in an ECE cache. When an event occurs, ECE uses the event definition to select a usage request builder, creating the usage request.

A default set of event definition data is included with ECE, but custom services and events can be accommodated by enriching event definitions in Pricing Design Center (PDC) and creating customized event definitions.

Usage requests are created when network mediation clients submit data to the ECE client:

- Online Charging: Real-time events, like prepaid calls, are routed from the network to the Diameter Gateway or HTTP Gateway, which utilizes the ECE client to generate a usage request. ECE processes the request, authorizes the call, and sends a response back to the gateway. ECE also manages interactions with the network to handle update requests, reauthorizations, and top-ups during the call. After the call ends, ECE rates the call.

Copyright © 2019, 2026, Oracle and/or its affiliates.

In addition to prepaid calls, online charging can be used for any service that the subscriber connects to and uses in real time, such as broadband access, digital content, streaming radio, and cable television.

- Offline Charging: Call Detail Records (CDRs) are processed by Oracle Communications Offline Mediation Controller, which acts as an ECE client application. Offline Mediation Controller creates usage requests for ECE to rate the events.
  Offline charging is used for batch rating of events, typically from post-paid telephone usage. Offline Mediation Controller performs mediation and normalization tasks, such as checking for duplicate calls and assembling calls that arrive in multiple records.

After an event is rated, ECE sends the rated event data to the BRM database, and the customer's balance is updated in both ECE and in the BRM database. The same process is used for loading online charging events and offline charging events.

> **Note**
>
> By default, when an increment quantity is configured for charging, ECE consumes the available balance according to the increment quantity. If the remaining available balance is less than the amount required to satisfy the increment quantity, there can be a leftover balance that remains ungranted and unconsumed.
>
> If the **reverseRateUseAllBalances** attribute is set to **true**, ECE can consume the maximum available balance for increment-quantity rounding scenarios, including when the remaining available balance is less than the increment quantity value.

About Integrating Applications with ECE

You can integrate external applications and network with ECE using the following:

- **Diameter Gateway**: Allows you to integrate ECE with a Diameter client. See "Connecting ECE to a Diameter Client".
- **HTTP Gateway**: Allows you to integrate ECE with a 5G network. See "Connecting ECE to a 5G Client".

  After the HTTP Gateway is set up, your 5G client can send usage requests to ECE through the ECE REST API for online charging, offline charging, or both. See *REST API Reference for Elastic Charging Engine*.
- **RADIUS Gateway**: Allows you to integrate ECE with a RADIUS client. This allows ECE to authenticate access requests and process accounting requests for online charging when your customers use your terminal server or NAS to connect to ECE. See "Connecting ECE to a RADIUS Client".
- **ECE Java API**: Provides a means to integrate client applications with ECE, such as for integration a top-up client. ECE is pre-integrated with Offline Mediation Controller for offline charging. ECE uses one of the gateways for network integration and online charging. See "About the ECE API".

About Customizing ECE

You can customize ECE in the following ways:

- Use the ECE SDK sample programs. For example, it shows how third-party systems can perform direct top ups in ECE. See "ECE Sample Programs".

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Use ECE extensions to customize ECE and its gateways. For example, implement pre-rating and post-rating customizations. See "Customizing Rating".

About Balance Management in a Prepaid Session

In prepaid sessions, ECE, the Diameter Gateway or HTTP Gateway, and the network system collaborate to manage events in real-time. When a subscriber initiates a prepaid call, the network sends authentication and authorization requests to ECE, which processes them immediately to establish 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 customer's ID with those 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 temporary or permanent credit 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 ends, ECE sends information about the rated event to the BRM database and updates the subscriber's balance in the BRM database. ECE then returns any unused reserved balance amounts to the customer's balance.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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. However, other charge offers owned by the same customer and that share the balance object with the original charge offer are also cleaned up.

During a prepaid 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 can sometimes not connect to 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.

  In this situation, ECE can cancel the authorization and return any reserved balances to the customer's account.
- 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), it 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 specific language). See "Configuring Subscriber Preferences".
- If the session uses policy-driven charging, it tracks the balance thresholds that trigger credit-limit notifications to the policy controller. Both in-session and out-of-session notifications are supported.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part I

Using the ECE Java API

This part provides information about using the Oracle Communications Elastic Charging Engine (ECE) Java API. It contains the following chapters:

- About the ECE API
- Configuring Multiple Services Credit Control
- Advice of Charge and Advice of Promotion
- Configuring Top-Ups
- Configuring Balance Queries

Copyright © 2019, 2026, Oracle and/or its affiliates.

2

About the ECE API

You use the Oracle Communications Elastic Charging Engine (ECE) API to integrate ECE with third-party clients, such as top-up systems.

Topics in this document:

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

About the ECE API

ECE is preintegrated with Oracle Communications Offline Mediation Controller. To integrate other clients, such as top-up systems, you use the ECE SDK and ECE API. See *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 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 *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

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Refund_Amount
- PriceEnquiry
- StartAccounting
- UpdateAccounting
- AccountingOn
- AccountingOff

To send usage requests to ECE, client applications can 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 *Elastic Charging Engine 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 operation types shown in Table 2-1.

Table 2-1 Charging Operation Types Supported by ECE

Operation Type	Description
Initiate	Commencement of a session-based charging operation.
Update	Continuation of a session-based charging operation.
Terminate	Conclusion of a single non-session based charging operation.
Cancel	Complete cancellation of a session-based charging operation.
Refund_Amount	Refund a specific amount to a specific balance resource.
Refund_Unit	Refund a calculated amount, based on units consumed, to the impacted resource(s).
Debit_Amount	Debit a specific amount to a specific balance resource.
Debit_Unit	Debit a calculated amount, based on Units consumed, to the impacted resource(s).
Price_Enquiry	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.
Start_Accounting	Begin tracking usage without incurring balance impacts.
Update_Accounting	Continue tracking usage without incurring balance impacts.
Balance_Query	Return the user balance.
Accounting_On_Off	Clean left open session and reservation for a specific network element.

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.brm.charging.messages.query** in *Elastic Charging Engine Java API Reference*.

About the Custom Plug-in APIs

You can use the custom plug-in APIs to:

- Format rated events into the format required by an external system. To do so, use the **SampleRatedEventFormatterCustomPlugin.java** sample custom plug-in in the ECE SDK package.
- Write rated events in JSON format to the Kafka Server. To do so, use the **SampleRatedEventFormatterKafkaCustomPlugin.java** sample plug-in in the ECE SDK package.

For more information, see **oracle.communication.brm.charging.ratedevent.custom.CustomPlugin** in *Elastic Charging Engine Java API Reference*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

3 Configuring Multiple Services Credit Control

You can configure Multiple Services Credit Control in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- About Multiple Services Credit Control

About Multiple Services Credit Control

ECE supports Multiple-Service Credit Control (MSCC) requests in which a Diameter application performs credit control for multiple 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 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 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 a 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. For more information, see "ECE Sample Programs".

Copyright © 2019, 2026, Oracle and/or its affiliates.

4

Advice of Charge and Advice of Promotion

You can configure Advice of Charge (AOC) and Advice of Promotion (AOP) in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- About Advice of Charge
- About Advice of Promotion

About Advice of Charge

ECE supports the 3GPP Advice of Charge (AoC) supplementary service, which informs customers about the cost of a requested service in monetary or nonmonetary format. ECE can provide AoC at the beginning or 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 in a JMX editor, such as JConsole. See "Accessing 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5

Configuring Top-Ups

You can configure top-ups in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- Integrating Top-Up Clients
- Detecting Duplicate Top-Up Requests
- Using the Top-Up API

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, and then 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".

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 in a JMX editor, such as JConsole. See "Accessing 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.

Using the Top-Up API

To validate top-up requests, 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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 *Elastic Charging Engine Java API Reference*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

6

Configuring Balance Queries

You can configure third-party clients to query Oracle Communications Elastic Charging Engine (ECE) for balance information.

Topics in this document:

- Integrating Balance Query Clients
- About Sending Authentication Queries
- About Sending Balance Queries
- Configuring Debit Request History
- About Balance Query Requests

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 SUMMARY, DETAILED, ALL, and TURBO 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 balances for one or more subscribers.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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. 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.

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 in a JMX editor, such as JConsole. See "Accessing 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.

About Balance Query Requests

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: The balance query response contains the balance element ID and the total balance at the balance element level.
- DETAILED: The balance query response contains the balance element ID and the detailed balance for each balance element. This includes balance element specification information (such as credit limits), grantor information (such as grantor ID and grantor type), reservation information (such as the active and consumed reservation), and rollover balance information (such as the rollover balance and the number of times the balance has been rolled over).

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> In Tariff Time Change (TTC)-based charging scenarios, a balance item's active reservation can exceed its current balance. When this occurs, ECE applies the TTC calculation to both the before-TTC and the after-TTC portions of the reservation to adjust the active reservation amount returned in a DETAILED balance query. The returned amount can be lower than the active reservation held for the session.

- ALL: The balance query mode specifies to return all the sub-balances after a particular time stamp. It is in the EM Gateway response.
- TURBO: The balance query mode specifies to return information about the balance element level, the current principal balance, and the loan balance. It is in the Diameter Gateway response.

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 *Elastic Charging Engine Java API Reference*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part II

Working with BRM

This part provides information about how Oracle Communications Billing and Revenue Management (BRM) and Oracle Communications Elastic Charging Engine (ECE) work together. It contains the following chapters:

- Synchronizing Data Between ECE and the BRM Database
- Loading ECE Rated Events into BRM
- Generating POIDs for Rated Events

Copyright © 2019, 2026, Oracle and/or its affiliates.

7

Synchronizing Data Between ECE and the BRM Database

You synchronize data with the Oracle Communications Billing and Revenue Management (BRM) database to ensure that Oracle Communications Elastic Charging Engine (ECE) is rating service usage events with the most current pricing data.

Topics in this document:

- About Synchronizing Data Between BRM and ECE
- Setting Up Synchronization Between BRM and ECE
- How ECE Gets Historical Data from the BRM Database
- Determining the Discount Sharing Group Owner for Rating

About Synchronizing Data Between BRM and ECE

When customer data is updated in the BRM database, the updates must be applied synchronously (in real time) to ECE. For example, when a CSR adds, cancels, or modifies an account or when an adjustment such as a cycle fee is applied to an account balance, that information must be updated in ECE so that ECE can properly rate service usage events.

The Oracle Data Manager (DM) synchronizes customer data between BRM and ECE. The Oracle DM sends customer data in a business event to an Advanced Queuing (AQ) database queue, where ECE Customer Updater retrieves the business event and updates the information in ECE cache, allowing ECE to rate events using the latest customer information. For more information, see "Synchronizing Account Data Between BRM and ECE" in *BRM System Administrator's Guide*.

You can view or modify the list of BRM business events that are sent to ECE by using the *ECE_home***/brm_config/payloadconfig_ece_sync.xml** file.

Setting Up Synchronization Between BRM and ECE

To set up your system to synchronize customer data between BRM and ECE:

1. Create and configure your AQ database queues. See "Configuring Your AQ Database Queues" in *BRM System Administrator's Guide*.
2. Configure BRM to send account data updates to ECE in real time. See "Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates".
3. Configure BRM to retrieve real-time balances for a service from ECE. See "Configuring the Connection Manager to Get Real-Time Balances for a Service from ECE".
4. Configure BRM to enable post-commit behavior and tune the timeout used for BRM update requests to reduce the likelihood of inconsistencies between BRM and ECE customer data synchronization. See "Configuring Post-Commit BRM-to-ECE Synchronization Behavior"

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 0.0.0.0
   -cm em_group ece PCM_OP_ECE_PUBLISH_EVENT
   -cm em_pointer ece ip emGateway_host emGateway_port
   ```

   where:

   - *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 0.0.0.0, replace it 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 Post-Commit BRM-to-ECE Synchronization Behavior

In some deployments, BRM completes a transaction successfully while the BRM-to-ECE synchronization for the same transaction fails or times out. This can leave ECE updated while BRM rolls back, or BRM updated while ECE does not receive the update. You can reduce the likelihood of these inconsistencies by enabling the post-commit behavior and tuning the timeout used for BRM update requests.

When you enable the post-commit behavior, ECE commits BRM-synchronized updates only after BRM commits its database transaction. BRM also commits ECE cache updates only after the BRM database commit.

> **Note**
>
> - If you enable post-commit behavior, plan for a temporary "customer in transaction" lock. Usage requests can fail until BRM completes and the retry limit is reached. The retry count depends on your **responseTimeout** setting, which is set in the **override-values.yaml** file for cloud native deployments and the **charging-settings.xml** file for on-premise deployments.

To configure post-commit behavior and timeouts:

1. Verify that BRM-to-ECE synchronization completes within the configured timeout. Review whether EM Gateway timed out during BRM-to-ECE synchronization (for example, due to slow ECE response or a network disconnect).
2. Set the EM Gateway timeout for BRM update requests. Increase the EM Gateway timeout by setting **updateRequestServerTimeout** to a value that matches your site load and test results in the **override-values.yaml** file for cloud native deployments and the **charging-settings.xml file** for on-premises deployments.
3. Enable post-commit behavior in ECE. Set **brmPostCommitEnabled** to **true** in the **override-values.yaml** file for cloud native deployments and the **charging-settings.xml** file for on-premises deployments to ensure that ECE commits only after BRM commits.
4. Enable post-commit behavior in BRM. Set **ece_post_commit_enabled** to **1** in the *BRM_home*/**sys/cm/pin.conf** file to ensure that BRM commits ECE cache updates only after the BRM database commit.

Copyright © 2019, 2026, Oracle and/or its affiliates.

About Processing Customer Updates When the ECE Persistence Database Is Unavailable

ECE stores customer data in cache and persists selected cache updates to the ECE persistence database. By default, ECE uses write-through persistence. You can configure ECE to use write-behind persistence so that it can continue processing specified customer update operations during a temporary ECE persistence database outage.

> **Note**
>
> Write-behind persistence lets ECE complete customer update operations without waiting for the ECE persistence database. It can improve latency and let ECE operate during a temporary database outage. However, if a cache partition is lost before queued updates are written to the database, those updates can be lost and BRM and ECE data can become inconsistent.

When configured to use write-behind persistence, ECE can continue processing the following operations during a temporary ECE persistence database outage:

- Creating subscribers
- Deleting existing subscribers
- Changing IMSIs

> **Note**
>
> ECE does not support deleting a subscriber that was created while the ECE persistence database was unavailable and whose queued write-behind operations have not yet been persisted. In that scenario, the delete operation might fail.

During customer update processing, ECE assembles product-level, customer-level, and internal balances from cache. To avoid unnecessary access to the persistence database, ECE assembles an internal balance only when it is already in cache. If the internal balance is not in cache, ECE does not retrieve it from the persistence database during the update flow.

ECE uses write-behind persistence for public user identity updates. This prevents public user identity update processing from waiting for the ECE persistence database when the database is temporarily unavailable. When the persistence database becomes available, ECE persists the queued write-behind operations to the database.

Configuring Write-Behind Persistence for Customer Updates

To support customer update processing when the ECE persistence database is temporarily unavailable, configure write-behind persistence for the **PublicUserIdentity** cache.

> **Note**
>
> When making the changes in the procedure below, ensure that you add the snippets below into the delivered cache configuration. Do not replace the file's entire contents with these snippets.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To configure write-behind persistence for customer updates:

1. Open the **charging-cache-config-persistence.xml** file. For cloud native deployments, the ECE Helm chart provides this file through the **charging-settings.yaml** ConfigMap.
2. In the existing cache-scheme-mapping section, add the following mapping for the **PublicUserIdentity** cache:

```
<cache-mapping>
    <cache-name>PublicUserIdentity</cache-name>
    <scheme-name>near-federated-wb</scheme-name>
    <init-params>
        <init-param>
            <param-name>transient</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>write-delay</param-name>
            <param-value>1s</param-value>
        </init-param>
    </init-params>
</cache-mapping>
```

3. In the existing caching-schemes section, add the following write-behind federated scheme:

```
<federated-scheme>
    <scheme-name>xref-federated-persisted-wb</scheme-name>
    <scheme-ref>xref-federated</scheme-ref>
    <backing-map-scheme>
        <transient>{transient false}</transient>
        <read-write-backing-map-scheme>
            <scheme-ref>rw-persist-wb</scheme-ref>
            <internal-cache-scheme>
                <local-scheme>
                    <unit-calculator>BINARY</unit-calculator>
                </local-scheme>
            </internal-cache-scheme>
        </read-write-backing-map-scheme>
    </backing-map-scheme>
</federated-scheme>
```

4. In the existing caching-schemes section, add the following **near-cache** scheme:

```
<near-scheme>
    <scheme-name>near-federated-wb</scheme-name>
    <front-scheme>
        <local-scheme>
            <eviction-policy>HYBRID</eviction-policy>
            <high-units>20500000</high-units>
        </local-scheme>
    </front-scheme>
    <back-scheme>
        <federated-scheme>
            <scheme-ref>xref-federated-persisted-wb</scheme-ref>
        </federated-scheme>
    </back-scheme>
    <invalidation-strategy>all</invalidation-strategy>
    <autostart>true</autostart>
</near-scheme>
```

5. Add the following Coherence JVM property:

```
-Dcoherence.rwbm.writebehind.remove.default=true
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

- For on-premises deployments, add this property to the ECE JVM tuning profile (**defaultTuningProfile.properties**).
- For cloud native deployments, add this property to **charging.ecs.jvmOpts** in the ECE **override-values.yaml** file for **oc-cn-ece-helm-chart**, then run a Helm upgrade to apply the change.

6. Restart ECE to apply the configuration changes.

How ECE Gets Historical Data from the BRM Database

Because a gap can occur between usage and rating, customer information can change during that time. ECE might need historical customer information to rate the usage correctly. For example, a customer might change the phone number before a call is rated, in which case ECE must look up the account data associated with the previous phone number.

Historical information is also important when the owner of a discount sharing group changes. When discount sharing group owner determination is enabled, ECE uses sharing agreement ownership information to determine which owner was effective when the usage occurred. This enables ECE to rate delayed or backdated usage against the owner that was effective at the time of the usage. For more information, see "Determining the Discount Sharing Group Owner for Rating".

To retrieve historical information, ECE gets data from audited objects. ECE uses current and audited sharing agreement data when loading customer information so that historical sharing agreement information is available for event-time processing.

By default, BRM auditing is disabled 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 requires. See "object_auditing.pl" in *BRM System Administrator's Guide*.

Determining the Discount Sharing Group Owner for Rating

If a discount sharing group has had an ownership change, ECE determines whether to apply a usage event to the previous owner or the new owner.

By default, the **sharingAgreementOwnerResolutionEnabled** attribute is set to **false**. When the attribute is set to **false**, ECE uses the owner ID stored in the member-side **UsedSharingAgreement** data to identify the discount sharing group owner.

You can configure ECE to determine the owner based on the effective time of the usage event. When this behavior is enabled, ECE rates usage that occurs before the owner change based on the previous owner and usage that occurs at or after the owner change based on the new owner.

For delayed or backdated usage, ECE uses the owner that was effective when the usage occurred.

For example, if an owner change is effective at 10:30:

- ECE rates usage with an effective time of 10:20 based on the previous owner regardless of whether the usage is received before or after 10:30.
- ECE rates usage with an effective time of 10:30 based on the new owner.
- ECE rates usage with an effective time of 10:40 based on the new owner.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> Before changing the owner of a discount sharing group, terminate all active sessions for group members. ECE does not support sessions that continue across the owner-change time.

To configure how ECE determines the discount sharing group owner for rating, set the **sharingAgreementOwnerResolutionEnabled** ECE attribute to one of the following in the **charging-settings.xml** file for on-premises or the **override-values.yaml** file for cloud native:

- **true**: ECE determines the owner based on the effective time of the usage event.
- **false** (default): ECE uses the existing sharing-customer discovery behavior and does not determine the owner based on the effective time of the usage event.

You can configure discount sharing group owner determination in one of two ways:

- At installation or before restart: see "Configuring Discount Sharing Group Owner Determination at Installation or Before Restart".
- During runtime: see "Configuring Discount Sharing Group Owner Determination During Runtime".

Configuring Discount Sharing Group Owner Determination at Installation or Before Restart

You can configure the **sharingAgreementOwnerResolutionEnabled** attribute during installation by editing the **charging-settings.xml** file. To do so:

1. Open the *ECE_home*/**config/management/charging-settings.xml** file.
2. Set the **sharingAgreementOwnerResolutionEnabled** attribute to **true**.
3. Restart ECE. See "Starting and Stopping ECE" in *BRM System Administrator's Guide*.

Configuring Discount Sharing Group Owner Determination During Runtime

You can use a JMX editor to configure the **sharingAgreementOwnerResolutionEnabled** attribute when ECE is running.

To configure the **sharingAgreementOwnerResolutionEnabled** attribute:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes** and select **sharingAgreementOwnerResolutionEnabled**.
5. Set the value of the attribute to **true**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> Changes to the **sharingAgreementOwnerResolutionEnabled** attribute take effect for subsequent requests. You do not need to restart ECE. If the owner cache does not contain the required timeline entry, ECE uses the member-side owner ID. Reload or rebuild the cache only to populate missing historical owner data.

Copyright © 2019, 2026, Oracle and/or its affiliates.

8

Loading ECE Rated Events into BRM

You load rated events from Oracle Communications Elastic Charging Engine (ECE) into the Oracle Communications Billing and Revenue Management (BRM) database by using Rated Event Loader.

Topics in this document:

- About Sending Rated Events to the BRM Database
- Adding a Rated Event Publisher Instance
- Configuring Rated Event Publisher
- Configuring Item Assignment for Rated Events
- Configuring Lifecycle States in ECE for BRM
- Including or Excluding a Customer’s Remaining Balance in Rated Events
- Accessing ECE Configuration MBeans

About Sending Rated Events to the BRM Database

After usage events are rated, ECE sends the rated event data to the BRM database by using Rated Event Loader and updates the customer's balance in both ECE and in the BRM database. The same process is used for loading online charging events and offline charging events.

For more information, see "About Loading Rated Events into the BRM Database" in *BRM Loading Rated Events*.

Adding a Rated Event Publisher Instance

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 add a Rated Event Publisher instance:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.ratedEventPublishers**.
4. Expand **Operations**.
5. Select **addRatedEventPublisherConfiguration**.
6. Enter a value for the instance **name** parameter.
7. Click **addRatedEventPublisherConfiguration**.
8. Use Elastic Charging Controller (ECC) to start the RE Publisher instance.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring Rated Event Publisher

To configure Rated Event Publisher:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations.***instance_name*.
4. Specify values for the fields in Table 8-1.

Table 8-1 Fields for Configuring Rated Event Publisher Connection

Name	Default	Description and Guideline
dataStoreConnection	localhost:5000	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 preconfigured Oracle NoSQL database system. The default is **localhost:5000** for connecting to a standalone Oracle NoSQL database system (KV-Lite).
dataStoreName	kvstore	This parameter configures the data store name to an Oracle NoSQL database system. The data store name is for using a preconfigured data store in an Oracle NoSQL database system. The default is **kvstore** for using a standalone Oracle NoSQL database system (KV-Lite).
name	noSQLConnection1	This parameter should match the **noSQLConnectionName** parameter in the **charging.ratedEventPublishers.***instance_name* configuration.

5. Expand the **charging.ratedEventPublishers.***instance_name* node.
6. Expand **Attributes**.
7. Specify values for the fields in Table 8-2.

Table 8-2 Fields for Configuring Rated Event Publisher

Name	Default	Description and Guideline
noSQLConnectionName	noSQLConnection1	This parameter should match the **name** parameter in the **charging.ratedEventPublishers.***instance_name* configuration for the correct connection configuration to the Oracle NoSQL database.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 8-2 (Cont.) Fields for Configuring Rated Event Publisher

Name	Default	Description and Guideline
threadPoolSize	4	This parameter configures the number of threads in the thread pool. Multiple threads can be used in a RatedEventPublisher module where each thread can publish rated events to an Oracle NoSQL database system independently. The valid number is greater than zero. For best performance, Oracle recommends that you set this parameter to the number of Oracle NoSQL database partitions. Setting the number of threads higher than the number of partitions does not increase performance. Threads that you configure higher than the number of partitions are not used.

8. Use ECC to stop and restart Rated Event Publisher.

Configuring Item Assignment for Rated Events

You configure item assignments 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 assignments.

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 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 as the **misc** item type. To map voice and data events to the **misc** item, ECE maps itemType="misc" to itemTag="VOICE_DATA_misc". The XML file that stores this configuration shows how the mapping works.

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

In this example, the VOICE_DATA_misc item tag includes two 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 will 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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Expand the **ECE Configuration** node.
3. Expand **charging.itemAssignmentConfig**.
4. Expand **Attributes**.
5. 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 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 delayed billing interval is set in the **ConfigBillingDelay** business parameter. See "Configuring Delayed Billing" in *BRM Configuring and Running Billing*.
   - **itemAssignmentInRerating**: Enter **true** to enable item assignment when ECE processes rerating requests. Enter **false** to disable item assignment during rerating. The default is **false**.
6. 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 reserved at ECE startup for the specified schema.
   d. Click the **setPoidQuantity** button.

Configuring Item Assignment During Rerating

By default, ECE does not assign charges to bill items during rerating. You can enable item assignment by setting the **itemAssignmentInRerating** configuration parameter.

- If set to **true**, ECE performs item assignment during rerating, linking the recalculated charge to the correct billing item.
- If set to **false**, ECE does not perform item assignment during rerating.

To configure ECE to perform item assignment during rerating:

1. Update the payload configuration to include the fields required for item assignment. See “Updating the Payload Configuration for Item Assignment During Rerating”.
2. Enable ECE to perform item assignment during rerating in one of these ways:
   - At installation or before a restart: See "Enabling Item Assignment During Rerating at Installation or Before Restart".
   - During runtime: See "Enabling Item Assignment in Rerating During Runtime".

Copyright © 2019, 2026, Oracle and/or its affiliates.

Updating the Payload Configuration for Item Assignment During Rerating

Configure BRM to send the fields required for item assignment to ECE during the synchronization process.

To update the payload configuration:

1. Open the *ECE_home*/**brm_config/payloadconfig_ece_sync.xml** file.
2. In the **RetatedAccounts** element, locate the **Services** extended info section and add the **ItemList** and **NextItemList** fields if they are not already present. For example:

```
<ReratedAccounts Source="EVENT" PinFld="PIN_FLD_RERATE_ACCOUNTS"
                 DataFrom="PIN_FLD_RERATE_ACCOUNTS" Tag="ReratedAccounts">
   ...
   <ExtendedInfo PinFld="PIN_FLD_SERVICES" Tag="Services">
       <Field PinFld="PIN_FLD_POID" Tag="ServiceObj"/>
       <Field PinFld="PIN_FLD_LIFECYCLE_STATE" Tag="LifeCycleState"/>
       <Field PinFld="PIN_FLD_SERVICE_STATE_EXPIRATION_T"
Tag="ExpirationTime"/>
       <Field PinFld="PIN_FLD_ITEM_POID_LIST" Tag="ItemList"/>
       <Field PinFld="PIN_FLD_NEXT_ITEM_POID_LIST" Tag="NextItemList"/>
   </ExtendedInfo>
   ...
</ReratedAccounts>
```

3. Save the file and restart ECE. See "Starting and Stopping ECE" in *BRM System Administrator's Guide*

Enabling Item Assignment During Rerating at Installation or Before Restart

You can configure the **itemAssignmentInRerating** parameter during installation or before a restart by editing the *ECE_home*/**config/management/charging-settings.xml** file. To do so:

1. Open your **charging-settings.xml** file.
2. In the server element, set the **itemAssignmentInRerating** attribute to true. For example:

```
<server config-
class="oracle.communication.brm.charging.appconfiguration.beans.BizParamCon
fig"
        ...
        itemAssignmentInRerating="true"
        ...
>
</server>
```

3. Restart ECE. See "Starting and Stopping ECE" in *BRM System Administrator's Guide*.

Enabling Item Assignment in Rerating During Runtime

To configure ECE to assign items for rerated events:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans" in *BRM System Administrator's Guide.*
2. Expand the **ECE Configuration** node.

Copyright © 2019, 2026, Oracle and/or its affiliates.

3. Expand **charging.server**.
4. Expand **Attributes** and select **itemAssignmentInRerating.**
5. Set the attribute to **true**.

> **Note**
>
> By default, the **itemAssignmentInRerating** attribute is set to **false**. No restart is required for this change to take effect.

Configuring Lifecycle States in ECE for BRM

ECE supports the BRM subscriber lifecycle state feature. If the subscriber lifecycle 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 lifecycle state feature is enabled in BRM, ECE supports the custom subscriber life cycle, which has the following states: Preactive, Active, Recharge Only, Credit Expired, Fraud Investigated, Dormant, Suspended, and Closed. See "Creating Custom Service Life Cycles" in *BRM Managing Customers* for more information.

You must configure the lifecycle states in ECE so they stay synchronized with the lifecycle states you add in BRM.

You configure lifecycle states by defining the following:

- The product type that the rule applies to, such as DATA.
- The state transitions the rule applies to. For example, the rule defines when a state can transition *to* or *from* the **Recharge Only** state.
- The conditions for transitioning from one state to another. For example,
  - When a session has an **Active** state, a balance element of USD, a request type of Terminate, and the credit limit has been reached, transition to the **Recharge Only** state with an expiration of 2 days.
  - When a session has a **Recharge Only** state, a balance element of USD, and the balance is replenished to below the credit limit, transition to the **Active** state with an expiration of 30 days.
- Whether the rule and transition apply to usage or external top-up events.

You can configure lifecycle states in ECE at installation using the **charging-settings.xml** file or during runtime by updating either the **charging-settings.xml** file in an on-premise system or the details as part of the helm chart and running a helm upgrade with a reload job.

Configuring Lifecycle States During Runtime

To configure lifecycle states in ECE for BRM during runtime:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.lifecycleConfiguration**.
4. Expand **Operations**.
5. Select **addLifecycleDetails**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

6. Enter values for these parameters:
   - **ProductType**: (String value) The product type for which you want to configure the rule and transition, such as VOICE, DATA, or SMS.
   - **Rule**: (String value) The rule for the specified product type to allow or disallow usage. The characters need to be escaped while provided the same input field.
   - **Transition**: (String value) The state transitions for the specified product type.
   - **IsUsage**: (Boolean value) True specifies that the rule and transition is for USAGE. False specifies that the rule and transition is for EXTERNAL_TOP_UP.
7. Click the **addLifecycleDetails** button.

Configuring Lifecycle States At Installation

You configure the lifecycle state mapping during installation in the **charging-settings.xml** file.

1. Open your **charging-settings.xml** file.
2. Define your states in the <lifecycleStateMappingConfiguration> element. For example, this defines the **PREACTIVE** lifecycle state. Include the **productStatus** attribute if you use BRM product status values for lifecycle state mapping.

```
<lifecycleStateMappingConfiguration
    config-
class="oracle.communication.brm.charging.appconfiguration.beans.productstate.LifecycleStateMapper">
  <lifecycleStateMappingGroup config-class="java.util.ArrayList">
    <lifecycleState
        config-
class="oracle.communication.brm.charging.appconfiguration.beans.productstate.LifecycleState"
        state="101" stateName="PREACTIVE" productStatus="10102"/>
  </lifecycleStateMappingGroup>
</lifecycleStateMappingConfiguration>
```

   Following are some key attributes in the definition:
   - **state**: specifies the lifecycle state ID from the *BRM_home*/**sys/data/config/config_lifecycle_states.xml** file.
   - **stateName**: specifies the ECE lifecycle state name, such as **Preactive**, **Active**, **Recharge Only**, or **Dormant**.
   - **productStatus**: specifies the BRM product status value that maps to this ECE lifecycle state, such as Active (**10100**), Inactive (**10102**), or Closed (**10103**).
3. Define the product type to which the rule applies in the **<productLifecycleConfiguration>** element. For example, this specifies the **VOICE** product type:

```
<productLifecycleConfiguration config-
class="oracle.communication.brm.charging.appconfiguration.beans.lifecycle.ProductLifecycleConfiguration" productType="VOICE">
```

4. Define the rules in the **<productLifecycleConfiguration>** element. You can configure the rule based on:
   - Items within the payload. For example:

```
getObject(&quot;oracle.communication.brm.charging.rating.RatingContext#request/payload/attribute?USAGE_DIRECTION/toString&quot;)
```

   - The default aliases. For example:

Copyright © 2019, 2026, Oracle and/or its affiliates.

```
DEFAULT_ALIASES.put("@birthday", GET_OBJECT + CTX +
      "#getCustomerBdayMmdd?Birthday\")");
```

- The business profile of the customer. For example:

  ```
  (business_profile([name:"PREPAID"])} == "true" )
  ```

- The request attribute. For example:

  ```
  getObject(&quot;oracle.communication.brm.charging.rating.RatingContext#request/
  eventType&quot;) == &quot;EventSessionTelcoGsmVoice&quot;)
  ```

- The attributes from the payload, using **getObject/getInteger/getBigDecimal/getBoolean** method calls for relevant attributes.

5. The rules defined can be used to define transitions. For example,

   - `(business_profile([name:"PREPAID"])} == "true")`

     The above condition causes the state transition to take place only for the PREPAID profile and can be used in the transition:

     ```
     <transitionConfiguration config-
     class="oracle.communication.brm.charging.appconfiguration.beans.lifecycle.Transit
     ionConfiguration" transition="(((@productState == "PREACTIVE") &&
     (@usageDirection == "0")) && ({business_profile([name:"PREPAID"])} == "true"))
     => [state:"ACTIVE",expiration:"2",duration:"days"]"/>
     ```

   - `(business_profile([name:"POSTPAID"])} == "true")`

     The above condition causes the state transition to take place only for the POSTPAID profile and can be used in the transition:

     ```
     <transitionConfiguration config-
     class="oracle.communication.brm.charging.appconfiguration.beans.lifecycle.Transit
     ionConfiguration" transition="(((@productState == "PREACTIVE") &&
     (@usageDirection == "0")) && ({business_profile([name:"POSTPAID"])} == "true"))
     => [state:"ACTIVE",expiration:"2",duration:"days"]"/
     ```

Loading Modified Custom Service Life Cycles into ECE

To update the application configuration:

1. Update the lifecycle configuration in the **charging-settings.xml** file.
2. Go to the *ECE_home***/bin** directory.
3. Run the following command:

   ```
   reloadChargingConfiguration.sh charging.lifecycleConfiguration
   ```

After running the script, the new configuration is loaded. This allows ongoing traffic to use the previous configuration until the reload succeeds, which prevents any failures.

See "reloadChargingConfiguration.sh" in *BRM System Administrator's Guide* for more information on the **reloadChargingConfiguration.sh** script.

Including or Excluding a Customer's Remaining Balance in Rated Events

ECE can send the current balance and loan balance information with rated events. This allows your custom client application to display balance data with rated events. These balances can be either currency or non-currency, depending on the type of resource the transaction impacts.

Copyright © 2019, 2026, Oracle and/or its affiliates.

By default, this value is set to **true** to send the balances with events. You can change the value to **false** if you want to retain the older event format.

To configure whether ECE sends balance data on rated events:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Set the **populateCurrentLoanAmountsOnRef** attribute to either **true** or **false** as desired.

Accessing ECE Configuration MBeans

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

1. Log in to the driver machine.
2. Start the ECE charging servers (if they are not started).
3. Connect to the ECE charging server node enabled for JMX management.
   This is the charging server node set to **start CohMgt = true** in the *ECE_home***/config/eceTopology.conf** file, where *ECE_home* is the directory in which ECE is installed.
4. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
5. In the editor's MBean hierarchy, find the ECE configuration MBeans.

Copyright © 2019, 2026, Oracle and/or its affiliates.

9

Generating POIDs for Rated Events

You can configure Oracle Communications Elastic Charging Engine (ECE) to generate POIDs for events that are created in ECE.

Topics in this document:

- About Generating POIDs in ECE
- Configuring ECE to Generate POIDs for Prepaid Events

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 9-1 for the description of each entry in the POID.

Table 9-1 POID Entries in ECE

Entry	Description
event_type	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.
date	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 "Changing Time and Date to Test ECE".

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 9-1 (Cont.) POID Entries in ECE

Entry	Description
cluster_id	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.
BRM_schema_id	A unique 6-bit number assigned to the BRM schema. The *BRM_schema_id* is limited to 0 to 31.
unique_id	A unique 34-bit number assigned to each POID.

You can configure multiple instances of Rated Event Formatter for uninterrupted POID allocation. 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 RE Formatter" in *Loading Rated Events*.

   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:

> **Note**
>
> In multischema systems, perform this task first on the primary BRM installation machine and then on the secondary BRM installation machines.

Copyright © 2019, 2026, Oracle and/or its affiliates.

1. Open the *BRM_home*/**sys/dm_oracle/pin.conf** file in a text editor.
2. Set the **prepaid_partition_transition_mode** entry to **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.

   ```
   - dm prepaid_partition_transition_mode 1
   ```
3. Save and close the file.
4. Create an editable XML file from the **system** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```
5. Set the **PrepaidPartitionSet** parameter to an integer value between **2** and **7**. For example:

   ```
   <PrepaidPartitionSet>2</PrepaidPartitionSet>
   ```
6. Save the file as **bus_params_system.xml**.
7. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_system.xml
   ```
8. Stop and restart the CM.
9. Go to the *BRM_home*/**apps/partition_utils** directory.
10. Enable prepaid-event partitions by running the following command:

    ```
    partition_utils -o enable -t prepaid
    ```
11. 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.

    For more information on enabling and adding partitions, see "Partitioning and Managing BRM Database Tables" in *BRM System Administrator's Guide*.

Configuring Cluster ID

To configure the cluster ID for ECE clusters:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.clusters.***Cluster_Name*, where *Cluster_Name* is the name of the ECE cluster that you are configuring.
4. Expand **Attributes**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. 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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.brmCdrPlugins.***Instance_Name*, where *Instance_Name* is the name of the BrmCdrPluginDirect Plug-in instance you are configuring.
4. Expand **Attributes**.
5. Set the **prepaidPartitionSet** attribute to the value that you specified in the **PrepaidPartitionSet** business parameter.

> **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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part III

Managing ECE Notifications

This part provides information about generating notifications for your customers or external systems from Oracle Communications Elastic Charging Engine (ECE). It contains the following chapters:

- Configuring Notifications in ECE
- Configuring Subscriber Preferences

Copyright © 2019, 2026, Oracle and/or its affiliates.

10

Configuring Notifications in ECE

You can configure Oracle Communications Elastic Charging Engine (ECE) to publish in-session notifications to subscribers and system notifications to external applications.

Topics in this document:

- About ECE Notifications
- Enabling ECE to Publish Notifications to External Applications
- Enabling Specific Notification Types
- Replicating Notifications to Kafka Partitions Dedicated to External Applications
- Replicating Notifications to Kafka Topics Dedicated to External Applications
- Publishing Asynchronous Notifications to a Separate Kafka Topic
- Publishing Asynchronous Notifications to Multiple External Topics
- Enabling In-Session Group Notifications in ECE
- Including Rollover Balances in Notifications
- Enriching Notifications Using ECE Extensions
- About Configuring BRM Gateway to Process ECE Notifications
- Modifying JMS Credentials for Publishing External Notifications

About ECE Notifications

ECE supports these types of notifications:

- **In-session notifications for your subscribers**: These notifications are sent to customers during online charging. For example, notifying customers about their reserved balance amounts.

  You can configure in-session notifications for individual subscribers by using subscriber preferences. See "Configuring Subscriber Preferences".

- **Notifications for external applications**: These notifications contain information that external applications need. For example:

  – The network mediation system can use data from the external notification in conjunction with customer policy data to implement network policy control.

    ECE sends notifications for external applications to the ECE Notification queue. You must configure your external application to retrieve and process notifications from the ECE Notification queue.

  – Oracle Communications Billing and Revenue Management (BRM) can use data in the external notification to run billing for a specific customer.

    ECE sends BRM notifications to the ECE Notification queue. You can configure BRM to retrieve notifications from the queue and send them to the BRM Server for processing.

All notifications are disabled by default. You can configure ECE to do the following:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Publish in-session notifications to your subscribers
- Publish notifications to external applications, such as BRM
- Publish specific notification types to a dedicated Kafka topic
- Specify the type of notifications that are supported
- Send notifications to everyone in a sharing group when one of its members triggers a notification
- Configure BRM Gateway to retrieve notifications from the ECE Notification queue

Enabling ECE to Publish Notifications to External Applications

To enable ECE to publish notifications to external applications such as BRM, do the following:

1. Open the *ECE_home***/config/charging-cache-config.xml** file.
2. Under the ServiceContext module, set **cache-store** to **oracle.communication.brm.charging.notification.internal.coherence.AsynchronousNotificationPublisher**:

```
<init-param>
  <param-name>cache-store</param-name>
    <param-
value>oracle.communication.brm.charging.notification.internal.coherence.AsynchronousNotificationPublisher</param-value>
```

3. Save the file.

Enabling Specific Notification Types

You can enable ECE to generate notifications for specific event types, such as exceeding a credit limit.

You specify whether a specific notification type is enabled and whether it can be sent to an external application, a subscriber, or both by setting its attribute to one of the values in Table 10-1.

Table 10-1 Notification Attribute Values

Attribute Value	Description
NONE	This notification type is disabled.
ASYNCHRONOUS	Send asynchronous notifications to external applications.
PIGGYBACK	Send in-session notifications to subscribers through the usage response message.
ASYNC_PIGGYBACK	Send both asynchronous notifications to external applications, and in-session notifications to subscribers through the usage response message.

To enable ECE to generate specific types of notifications:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

4. Expand **Attributes**.
5. Specify which type of notifications can be sent to your customers, external applications, or both by setting the attributes in Table 10-2 to **NONE**, **ASYNCHRONOUS**, **PIGGYBACK**, or **ASYNC_PIGGYBACK**.

> **Note**
>
> An asterisk (*****) appears next to the default attribute value.

Table 10-2 Notification Types

Attribute Name	Description	Supported Values
creditCeilingBreachNotificationMode	Sends notifications when a customer's balance has breached a credit ceiling.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
creditFloorBreachNotificationMode	Sends notifications when a customer's balance has breached a credit floor.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
thresholdBreachNotificationMode	Sends notifications when a customer's balance has breached a credit threshold. **Note:** During reverse rating, ECE does not generate threshold notifications unless the operation type is DEBIT_UNIT or DEBIT_AMOUNT.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
topUpNotificationMode	Sends notifications when a customer's balance is topped up.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
rarNotificationMode	Sends notifications when an ongoing session requires a reauthorization request. See "Enabling Server-Initiated Reauthorization Requests".	**NONE*** **ASYNCHRONOUS**
externalTopUpNotificationMode	Sends notifications when a customer's balance is topped up through an external application.	**NONE*** **ASYNCHRONOUS**
billingNotificationMode	Sends notifications 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.	**NONE*** **ASYNCHRONOUS**
adviceOfChargeNotificationMode	Sends notifications to support the Advice of Charge (AoC) supplementary service. Notifications are sent at the beginning and end of a session.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
replenishPoidIdNotificationMode	Sends notifications when ECE needs to obtain POID IDs from BRM.	**NONE** **ASYNCHRONOUS***
lifeCycleTransitionNotificationMode	Sends notifications when a customer's life-cycle state has changed.	**NONE*** **ASYNCHRONOUS**

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 10-2 (Cont.) Notification Types

Attribute Name	Description	Supported Values
firstUsageValidityInitNotificationMode	Sends notifications to synchronize the validity of first usage balance elements from ECE to external applications.	**NONE*** **ASYNCHRONOUS**
offeringUsageValidityInitNotificationMode	Sends notifications to synchronize the validity of offer usage balance elements from ECE to external applications.	**NONE** **ASYNCHRONOUS***
spendingLimitNotificationMode	Sends notifications when a threshold for a policy-driven charging rule has been reached.	**NONE*** **ASYNCHRONOUS**
aggregatedSpendingLimitNotificationMode	Sends aggregated notifications when a threshold for a policy-driven charging rule has been reached.	**NONE*** **ASYNCHRONOUS**
subscriberPreferenceUpdateNotificationMode	Sends notifications when a subscriber's notifications are updated.	**NONE*** **ASYNCHRONOUS**
rerateJobCreateNotificationMode	Sends notifications when a rerate job is created.	**NONE** **ASYNCHRONOUS***
customEventNotificationMode	Sends notifications when a custom event occurs.	**NONE*** **ASYNCHRONOUS**
enrichBalanceQueryResponseMode	Enriches the ECE balance query response with the subscriber's preferences.	**NONE*** **PIGGYBACK**
loanGrantNotificationMode	Sends notifications when a customer is granted a loan.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
automaticTopUpTriggerNotificationMode	Sends notifications when a customer's balance requires an automatic top up.	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**
customBrmOpCodeEventNotificationMode	Sends notifications when a custom event requires a call to a BRM opcode.	**NONE*** **ASYNCHRONOUS**
renewalGrantActivationNotificationMode	Sends notifications when an automatically renewed allowance is renewed or fully depleted. See "Enabling Notifications for Automatically Renewed Granted Allowances".	**NONE*** **ASYNCHRONOUS** **PIGGYBACK** **ASYNC_PIGGYBACK**

Replicating Notifications to Kafka Partitions Dedicated to External Applications

> **Note**
>
> This functionality applies to ECE systems using the partition-based Kafka architecture only. (That is, the **kafkaCustomPartitionAssignorEnabled** ECE MBean attribute is set to **false**).

Copyright © 2019, 2026, Oracle and/or its affiliates.

You can configure ECE to replicate one or more asynchronous notification types and publish them to multiple partitions in the ECE notification topics. This allows you to send the same notification to numerous external applications, with each application consuming notifications from a dedicated partition. For example, ECE could replicate a threshold-breach notification and then:

- Publish the original notification to Partition 1, which is connected to BRM
- Publish the replicated notification to Partition 2, which is connected to Oracle Communications Convergent Charging Controller

> **Note**
>
> For Kafka partition-based routing, every partition ID specified by **kafkaPartition** or **kafkaPartitionForInternalExternal** must be unique across the ECE deployment. Do not reuse a partition ID that is assigned to a Diameter Gateway instance, HTTP Gateway replica, BRM Gateway instance, or another internal/external notification consumer.

You configure an ECE on-premises installation to replicate one or more asynchronous notification types and publish them to multiple partitions by setting the **eventListForInternalExternalPublish** attribute in your **charging-settings.xml** file and then restarting your ECE system. For an ECE cloud native deployment, you set the **eventListForInternalExternalPublish** attribute in your **override-values.yaml** file for **oc-cn-ece-helm-chart** and then run the **helm upgrade** command.

To configure ECE to publish a notification to multiple partitions in the ECE notification topic without restarting ECE:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.
4. Expand **Attributes**.
5. In the **eventListForInternalExternalPublish** attribute, enter the list of notification types to replicate and publish to multiple partitions. Separate multiple notification types with a comma. The default is OFFERING_VALIDITY_INITIALIZATION_EVENT and EXTERNAL_TOP_UP_NOTIFICATION_EVENT.

   Table 10-3 lists the valid notification types.

Table 10-3 Notification Types

Notification Type	Description
SPENDING_LIMIT_NOTIFICATION_EVENT	Generated when a threshold for a policy-driven charging rule has been reached.
SUBSCRIBER_PROFILE_UPDATE_EVENT	Generated when a subscriber preference has been created, modified, and deleted.
CREDIT_FLOOR_BREACH_EVENT	Generated when a customer's balance has breached a credit floor.
CREDIT_CEILING_BREACH_EVENT	Generated when a customer's balance has breached a credit ceiling.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 10-3 (Cont.) Notification Types

Notification Type	Description
ADVICE_OF_CHARGE_EVENT	Generated when an Advice of Charge (AoC) event occurs. Notifications are sent at the beginning and end of a session.
THRESHOLD_BREACH_EVENT	Generated when a customer's balance has breached a credit threshold.
AGGREGATED_SPENDING_LIMIT_NOTIFICATION_EVENT	Aggregated notifications are generated when a threshold for a policy-driven charging rule has been reached.
RAR_NOTIFICATION_EVENT	Generated when an ongoing session requires a reauthorization request.
CUSTOM_EVENT_TYPE	Generated when a custom event occurs.
BILLING_NOTIFICATION_EVENT	Generated 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.
REPLENISH_POID_ID_EVENT	Generated when ECE needs to obtain POID IDs from BRM.
EXTERNAL_TOP_UP_NOTIFICATION_EVENT	Generated when a customer's balance is topped up through an external application.
LIFECYCLE_TRANSISTION_EVENT	Generated when a service's life cycle transitions to a new state.
FIRST_USAGE_VALIDITY_INIT_EVENT	Generated to synchronize validity of first usage balance elements from ECE to external applications.
RERATE_CREATE_JOB_EVENT	Generated when a rerate job is created.
OFFERING_VALIDITY_INITIALIZATION_EVENT	Generated to synchronize the validity of offer usage balance elements from ECE to external applications.
AUTOMATIC_TOP_UP_TRIGGER_NOTIFICATION_EVENT	Generated when a customer's balance requires an automatic top-up.

Replicating Notifications to Kafka Topics Dedicated to External Applications

> **Note**
>
> This functionality applies to ECE systems using the topic-based Kafka architecture only. (That is, the **kafkaCustomPartitionAssignorEnabled** ECE configuration MBean is set to **true**).

You can configure ECE to replicate one or more asynchronous notification types and publish them to multiple topics. This allows you to send the same notification to numerous external applications, with each application consuming notifications from a dedicated topic. For example, ECE could replicate a threshold-breach notification and then:

- Publish the original notification to topic 1, which is connected to BRM
- Publish the replicated notification to topic 2, which is connected to Oracle Communications Convergent Charging Controller

Copyright © 2019, 2026, Oracle and/or its affiliates.

ECE creates each notification topic using this naming convention:

Notify.*clusterName.consumerGroup*[.*notificationName*][.*networkClientId*][.*eceGatewayId*]

where:

- *clusterName* is the string that represents the site name from your **charging-settings.xml** file. The allowable values are BRM1, BRM2, ECE1, ECE2, and so on.
- *consumerGroup* is the name of the consumer group. The possible values are **DiamNetwork**, **RadNetwork**, **ChfNetwork**, **Brm**, **Ext**, **Suspense**, and **Overage**.
- *notificationName* is the notification event type. This is included for **DiamNetwork**, **RadNetwork**, **ChfNetwork**, **Brm**, and **Ext** consumer groups.
- *networkClientId* is used to differentiate the CHF Network (for the HTTP Gateway) from the Diameter Client (for the Diameter Gateway). This is included for **DiamNetwork** and **ChfNetwork** consumer groups.
- *eceGatewayId* is the value taken from the Diameter Gateway ID. This is included for **DiamNetwork** consumer groups.

The following shows sample notification topic names:

```
Notify.Ece1.DiamNetwork.RarNotificationEvent.pcrf1.oracle.com.dgw-11
Notify.Ece1.ChfNetwork.SpendingLimitNotificationEvent.ChfNetwork001
Notify.Ece2.RadNetwork.UserDisconnectNotificationEvent
Notify.Ece2.Brm.LifecycleTransitionNotification
Notify.Ece1.Ext.CreditCeilingBreachNotification
Notify.Ece1.Suspense
Notify.Ece2.Overage
```

To configure ECE to replicate one or more asynchronous notification types and publish them to multiple topics, follow the configuration instructions in "Replicating Notifications to Kafka Partitions Dedicated to External Applications".

Publishing Asynchronous Notifications to a Separate Kafka Topic

> **Note**
>
> - This functionality applies to ECE systems using the partition-based Kafka architecture only. (That is, the **kafkaCustomPartitionAssignorEnabled** ECE configuration MBean is set to **false**).
> - ECE always publishes notification events to the internal ECE notification topics first. ECE publishes to the External Notification topic only after the internal publish succeeds. Therefore, internal Kafka topics must exist and be operational even if no notifications are configured for internal consumption.

ECE publishes asynchronous notifications for external applications to the ECE notification topics. You can configure ECE to also publish one or more asynchronous notifications to multiple partitions in a separate Kafka topic (called the ECE External Notification topic). When the ECE External Notification topic is enabled, ECE:

- Publishes notifications to the internal ECE notification topics as well as the ECE External Notification topic.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Routes notifications to partitions in the ECE External Notification topic using the customer ID as a partition key. This means each notification type has a dedicated partition for a given customer.

Having separate partitions allows you to manage notifications more effectively and replicate the external partitions across sites to increase resiliency.

You configure an ECE on-premises installation to publish asynchronous notifications to a separate Kafka topic using the following entries in your **charging-settings.xml** file and then restarting your ECE system. For an ECE cloud native deployment, you set these entries in your **override-values.yaml** file for **oc-cn-ece-helm-chart** and then run the **helm upgrade** command:

- **externalTopicEnabled**: Set this to **true**. The default is **false**.
- **externalTopicName**: Set this to the name of the ECE External Notification topic. The default is **EceExtNotifications**.
- **externalPartitions**: Set this to the total number of Kafka partitions in the external topic. The default is **15**.
- **eventListForECEExternalTopic**: Set this to one or more notification types listed in Table 10-3, separated by commas. ECE publishes these notification types, including service-event notification types, to the ECE External Notification topic.

  This list controls which notification types ECE publishes to the ECE External Notification topic. It does not disable publishing to the internal ECE notification topics.

To configure these entries without restarting ECE:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.kafkaConfigurations.BRM**.
4. Expand **Attributes**.
5. Enable the ECE External Notification topic and specify its configuration values:
   - **externalTopicEnabled**: Set this to **true**. This specifies that the ECE External Notification topic is created and in use.
   - **externalTopicName**: Set this to the name of the ECE External Notification topic.
   - **externalPartitions**: Set this to the total number of Kafka partitions in the ECE External Notification topic.
6. Under the **ECE Configuration** node, expand **charging.notification**.
7. Expand **Attributes**.
8. In the **eventListForECEExternalTopic** attribute, enter the list of notification types to publish to the ECE External Notification topic. Separate multiple notification types with a comma. See Table 10-3 for the list of valid notification types.
9. Perform a rolling upgrade of the ecs server.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Publishing Asynchronous Notifications to Multiple External Topics

> **Note**
>
> - This functionality applies to ECE systems using the topic-based Kafka architecture only. (That is, the **kafkaCustomPartitionAssignorEnabled** ECE configuration MBean is set to **true**).
> - ECE always publishes notification events to the internal ECE notification topics first. ECE publishes to the External Notification topic only after the internal publish succeeds. Therefore, internal Kafka topics must exist and be operational even if no notifications are configured for internal consumption.

You can configure ECE to publish one or more asynchronous notifications for external applications to multiple external topics, with one topic per notification type. This means each notification type has a dedicated partition for a target external application.

ECE creates the notification topics using this naming convention:

`Notify.`*`clusterName.consumerGroup`*`[.`*`notificationName`*`]`

where:

- *clusterName* is the string that represents the site name from your **charging-settings.xml** file. The allowable values are BRM1, BRM2, ECE1, ECE2, and so on.
- *consumerGroup* is the name of the consumer group. The possible values are **Ext**.
- *notificationName* is the notification event type. This is included for the **Ext** consumer group.

The following shows sample notification topic names:

```
Notify.Ece1.Ext.CreditCeilingBreachNotification
```

To configure ECE to publish one or more asynchronous notifications for external applications to multiple external topics, follow the configuration instructions in "Publishing Asynchronous Notifications to a Separate Kafka Topic".

Enabling In-Session Group Notifications in ECE

You can configure ECE to send notifications to everyone in a sharing group when one of its members triggers a notification. For example, when a member breaches a credit limit or credit threshold, a notification is sent to all members in the sharing group.

When group notifications are disabled, ECE sends notifications only to the member that triggered the notification.

> **Note**
>
> ECE selects sharing-group notification recipients based on the group membership that was effective when the notification event occurred. ECE does not notify members added after the event. ECE can notify members removed after the event if they were group members when the event occurred.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To enable group notifications:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **groupNotificationEnabled** attribute to **True**.
6. (For **firstUsageValidityInitNotificationMode** notification types only) Set **replicateFirstUsageValidityInitEvent** to **true**.

> **Note**
>
> For **firstUsageValidityInitNotificationMode** notification types, use the **replicateFirstUsageValidityInitEvent** attribute to control whether ECE replicates first-usage-validity initialization notifications. When this attribute is set to **true**, ECE replicates the notification when replication is applicable. When it is set to **false**, ECE does not replicate the notification.

7. Configure the **NotificationEnabledAgreements** subscriber preference:
   a. Add a string for **NotificationEnabledAgreements** in the *BRM_home***/sys/msgs/active_mediation/active_mediation.en_US** file. For example:

```
STR
  ID = string_id;
  Version = 1;
  STRING = "NotificationEnabledAgreements";
END
```

   where *string_id* is a unique numerical ID for the string. See "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide* for information about adding strings.

   b. Add **NotificationEnabledAgreements** to the *BRM_home***/sys/data/config/config_subscriber_preferences_map.xml** file. For example:

```
<SUBSCRIBER_PREFERENCES elem="preference_id">
       <NAME>NotificationEnabledAgreements</NAME>
       <SUBSCRIBER_PREFERENCE_ID>preference_id</SUBSCRIBER_PREFERENCE_ID>
       <STRING_ID>string_id</STRING_ID>
       <STR_VERSION>1</STR_VERSION>
       <DEFAULT></DEFAULT>
       <TYPE>1</TYPE>
 </SUBSCRIBER_PREFERENCES>
```

   where:

   - *preference_id* is a unique numerical ID for the preference.
   - *string_id* is the same value you used for **ID** in the **active_mediation.en_US** file.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> Leave the **<DEFAULT>** element blank to ensure that only subscribers who opt in will receive notifications.

When group notifications are enabled, you set subscriber preferences at the profile level to enable notifications for the individual subscribers who want to receive them. See "Configuring Group Notifications" for more information.

Including Rollover Balances in Notifications

Rollover data is included in the balance data synchronized from BRM to ECE using the PIN_FLD_ROLLOVER_DATA field. PIN_FLD_ROLLOVER_DATA is a 32-bit packed value. You read the bits from the most significant bit (MSB) to the least significant bit (LSB) as follows:

- Bits 31–30 (2 bits): Rollover flag
- Bits 29–26 (4 bits): Rollover unit
- Bits 25–16 (10 bits): Frequency
- Bits 15–0 (16 bits): Count

A count of **0** indicates that a rollover has not occurred yet.

> **Note**
>
> The specific meanings of the rollover flag and unit codes depend on your BRM and ECE configuration and product policies. Use your configuration to map these codes to business semantics.

You can customize notifications sent by ECE to include the PIN_FLD_ROLLOVER_DATA field using the ECE SDK. To do so, extend ECE at the post-charging extension point to call the **getRolloverData()** method.

Enriching Notifications Using ECE Extensions

Using the ECE SDK, you can customize the information included in notifications sent by ECE. Enriching the notification payload can help you provide subscribers with relevant data like their initial granted balance, current balance, session ID, and notification type.

To enrich notifications, extend ECE at the post-charging extension point and use the **SamplePostChargingExtension** program. See "Post-Charging Extension - Enriching Notifications" for more information.

About Configuring BRM Gateway to Process ECE Notifications

ECE publishes notifications intended for external applications to the ECE notification queue. If the notifications are targeted for BRM, BRM Gateway retrieves the notifications from the queue and calls the appropriate BRM opcode. Table 10-4 lists the BRM opcode called for each ECE notification type.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 10-4 ECE Notification to BRM Opcode Mapping

ECE Notification Type	BRM Opcode Called	Description
AUTOMATIC_TOP_UP_TRIGGER_NOTIFICATION_EVENT	PCM_OP_PYMT_TOPUP	Generated for top ups made directly via ECE.
BILLING_NOTIFICATION_EVENT	PCM_OP_BILL_MAKE_BILL	Generated when an event occurs after a customer's accounting cycle has terminated. Billing is triggered. Relevant charges and changes in state are synchronized to ECE through the EM Gateway when the opcode completes.
CUSTOM_BRM_OP_CODE_EVENT_NOTIFICATION_EVENT	The opcode called is based on the opcode XML tag.	Permits custom opcodes to be called through the BRM Gateway.
EXTERNAL_TOP_UP_NOTIFICATION_EVENT	PCM_OP_BILL_DEBIT	Synchronizes external top ups, received directly in ECE, for persistence within BRM.
FIRST_USAGE_VALIDITY_INITIALIZATION_EVENT	PCM_OP_BAL_CHANGE_VALIDITY	Initializes BRM-side balance validities based on first-usage alignment configuration.
LIFECYCLE_TRANSITION_NOTIFICATION_EVENT	PCM_OP_CUST_UPDATE_SERVICES	Life cycle transitions are invoked (for things like pre-active to active SIMs), which trigger service status changes on BRM. These changes may indirectly trigger other configured activity.
LOAN_GRANT_EVENT	PCM_OP_LOAN_NOTIFY_THRESHOLD	For triggering loan assessment in BRM (usually a low-balance automatic loan that may permit a session to continue rather than exhaust a customer's credit).
OFFER_VALIDITY_INITIALIZATION_EVENT	PCM_OP_SUBSCRIPTION_SET_VALIDITY	Initializes BRM-side subscription validity based on first-usage alignment configuration.
REPLENISH_POID_ID_NOTIFICATION_EVENT	PCM_OP_GET_POID_IDS	POIDs (used for **/item** assignment in ECE) are preemptively replenished via this opcode call when about 20% of the initial POID cache is remaining.
RERATE_CREATE_JOB_EVENT	PCM_OP_RERATE_INSERT_RERATE_REQUEST	Triggers rerating requests in BRM.
SUBSCRIPTION_CYCLE_FORWARD_EVENT	PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Permits triggering of cycle-forward events in BRM whenever a balance (that is considered recurring) is expiring as part of an ongoing ECE session.

Configuring BRM Gateway to Process ECE Notifications

> **Note**
>
> The following steps apply to ECE systems using the partition-based Kafka architecture only. (That is, when the **kafkaCustomPartitionAssignorEnabled** attribute is set to **false**.)

To configure BRM Gateway to process ECE notifications, you must connect BRM Gateway to both BRM and the ECE notification topics or queues:

1. When you install ECE:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Add details for connecting the BRM Gateway to the BRM Connection Manager (CM).
- If you are using Oracle WebLogic Server for notification handling, specify to create WebLogic queues and enter the details for your ECE Notification queue and Suspense queue.
- If you are using Apache Kafka for notification handling, specify to create Kafka topics and enter the details for your ECE notification topics and Suspense topic.

> **Note**
>
> Systems that support 5G networks must use Apache Kafka for notification handling.

For more information, see "Installing an ECE Integrated System" in *Elastic Charging Engine Installation Guide*.

2. During the ECE postinstallation process, do this:
   - If you are using Oracle WebLogic for notification handling, run the **post_Install.pl** script to create your ECE Notification queue, Suspense queue, and Acknowledgment queue. See "Creating WebLogic JMS Queues for BRM" in *Elastic Charging Engine Installation Guide*.
   - If you are using Apache Kafka for notification handling, run the **kafka_post_install.sh** script to create your ECE notification topics and Suspense topic. Then, run the **post_Install.pl** script and choose to create only the Acknowledgment queue. See "Creating Kafka Topics for ECE" and "Creating WebLogic JMS Queues for BRM" in *Elastic Charging Engine Installation Guide*.
3. Configure your BRM Gateway instances:
   - If you want to configure a single BRM Gateway instance, see "Configuring the BRM Gateway".
   - If you want to configure multiple BRM Gateway instances, see "Configuring Multiple BRM Gateway for Multi-Schema Deployments".
4. Configure the BRM Gateway queues or topics:
   - If you are using WebLogic queues, see "Configuring WebLogic Queues for BRM Gateway".
   - If you are using Kafka topics, see "Connecting BRM Gateway to Kafka Topics and BRM".
   - If you are using queues from an external application, see "Considerations for Using a Non-WebLogic Server JMS Provider".

Configuring the BRM Gateway

You must configure a BRM Gateway instance for each schema in your BRM database. To configure a single BRM Gateway instance:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.brmGatewayConfiguration**.
4. Expand **Attributes**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. Use the following attributes to configure BRM Gateway:
   - **name**: The name of this BRM Gateway instance.
   - **schemaNumber**: The number of the database schema BRM Gateway connects to in the BRM database.
   - **kafkaPartition**: The dedicated Kafka partition ID consumed by this BRM Gateway instance for Kafka partition-based notification routing. The partition ID must be unique across gateways and must not overlap with partition IDs assigned to Diameter Gateway instances, HTTP Gateway replicas, or partition IDs configured by **kafkaPartitionForInternalExternal**. For Kafka topic-based routing, this attribute is ignored.
   - **emptyQueueThreadSleepInterval**: Specifies the interval, in milliseconds, in which the gateway thread pings the ECE Notification queue to check if a connection is available when the queue is empty.
   - **jmsReceiveSleepInterval**: Specifies the interval, in milliseconds, where the BRM Gateway will wait and continue if there are no messages from the Kafka or WebLogic queue.
   - **brmResponseTimeOutInterval**: Specifies the interval, in milliseconds, where BRM opcode service will wait for all responses from CM.
   - **gatewaySleepInterval**: Specifies the number of milliseconds between connection retry attempts for the Notification queue.
   - **jmsBatchSize**: Specifies the maximum number of failed update requests BRM Gateway can retrieve from the Suspense queue in a batch.
   - **jmsReceiveTimeout**: Specifies the timeout in milliseconds for Kafka/JMS to receive messages from the queue.
   - **brmWorkerThreads**: Specifies the number of gateway threads, which, as part of the charging settings configuration, is 10 but should not be less than 1.
   - **brmSchedulerThreadInitialDelay**: Specifies the initial delay in milliseconds before the BRM opcode service task is executed.
   - **brmSchedulerThreadDelayPeriod**: Specify the interval, in milliseconds, at which the BRM opcode service starts executing continuously within the given period.
   - **brmSuspenseQueuePeriod**: Specifies the maximum amount of time, in milliseconds, BRM Gateway has to retrieve a batch from the Suspense queue.
   - **connectionRetryCount**: Specifies the number of times BRM Gateway can retry a connection to the BRM CM after it fails.
   - **connectionRetryInterval**: The amount of time, in milliseconds, between reconnection attempts to the CM.

Configuring Multiple BRM Gateway for Multi-Schema Deployments

You must configure a BRM Gateway instance for each schema in your BRM database, i.e., you cannot have a single BRM Gateway in multi-schema. For example, in a system with three schemas, you must have BRM Gateway 1 connected to Schema 1, BRM Gateway 2 connected to Schema 2**,** and BRM Gateway 3 connected to Schema 3.

In on-premises systems, you can optionally define more than one BRM Gateway instance for each schema. In this case, ECE automatically makes one of the BRM Gateways active while the other instances are inactive. One of the inactive instances comes online only if the active instance goes down. For example, assume BRM Gateway 1 and 2 are connected to Schema

Copyright © 2019, 2026, Oracle and/or its affiliates.

1. When BRM Gateway 1 is active, it remains active and continues processing ECE notifications until the instance goes down. Then, BRM Gateway 2 becomes active and starts processing notifications.

To configure multiple BRM Gateway instances:

1. Open the *ECE_home*/**config/management/charging-settings.xml** file.
2. Delete the existing **charging.brmGatewayConfiguration** section:

```
<brmGatewayConfiguration
    config-
class="oracle.communication.brm.charging.appconfiguration.beans.connection.BRMGateway
Configuration"
    name="brmGatewayConfiguration"
    emptyQueueThreadSleepInterval="50"
    jmsReceiveSleepInterval="100"
    brmResponseTimeOutInterval="600000"
    gatewaySleepInterval="2000"
    jmsBatchSize="10"
    jmsReceiveTimeout="2000"
    brmWorkerThreads="10"
    brmSchedulerThreadInitialDelay="10"
    brmSchedulerThreadDelayPeriod="3"
    brmSuspenseQueuePeriod="1800000"
    connectionRetryCount="10"
    connectionRetryInterval="10000">
</brmGatewayConfiguration>
```

3. Copy the following **charging.brmGatewayConfigurations** section into the file:

```
<brmGatewayConfigurations
    config-
class="oracle.communication.brm.charging.appconfiguration.beans.connection.BRMGateway
Configurations">
    <brmGatewayConfigurationList config-class="java.util.ArrayList">
        <brmGatewayConfiguration
            config-
class="oracle.communication.brm.charging.appconfiguration.beans.connection.BRMGateway
Configuration"
            name="brmGatewayN"
            clusterName="@CLUSTER_NAME@"
            emptyQueueThreadSleepInterval="50"
            jmsReceiveSleepInterval="100"
            brmResponseTimeOutInterval="600000"
            gatewaySleepInterval="2000"
            jmsBatchSize="10"
            jmsReceiveTimeout="2000"
            brmWorkerThreads="10"
            brmSchedulerThreadInitialDelay="10"
            brmSchedulerThreadDelayPeriod="3"
            brmSuspenseQueuePeriod="1800000"
            connectionRetryCount="10"
            connectionRetryInterval="10000"
            schemaNumber="N"/>
</brmGatewayConfigurationList>
```

4. Under **brmGatewayConfigurationList**, add this section for each additional BRM Gateway instance that you want to create:

```
<brmGatewayConfiguration
    config-
class="oracle.communication.brm.charging.appconfiguration.beans.connection.BRMGateway
Configuration"
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
    name="brmGateway"
    clusterName="@CLUSTER_NAME@"
    emptyQueueThreadSleepInterval="50"
    jmsReceiveSleepInterval="100"
    brmResponseTimeOutInterval="600000"
    gatewaySleepInterval="2000"
    jmsBatchSize="10"
    jmsReceiveTimeout="2000"
    brmWorkerThreads="10"
    brmSchedulerThreadInitialDelay="10"
    brmSchedulerThreadDelayPeriod="3"
    brmSuspenseQueuePeriod="1800000"
    connectionRetryCount="10"
    connectionRetryInterval="10000"
    schemaNumber="1"/>
</brmGatewayConfigurationList>
```

5. Edit these properties for each BRM Gateway instance:
   - **name**: Set this to the name of your BRM Gateway instance in this format: **brmGateway***N*. For example, name the first instance brmGateway1, the second instance brmGateway2, and so on.
   - **schemaNumber**: Set this to the schema number that the BRM Gateway instance connects to in the BRM database. Enter **1** to connect to the first schema, **2** to enter the second schema, and so on.
6. Save and close the **charging-settings.xml** file.
7. Stop BRM Gateway:

   ```
   stop brmGateway
   ```
8. Open the *ECE_home***/config/eceTopology.conf** file.
9. Add a row to the file for each BRM Gateway instance.

   For example, to configure three BRM Gateway instances, add three rows in which the node name value is brmGateway1, brmGateway2, and brmGateway3, and the role value for all three rows is brmGateway:

   ```
   #node-name          |role                |host name (no spaces!) |host ip |JMX port
   |start CohMgt |JVM Tuning File
   brmGateway1         |brmGateway          |localhost              |        |9994
   |false        |defaultTuningProfile
   brmGateway2         |brmGateway          |localhost              |        |9994
   |false        |defaultTuningProfile
   brmGateway3         |brmGateway          |localhost              |        |9994
   |false        |defaultTuningProfile
   ```
10. Save and close the **eceTopology.conf** file.
11. From the *ECE_home***/bin** directory, start Elastic Charging Controller (ECC):

    ```
    ./ecc
    ```
12. Start all BRM Gateway instances:

    ```
    start brmGateway
    ```

Connecting BRM Gateway to Kafka Topics and BRM

To connect BRM Gateway to the ECE Notification Kafka topic and BRM:

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **kafkaEnabledForNotifications** property to **true**.
6. Expand **charging.kafkaConfigurations**.
7. Expand **Attributes**.
8. Specify the Kafka configuration values for the attributes in Table 10-5.

   Verify that the name, hostname, and topic names that you provide match the settings entered when installing ECE.

Table 10-5 Settings for Connecting BRM Gateway to Kafka Topics

Property Name	Description
name	The name of your ECE cluster.
hostname	The host name and port number of the machine in which Apache Kafka is up and running. If it contains multiple Kafka brokers, create a comma-separated list.
topicName	The name of the Kafka topic in which BRM Gateway retrieves notifications from ECE.
partitions	The total number of partitions in the ECE notification topic for Kafka partition-based routing. For the calculation, examples, and validation rule, see "Determining the Number of Partitions Required" and "Creating Kafka Topics for ECE Integration." For gateway partition assignment and uniqueness requirements, see "Configuring Kafka Partition Assignments for Gateways"
kafkaBRMReconnectionInterval	The amount of time, in milliseconds, BRM Gateway waits before attempting to reconnect to the Kafka topic.
kafkaBRMReconnectionMax	The maximum amount of time, in milliseconds, BRM Gateway waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaBRMReconnectionInterval** will increase exponentially for each consecutive connection failure up to this maximum.

9. Expand **charging.connectionConfigurations.brmConnection**.
10. Expand **Attributes**.
11. Specify the configuration values for connecting BRM Gateway to the BRM Connection Manager (CM):
    - **hostName**: Enter the host name of the server in which the CM is running. For example: abc01.example.com.
    - **loginName**: Enter the user name for logging in to BRM. The default is **root.0.0.0.1**.
    - **cmPort**: Enter the port number for the BRM CM.

Copyright © 2019, 2026, Oracle and/or its affiliates.

About BRM Gateway Error Handling

When BRM Gateway calls an opcode, it checks whether the opcode ran successfully. If it did not, the exception from the opcode is checked and, if the exception is one of the following, it retries the opcode call.

- ERR_TIMEOUT
- ERR_DEADLOCK
- ERR_STREAM_EOF
- ERR_IM_CONNECT_FAILED
- ERR_EM_CONNECT_FAILED
- ERR_NAP_CONNECT_FAILED

If the opcode continues to fail after all of the configured retry attempts, the notification is moved to the Suspense topic.

If the exception is due to a duplicate message (ERR_DUPLICATE), the opcode call is not retried and the message is not moved to the Suspense topic.

Configuring WebLogic Queues for BRM Gateway

To configure WebLogic queues for BRM Gateway:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations.brmConnection**.
4. Expand **Attributes**.
5. Use the following attributes to specify connection values to the BRM Connection Manager (CM):
   - **loginName**: Enter the user name for logging in to BRM. The default is **root.0.0.0.1**.
   - **hostName**: Enter the IP address or the host name of the computer on which BRM is configured.
   - **cmPort**: Enter the port number for the BRM CM.

Setting the Transaction Timeout for the connectionFactory

For the connectionFactory created within the WebLogic server for creating connections to the JMS topic, do the following:

Refer to the Oracle WebLogic Server documentation for information about setting up a JMS topic on WebLogic Server.

1. Log on to the WebLogic Server on which the JMS topic resides.
2. Click **Edit Tree**, then **Services**, and then **JMS Modules**.

   The Summary of JMS Modules page appears.
3. In the JMS Modules table, click the JMS topic used for ECE notifications.
4. Click **Connection Factories** in the tree in the left pane.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. In the JMS Connection factories table, click **NotificationFactory**.
6. Click the **Transaction** tab and set the **Transaction Timeout** field to **2147483647.**
7. Click **Save.**
8. Click the shopping cart at the top right, and then click **Commit Changes** to commit your changes.

Considerations for Using a Non-WebLogic Server JMS Provider

If you choose to use a JMS provider other than WebLogic Server for publishing ECE notification events, you must do the following on the driver machine:

- Copy the other JMS provider's client JARs to the *ECE_home***/lib** directory.
- Rename the other JMS provider JAR file to **wlthint3client.jar**.
- Update the *ECE_home***/config/JMSConfiguration.xml** file to specify the **InitialContextFactory** and protocol information of the other JMS provider.

Modifying JMS Credentials for Publishing External Notifications

When you install ECE, you can specify to publish notifications to a WebLogic JMS queue (named ECE Notification queue) as well as how to connect to the queue.

If you need to change the connection information after ECE is installed, you can edit the parameters in the **JMSConfiguration.xml** file.

To modify the JMS credentials in ECE, do the following:

1. Open the *ECE_home***/config/JMSConfiguration.xml** file.
2. Locate the **<MessagesConfigurations>** section.
3. Specify the values for the parameters in the **NotificationQueue** section:

> **Note**
>
> Do not change the value of the **JMSDestination name** parameter.

   a. For the **HostName** parameter, specify the host name of the WebLogic server on which the JMS topic resides.
   b. For the **Port** parameter, specify the port number on which the WebLogic server resides.
   c. For the **UserName** parameter, specify the user for logging in to the WebLogic server.

      This user must have write privileges on the JMS topic created.
   d. For the **Password** parameter, specify the password for logging in to the WebLogic server.

      When you install ECE, the password you enter is encrypted and stored in the KeyStore. If you change the password, you must first run a utility to encrypt the new password before entering it here.
   e. For the **ConnectionFactory** parameter, specify the connectionFactory created within the WebLogic server for creating connections to it.

Copyright © 2019, 2026, Oracle and/or its affiliates.

You must also configure settings in the WebLogic server for the connectionFactory. See "Setting the Transaction Timeout for the connectionFactory" for information.

f. For the **QueueName** parameter, specify the JMS topic.

   This is the JMS topic which holds the published external notification messages.

g. For the **Protocol** parameter, specify the wire protocol used. For the WebLogic server, set this to **t3://**.

4. Save and close the file.

Copyright © 2019, 2026, Oracle and/or its affiliates.

11

Configuring Subscriber Preferences

You can configure subscriber preferences such as how they want to receive notifications from the network in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- How BRM Manages Subscriber Preferences
- Tasks for Including Subscriber Preference Information in Notifications
- Collecting Subscriber Preferences Using Client Tools
- Granting CSR Access to Subscriber Preferences

How BRM Manages Subscriber Preferences

You can use BRM 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 by SMS.

By default, you can manage the following subscriber preferences in BRM:

- Preferred communication channel, such as IVR, SMS, or email
- Preferred communication language, such as English or French
- Number of days in advance that the subscriber wants to receive the notification
- Interval between successive notifications
- Timestamp of the last notification sent to the subscriber

BRM stores each subscriber's preferences in a subscriber profile repository. The **/config/subscriber_preferences_map** object stores the preference types that you track and their default values. BRM stores each subscriber's preferences in individual **/profile/subscriber_preferences** objects at the account and service levels.

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

Tasks for Including Subscriber Preference Information in Notifications

To set up ECE to include subscriber preference information in notifications:

1. Enable in-session notifications. See "Enabling Specific Notification Types".
2. Specify the subscriber preferences that can be collected along with their default values. See "Configuring Which Subscriber Preferences to Collect".
3. (Optional) Configure ECE to enrich external notifications with subscriber preference information. See "Configuring ECE to Enrich External Notifications with Subscriber Preference Information".

Copyright © 2019, 2026, Oracle and/or its affiliates.

4. (Optional) Configure ECE to send credit limit and threshold breach notifications to multiple members of a sharing group. See "Configuring Group Notifications".

Afterward, you can start collecting your subscribers' preferences through Billing Care or your custom client application. See "Collecting Subscriber Preferences Using Client Tools".

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 all or a subset of subscriber preferences:

- 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

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 preferences that you previously set in your BRM system.
2. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing 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:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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 version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
<NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
<PublicUserIdentities>
<PublicUserIdentity>6500000001</PublicUserIdentity>
</PublicUserIdentities>
<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>
```

Configuring Which Subscriber Preferences to Collect

You can configure which subscriber preferences your client applications display and collect. To configure these preferences, update the **config_subscriber_preferences_map.xml** file and use the **load_config** utility to load its contents into the database. The default **config_subscriber_preferences_map.xml** file includes sample subscriber preferences that you can add, modify, or remove.

To configure which subscriber preferences your client applications collect:

1. (Optional) Customize the subscriber preferences:
   a. Open the *BRM_home*/**sys/data/config/config_subscriber_preferences_map.xml** file.
   b. Modify or delete the existing subscriber preferences.
   c. For each custom subscriber preference, add a **<SUBSCRIBER_PREFERENCES>** array element and define the fields described in Table 11-1.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 11-1 Elements That Store Subscriber Preferences

Element	Description
<NAME>	The name of the preference.
<SUBSCRIBER_PREFERENCE_ID>	The ID associated with the preference.
<STRING_ID>	The string ID that Billing Care uses to display the localized preference name.
<STR_VERSION>	The string field version.
<DEFAULT>	The default value for the preference field.
<TYPE>	The preference can be assigned one of these value 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 the **<VALUES>** element in this table. • **4**: DECIMAL • **5**: TSTAMP (timestamp) • **6**: TIME (time in 24-hour format *HH:mm*) • **7**: MULTI SELECT ENUM (a fixed list that is displayed as a string with checkboxes or combo boxes. The list is stored as a string in CSV format.) • **262**: TIME RANGE (time range in 24-hour format: *HH:mm-HH:mm*. For example, 16:00-20:00) To provide a set of possible values, set **<TYPE>** to **3** and enter the values in **<VALUES>**.
<VALUES>	An array of possible values for the field. Use this element only for **ENUM** types.

For example, the following entries define a subscriber preference named **Include Current Date**. The possible values are **True** and **False**, with **False** as the default.

```
<ConfigObject>
   <DESCR>Subscriber Prefs Map</DESCR>
   <NAME>Subscriber Prefs Map</NAME>
   <SUBSCRIBER_PREFERENCES elem="x">
      <NAME>Include Current Date</NAME>
      <SUBSCRIBER_PREFERENCE_ID>10</SUBSCRIBER_PREFERENCE_ID>
      <STRING_ID>10</STRING_ID>
      <STR_VERSION>1</STR_VERSION>
      <DEFAULT>909</DEFAULT>
      <TYPE>3</TYPE>
         <VALUES elem="0">
            <VALUE>True</VALUE>
            <STRING_ID>908</STRING_ID>
            <STR_VERSION>998</STR_VERSION>
         </VALUES>
         <VALUES elem="1">
            <VALUE>False</VALUE>
            <STRING_ID>909</STRING_ID>
            <STR_VERSION>999</STR_VERSION>
         </VALUES>
   </SUBSCRIBER_PREFERENCES>
</ConfigObject>
```

d. Save the **config_subscriber_preferences_map.xml** file.

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Open the *BRM_home*/**apps/load_config/pin.conf** file in a text editor.
3. At the end of the file, add this entry:

   ```
   - load_config validation_module libLoadValidTCFAAA LoadValidTelcoAAA_init
   ```

4. Save the **pin.conf** file.
5. Run the **load_config** utility to load the **config_subscriber_preferences_map.xml** file into the database:

   ```
   load_config config_subscriber_preferences_map.xml
   ```

   > **Note**
   >
   > - The **load_config** utility requires a **pin.conf** configuration 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 more information, see "load_config" in *BRM Developer's Guide*.

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

To verify that the updated subscriber preference configuration was loaded, display the **/config/subscriber_preferences_map** object by using the Object Browser or the **robj** command with the **testnap** utility.

For more information about the **/config/subscriber_preferences_map** object, see *BRM Storable Class Reference*.

Configuring Group Notifications

You can configure ECE to send credit limit and threshold breach notifications to multiple members of a sharing group. By default, only the member who triggered the breach is notified.

To configure group notifications:

1. Confirm that group notifications are enabled as described in "Enabling In-Session Group Notifications in ECE".
2. For group owners, set the **ResourcesForSendingNotification** subscriber preference at the account level to a comma-separated list of resource balance element IDs that the owner wants to send notifications for.
3. For group members:
   a. Set **ResourcesForReceivingNotification** at the account level to a comma-separated list of resource balance element IDs that the member wants to receive notifications for. Values included in this list must also be listed in **ResourcesForSendingNotification** for the group owner.
   b. Set **NotificationEnabledAgreements** at the account or service level as a comma-separated list of sharing group names that the subscriber wants notifications for. For example:

      ```
      SharingAgreement12, Charge_Sharing14, DataSharing_143
      ```

      These sharing groups must contain the resources specified in **ResourcesForReceivingNotification**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

c. Set **OfflineNotificationEnabled** at the account or service level to **true** if the group member wants to receive notifications when they are offline for breaches caused by other group members. The setting at the service level overrides the setting at the account level.

> **Note**
>
> Because group owners can also be group members, an individual subscriber might have all of these preferences set in their subscriber profile.

Collecting Subscriber Preferences Using Client Tools

You can add subscriber preferences to your customers' accounts by using the following client tools:

- Billing Care. See "Viewing and Adding Subscriber Preferences" in *Billing Care Online Help*.
- A custom client application calling the Profiles REST endpoints in the Billing Care REST API. See *REST API References for Billing Care*.
- A custom client application calling the subscriber profile opcodes. See "Managing and Customizing Profiles" in *BRM Opcode Guide*.

These client tools use the configurations in the **/config/subscriber_preferences_map** object to dynamically list the preferences that a subscriber can configure. The client tool does not display the subscriber preferences pages and dialog boxes if this object is missing.

Granting CSR Access to Subscriber Preferences

By default, CSRs cannot access subscriber preferences in Billing Care. You grant permissions to view and update subscriber preferences in Billing Care by granting CSRs the **ProfileResource** authorization resource. See "About Billing Care Authorization Resources" in *BRM Security Guide*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part IV
Managing Charging Sessions

This part provides information about managing online and offline charging sessions in Oracle Communications Elastic Charging Engine (ECE). It contains the following chapters:

- Understanding Linear and Non-Linear Rating in ECE
- Managing Midsession-Rated Events
- Managing Online Charging Sessions
- Managing Session Start and End Times
- Managing Reservations for Online Sessions
- Managing Rounding and Consumption Rules
- Managing Granted Allowances
- Managing Charging During System Downtimes
- using-impact-based-candidate-selection.ditamap#GUID-ABAA5072-E9B9-4251-9F31-A9522EF7FA60

Copyright © 2019, 2026, Oracle and/or its affiliates.

12

Understanding Linear and Non-Linear Rating in ECE

Learn about linear and non-linear rating in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- How ECE Applies Linear Rating
- How Non-Linear Rating Manages Tariff Boundaries
- Configuring Linear or Non-Linear Rating

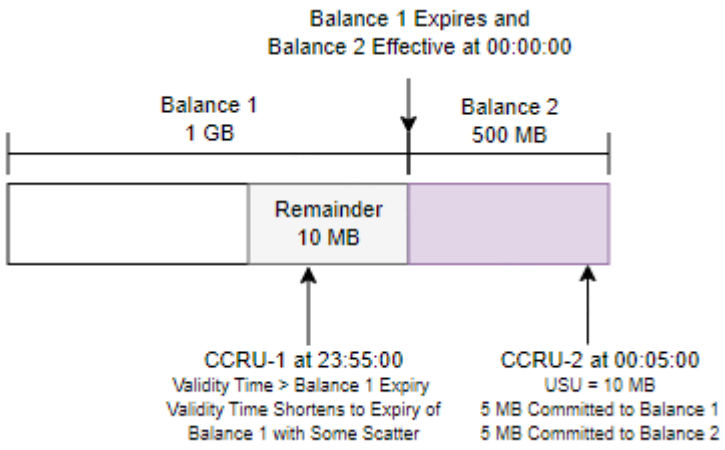
How ECE Applies Linear Rating

Linear rating, a concept widely used in the telecom industry, assumes that resource consumption occurs at a constant rate throughout the usage period. For example, linear rating can apply a rate of 1 MB per minute for data usage or 1 included minute per minute for voice usage. If a customer consumes 100 MB in 10 minutes, linear rating assumes a usage rate of 10 MB per minute.

However, linear rating can produce inaccurate results when a boundary condition occurs and consumption is not evenly distributed over time. For this reason, linear rating is best suited for voice and event usage with constant consumption rates.

Figure 12-1 shows a linear rating example. ECE grants usage at 23:55:00 with CCRU-1. At 00:05:00, the network reports 10 MB of usage in CCRU-2.

Figure 12-1 Linear Rating Example



In this example, the total usage is 10 MB in 10 minutes, so linear rating calculates a consumption rate of 1 MB per minute. Because Balance 1 expires at 00:00:00, linear rating distributes the usage as follows:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 5 MB is consumed between 23:55:00 and 23:59:59 and charged to Balance 1.
- 5 MB is consumed between 00:00:00 and 00:05:00 and charged to Balance 2.

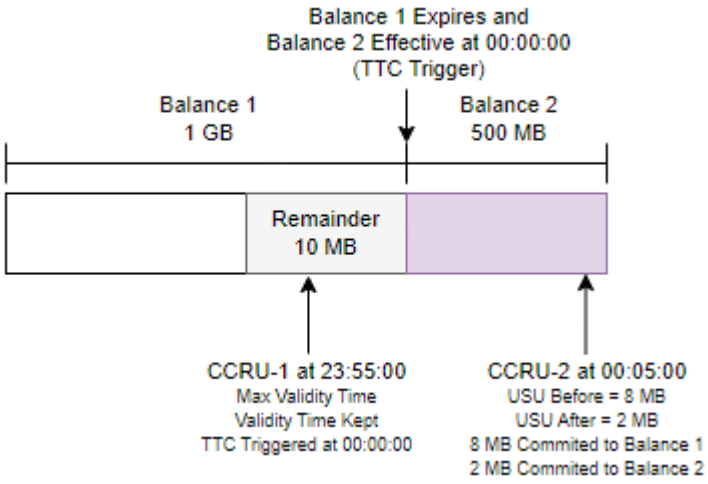
Linear rating reduces Balance 1 from 10 MB to 5 MB and Balance 2 from 500 MB to 495 MB.

How Non-Linear Rating Manages Tariff Boundaries

Non-linear rating does not assume a constant resource consumption rate. For example, it can track a customer who consumes 10 MB in the first five minutes and another 20 MB in the next five minutes. Non-linear rating calculates resource consumption based on the granted usage amount and uses Tariff Time Change (TTC) to manage tariff-boundary conditions. This method provides precise usage reporting before and after tariff-boundary changes.

Figure 12-2 shows a non-linear rating example in which ECE grants 10 MB in CCRU-1 from 23:55:00 until 23:59:59 because a TTC occurs at midnight. ECE grants a separate usage amount in CCRU-2 from 00:00:00 until the next TTC or credit-limit boundary.

Figure 12-2 Non-Linear Rating Example



In this example, CCRU-2 reports that the customer consumed 8 MB before the TTC and 2 MB after the TTC. Non-linear rating distributes the usage as follows:

- 8 MB is consumed between 23:55:00 and 23:59:59 and charged to Balance 1
- 2 MB is consumed between 00:00:00 and 00:05:00 and charged to Balance 2

Balance 1 reduces from 10 MB to 2 MB, and Balance 2 reduces from 500 MB to 498 MB.

How ECE Reserves Resources for Non-Linear Rating

ECE applies non-linear rating to all usage by default. Non-linear rating enables ECE to rate long network sessions incrementally based on the exact data consumed between tariff changes. It also lets operators show subscribers a running balance after each tariff change based on actual data consumption.

When using non-linear rating, ECE determines whether a tariff change occurs during the authorization and reservation process for a session request from the network. ECE creates a reservation for resources before and after the TTC and sends a Tariff-Time-Change AVP in the

Copyright © 2019, 2026, Oracle and/or its affiliates.

response so the network can report the exact data consumed before and after the tariff change.

Additionally, ECE conducts reverse rating to calculate the amount of usage the subscriber can afford and reserves the balance for the RSUs based on the worst-case charging condition (the maximum charge that can be applied for the RSUs). This process ensures that the overall usage does not exceed the customer's credit limit and that there is no revenue leakage, regardless of whether the balance is consumed before or after the tariff change.

How ECE Creates Separate Rated Events with Nonlinear Rating and TTC

When non-linear rating and TTC are enabled, ECE may create separate rated events for the same network session. This can occur even when midsession CDR generation is disabled, because the rated-event split is based on rating and tariff boundary processing, not only on the midsession CDR generation setting.

Using RAR for In-Session Quota and Policy Updates

RAR is recommended when the charging system must actively update a subscriber's quota or charging rules while the subscriber is consuming a service, such as after an in-session purchase, adjustment, or balance change. If RAR is not enabled in the network, verify how the deployment handles in-session quota and policy updates.

How ECE Applies Non-Linear Rating at Time-Model Boundaries

When a usage request spans a time-model boundary where one timeModel rate plan applies before the boundary and another rate plan applies after the boundary, ECE evaluates the request across the changeover and returns a TTC time point at the boundary. The TTC reason is VALIDITY_BASED.

Configuring Linear or Non-Linear Rating

You can configure ECE to use linear or non-linear rating for all usage events, for specific products, or for specific product and event combinations. By default, ECE uses non-linear rating for all usage events.

- To configure ECE to use linear rating for all usage events, see "Configuring ECE to Use Linear Rating for All Usage Events".
- To override the default rating type for individual products, see "Configuring Linear or Non-Linear Rating for Specific Products".
- To override the default rating type for specific event-and-product combinations, see "Configuring Linear or Non-Linear Rating for Product and Event Combinations".

> **Note**
>
> For data sessions, enable non-linear rating and TTC for all usage events. Also, enable non-linear rating for the applicable data product or data event-and-product combination. This configuration is required when an in-session purchase or balance adjustment affects data usage balances.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring ECE to Use Linear Rating for All Usage Events

By default, ECE uses non-linear rating for all usage events. Non-linear rating enables ECE to rate long-running sessions incrementally based on the exact usage consumed between tariff changes.

You can configure ECE to use linear rating for all usage events instead. When you use linear rating, ECE evenly distributes resource consumption across the usage period and does not split rating based on Tariff-Time-Change processing.

To configure ECE to use linear rating for all usage events:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **attributes**.
5. Select the **nonLinearRatingEnabled** attribute and set its value to **false**.

   The default is **true**.
6. Select the **tariffTimeChangeSupported** attribute and set its value to **false**.

   The default is **true**.

Alternatively, in the *ECE_home*/**config/management/charging-settings.xml** file, set the **nonLinearRatingEnabled** and **tariffTimeChangeSupported** entries to **false**. Restart ECE for the changes to take effect.

Configuring Linear or Non-Linear Rating for Specific Products

You can override the default rating type for specific products. By default, ECE applies the rating type specified by **nonLinearRatingEnabled** to all usage events. For example, you might configure ECE to apply non-linear rating to usage events by default and apply linear rating to voice usage.

To override the default rating type for a specific product:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.reservationConfig**.
4. Expand **Operations**.
5. For each product that requires an override, select **enabledOrDisableNonLinear** and then set these attributes:
   - **productType**: Specify the product name defined in the ECE request specification data, such as TelcoVoice.
   - **enableOrDisable**: Set this attribute as follows:
     - **false**: Use linear rating.
     - **true**: Use non-linear rating.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring Linear or Non-Linear Rating for Product and Event Combinations

You can override the default rating type for specific event-and-product combinations. By default, ECE applies the rating type specified by **nonLinearRatingEnabled** to all usage events.

> **Note**
>
> The initial product-and-event configuration is applied when ECE is deployed. Subsequent changes to this configuration are not applied dynamically through JConsole. To change the linear or non-linear rating setting for a product-and-event combination, update the **charging-settings.xml** file and redeploy ECE.

To override the default rating type for a specific product and event combination:

1. Open the *ECE_home*/**config/management/charging-settings.xml** file.
2. In the **reservationConfig** section, locate the **productConfigurationGroup** element.
3. Add a separate **productConfiguration** element for each product and event combination that requires an override.

   > **Note**
   >
   > Use separate **productConfiguration** elements when events for the same product type require different linear or non-linear rating behavior.

4. Set the following attributes for each **productConfiguration** element:
   - **type**: Specify the product type defined in the ECE request specification data.
   - **eventType**: Specify the event type to which the configuration applies.
   - **nonLinearEnabled**: Set this attribute as follows:
     - **false**: Use linear rating for this product and event combination.
     - **true**: Use non-linear rating for this product and event combination.
5. Deploy or redeploy the ECE Helm chart.

For example, the following configuration uses linear rating for SMS events and non-linear rating for voice and data events for the **Telco1G** product type:

```
<productConfigurationGroup config-class="java.util.ArrayList">
   <productConfiguration
      config-
class="oracle.communication.brm.charging.appconfiguration.beans.reservation.ProductConfig
uration"
      connectionTimeEnabled="true"
      eventType="EventDelayedSessionTelco1GSms"
      nonLinearEnabled="false"
      type="Telco1G">
      <rumConfigurationGroup config-class="java.util.ArrayList">
         <!-- SMS RUM configuration -->
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
      </rumConfigurationGroup>
   </productConfiguration>

   <productConfiguration
      config-
class="oracle.communication.brm.charging.appconfiguration.beans.reservation.ProductConfig
uration"
      connectionTimeEnabled="true"
      eventType="EventDelayedSessionTelco1GVoice"
      nonLinearEnabled="true"
      type="Telco1G">
      <rumConfigurationGroup config-class="java.util.ArrayList">
         <!-- Voice RUM configuration -->
      </rumConfigurationGroup>
   </productConfiguration>

   <productConfiguration
      config-
class="oracle.communication.brm.charging.appconfiguration.beans.reservation.ProductConfig
uration"
      connectionTimeEnabled="true"
      eventType="EventDelayedSessionTelco1GData"
      nonLinearEnabled="true"
      type="Telco1G">
      <rumConfigurationGroup config-class="java.util.ArrayList">
         <!-- Data RUM configuration -->
      </rumConfigurationGroup>
   </productConfiguration>
</productConfigurationGroup>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

13

Managing Midsession-Rated Events

You can configure Oracle Communications Elastic Charging Engine (ECE) to generate a rated event during the middle of a network session based on the trigger criteria that you specify.

Topics in this document:

- About Midsession-Rated Events
- Generating Midsession-Rated Events for Charge Offer Changes
- Generating Midsession-Rated Events When the USU is Block Missing
- Configuring Threshold-Based Tier Selection for Split Rating Periods
- Customizing the Worst-Case Charging Reservation
- Viewing the Reason for a Midsession-Rated Event

About Midsession-Rated Events

By default, ECE generates a rated event for a network session when a session ends. You can also configure ECE to generate rated events when an update operation occurs during the 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 these 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)
- Custom criteria, such as network condition changes or spans from one offer to another due to balance exhaustion

> **Note**
>
> To trigger a midsession-rated event based on custom criteria, you must extend ECE at the pre-rating or post-rating extension points. See "Pre-Rating Extension – Generating Midsession-Rated Event" and "Post-Rating Extension – Generating Midsession-Rated Events" for more information.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> ECE checks for trigger conditions only during update operations. For example, 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.

Configuring ECE to Generate Midsession-Rated Events

To configure ECE to generate midsession-rated events:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.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 13-1.

Table 13-1 Fields for Defining Midsession-Rated Event Triggers

Field	Description
productType	The name of the service for which you are creating the trigger (for example, **"DATA"**).
eventType	The name of the event for which you are creating the trigger (for example, **"DATA_USAGE"**). **Note:** You must specify only a single event for **eventType**. Multiple comma-separated event names are not supported. To generate midsession-rated events for multiple events, configure a separate trigger for each service-event pair.
triggerName	The name of the trigger you are defining.
qtyFields	The name of one or more event fields to which a quantity condition applies (for example, **"input_volume; output_volume"**). Use a semicolon (**;**) to separate field names. Values in the fields are summed.
qtyUnit	The unit of measure for conditions based on quantity (for example, **"MEGABYTES"**).
qtyValue	The total quantity of the unit that triggers event generation (for example, **"70"**).
durationUnit	The unit of measure for conditions based on duration (for example, **"HOURS"**).
durationValue	The amount of the unit that triggers event generation (for example, **"70"**).

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 13-1 (Cont.) Fields for Defining Midsession-Rated Event Triggers

Field	Description
timeOfDay	The particular time of day in a 24-hour clock at which to generate the event (for example, **"23:00:00"**, which indicates 11 p.m.). Use the *hh*:*mm*:*ss* format.
midSessionForNonCounterGrants	Whether to generate an event based on the consumed unit.
offerChangeTrigger	Whether to create midsession-rated events for charge offer changes. Can have values of **true** or **false**. **Note:** If this value is set to **true**, you must also enable the **postRatingMidSessionExtension** extension. For more information, see "Generating Midsession-Rated Events for Charge Offer Changes".

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.

> **Note**
>
> - All conditions in a **TriggerConfiguration** block must be met (criteria are assumed to be joined by AND).
> - If a trigger contains multiple **TriggerConfiguration** blocks, the conditions in only one block must be met (blocks are assumed to be joined by OR).

Generating Midsession-Rated Events for Charge Offer Changes

You can configure ECE to generate midsession-rated events whenever it starts rating a customer's usage with a new charge offer. For example, assume a session that consists of the following:

- Talk time reported: 30 minutes
- First 12 minutes are rated using Charge Offer 1
- Next 15 minutes are rated using Charge Offer 2
- Last 3 minutes are rated using Charge Offer 3

For this session, ECE generates two midsession-rated events:

- CDR 1 for the first 12 minutes
- CDR 2 for the next 15 minutes

The remaining 3 minutes stay in the active session.

Copyright © 2019, 2026, Oracle and/or its affiliates.

If the session is split into multiple CDRs, the start and end times of each CDR can be estimated.

To generate midsession-rated events when rating moves from one charge offer to another, you must set **offerChangeTrigger** to **true** for the applicable service-event pair, and set **postRatingMidSessionExtension** to **oracle.communication.brm.charging.sdk.extensions.SamplePostRatingMidSessionExtension** or to the fully qualified class name of your custom extension.

To configure ECE to generate midsession-rated events for charge offer changes:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.midSessionCdrConfiguration**.
4. Expand **Attributes**.
5. Set the **MidSessionCdrEnabled** attribute to **true**.
6. To define a new trigger, follow the steps in "Configuring ECE to Generate Midsession-Rated Events".
7. Expand **Operations**.
8. Click **addOrUpdateMidSessionCdrTriggerDetails**.
9. In the **offerChangeTrigger** field, enter **true**.
10. Under the **ECE Configuration** node, expand **charging.extensions**.
11. Expand **Attributes**.
12. Set the **postRatingMidSessionExtension** attribute to **oracle.communication.brm.charging.sdk.extensions.Sample postRatingMidSessionExtension** or to the name of your custom extension.

> **Note**
>
> ECE does not generate midsession-rated events when:
>
> - Only one charge offer is applied to the rating result.
> - The **postRatingMidSessionExtension** extension hook class is not defined.
> - The charge offer has a rate plan with multiple tiers.

Table 13-2 describes how ECE behaves based on the settings of the **postRatingMidSessionExtension** attribute, the **offerChangeTrigger** attribute, and whether the rate plan has multiple tiers.

Table 13-2 Attribute and Rate Plan Settings

postRatingMidSessionExtension is defined	offerChangeTrigger is enabled	Rate plan is multi-tier	Rated Event
True	True	False	A midsession-rated event is created for each charge offer, except for the last one.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 13-2 (Cont.) Attribute and Rate Plan Settings

postRatingMidSession Extension is defined	offerChangeTrigger is enabled	Rate plan is multi-tier	Rated Event
True	True	True	No midsession-rated events are created for charge offer changes.
True	False	True or False	No midsession-rated events are created for charge offer changes.
False	True	True or False	No midsession-rated events are created for charge offer changes.
False	False	True or False	No midsession-rated events are created for charge offer changes.

Generating Midsession-Rated Events When the USU is Block Missing

When the network sends a Final Unit Indicator (FUI) followed by a top up, ECE generates a reauthorization request (RAR). The network then sends an update request, sometimes not including the Used Service Units (USU) block for the ongoing session. When this occurs, ECE generates a midsession-rated event. This ensures that any part of the reservation consumed by the account is reported, preventing revenue loss.

Configuring Threshold-Based Tier Selection for Split Rating Periods

You can control which threshold-based tier ECE uses when rating a session that is divided into split rating periods, such as after a tariff change, balance update, or validity change.

You can configure ECE to select the threshold tier based on one of the following quantities:

- The original session quantity
- The quantity of each split rating period

For example, assume that a customer has a 35-second voice session that is divided into two split rating periods: a 33-second period and a 2-second period. The threshold-based voice rate plan includes the following threshold tiers:

- Tier 1: Quantities of 2 seconds or less are charged at 0.
- Tier 2: Quantities greater than 2 seconds are charged at a paid rate.

When ECE is configured to use the original session quantity, both the 33-second and 2-second split rating periods use the original 35-second session quantity for threshold tier selection and select Tier 2. When ECE is configured to use the quantity of each split rating period, the 33-second split rating period selects Tier 2, and the 2-second split rating period selects Tier 1.

The behavior of **QuantityIn[...]** is unchanged.

To configure how ECE selects quantity thresholds for split rating periods:

Copyright © 2019, 2026, Oracle and/or its affiliates.

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **attributes**.
5. Select the **thresholdBasedTierSelectionMode** attribute and set its value to one of the following:
   - **SPLIT_PERIOD_QTY**: Each split rating period uses its own quantity for threshold tier selection. This is the default.
   - **ORIGINAL_SESSION_QTY**: Split rating periods use the original session quantity, when available, for threshold tier selection.

Customizing the Worst-Case Charging Reservation

When resources are requested around tariff-time changes, ECE performs reverse rating to calculate the usage the subscriber can afford and reserves the balance for the requested service units based on the worst-case charging condition.

You can customize how ECE performs the worst-case charging reservation by using the ECE SDK. To do so, extend ECE at the post-rating or post-charging extension point to do the following in **ExtensionRatingPeriod**:

- Invoke the **isAdjustedForWorstCost** method, which determines whether worst-case charging was applied when calculating the reservation amount. The method returns a Boolean value.
- Add customizations to perform if **isAdjustedForWorstCost** is **true**.

For example:

```
Collection<ExtensionRatingPeriod> ratingPeriodCollection =
extensionContext.getRatingPeriods();

for (ExtensionRatingPeriod ratingPeriod : ratingPeriodCollection) {
        if(ratingPeriod.isAdjustedForWorstCost()){
                //your customizations
        }
}
```

Viewing the Reason for a Midsession-Rated Event

When ECE generates a midsession-rated event, it automatically records why the event was split in the event's **midSessionCDRSplitReason** field. ECE sets the field to one of the preconfigured reason codes in Table 13-3.

Table 13-3 Reasons for Creating Midsession-Rated Event

Reason Code	Description	Enum Value
CONFIGURED_VOLUME_REACHED	The event exceeded a configured volume, such as 100 MB.	1

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 13-3 (Cont.) Reasons for Creating Midsession-Rated Event

Reason Code	Description	Enum Value
CONFIGURED_DURATION_REACHED	The event exceeded a configured amount of time, such as 2 hours.	2
RATING_CONDITION_CHANGE	The trigger was caused by a change in tariffs.	3
CONFIGURED_TIME_OF_THE_DAY_CROSSED	The event crossed a configured time of day, such as midnight.	4
INTERNAL_TRIGGER	The trigger was caused by a context change.	5
EXTERNAL_TRIGGER	The midsession trigger condition was met.	6
MULTIPLE_USU	The event contained multiple Used Service Units (USUs). ECE generates a midsession-rated event for each USU.	7
MULTIPLE_USU_AND_EXTERNAL_TRIGGER_OR_INTERNAL_TRIGGER	The event was caused by any of the above triggers.	8
PRE_MIDSESSION_CONDITION	The event reached a custom trigger at the prerating extension point.	9
POST_MIDSESSION_CONDITION	The event reached a custom trigger at the post-rating extension point.	10
ZONE_CHANGE	The trigger was caused by a change in zone.	11
BALANCE_EXHAUST	The event was triggered after the complete exhaustion of an existing balance.	12

The following shows a sample snippet of a midsession-rated event with the **reasonForMidSessionCDRSplit** field set:

```
balanceOutputMap = {BALANCE=BalanceOutputImpl{ customerId=39384585, balanceId='BALANCE',
status='SUCCESS', firstUsageValidityMap{},
impacts{840=[OutImpactImpl{ balanceItemId=1, impact=UnitValue{quantity=60.00,
unit=Money{curr=USD}}, validityRule=null,
firstUsageCreatedTime=null, type=0}]}, grantValidityMap={840={1=[Pair{first=null},
{second=null}]}}, recurringImpactMap=null,
firstUsageValidityRuleMap={}, firstUsageCreatedTimeMap={},
currentAndLoanAmounts={840=[Pair{first=60.00}, {second=0}]}}}}'}}
{reasonForMidSessionCDRSplit=[CONFIGURED_DURATION_REACHED]}]
```

You can also extend ECE to add these preconfigured reason codes to midsession-rated events generated by custom criteria. To do so, you extend ECE at the pre-rating and post-rating midsession extension points.

For more information, see these ECE SDK sample programs:

- **SamplePreRatingMidSessionExtension**. See "Pre-Rating Extension – Generating Midsession-Rated Event".
- **SamplePostRatingMidSessionExtension**. See "Post-Rating Extension – Generating Midsession-Rated Events".

Copyright © 2019, 2026, Oracle and/or its affiliates.

14

Managing Online Charging Sessions

You can learn how to configure online charging sessions in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- Managing Dynamic Charging Overrides for Online Sessions
- Enabling Server-Initiated Reauthorization Requests
- Configuring ECE to Return Remaining-Balance Information in Usage Responses
- Configuring Default Taxes in ECE
- Logging Prepaid Usage Overage
- Managing Direct Debit Data in ECE Cache
- Configuring ECE to Redirect Subscriber Sessions to a Service Portal
- Enabling Match Factor in ECE
- Customizing Consumption Order of Loan and Default Balances
- Location-Based Charging
- About Processing Conditional Balance Impacts
- Customizing ECE to Apply Fixed Charge for Zero Usage
- Accessing ECE Configuration MBeans
- About Overload Protection in ECE
- Redirecting Customer Sessions to a Service Portal

Managing Dynamic Charging Overrides for Online Sessions

Dynamic charging allows you to override an offer's default rate on a per customer basis based on the date. For example, you could create a dynamic override that charges new customers $0.04 per MB for the first six months, $0.05 per MB for the next four months, and then the default rate for all subsequent months.

You define dynamic charging overrides by creating a pricing tag and associating it with an event type, date ranges, and an amount. For example, you could create a pricing tag named **MyPrice** associated with a $0.05 per MB charge for **/event/session/telco/gsm** events from June through September. You could then add the pricing tag to your offers' balance impacts.

You configure dynamic charging overrides using PDC or the **ImportExportPricing** utility. See "Configuring Dynamic Pricing for Usage Events" in *PDC Creating Product Offerings*.

BRM stores information about dynamic charging overrides in the **/offering_override_values** object.

During rating, ECE determines whether a pricing tag in a balance impact matches a pricing tag and date range combination in the **/offering_override_values** object. If it finds a match, ECE charges the amount associated with the pricing tag. If it does not find a match, ECE charges the default amount.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 by the following conditions:

- 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 in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.
4. Expand **Attributes**.
5. Set the **rarNotificationMode** attribute to **ASYNCHRONOUS**.

   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**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.
   - 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. This is the default.

> **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.

Customizing Server-Initiated Reauthorization for Sharing Groups

By default, when ECE performs a server-initiated RAR during a customer's ongoing session, it generates RAR notifications for all of the customer's ongoing sessions.

You can customize ECE to support server-initiated RARs for sharing group members. In this case, ECE initiates RAR notifications for all ongoing sessions of the customer responsible for triggering the RAR along with all ongoing sessions of the sharing group members. For example, assume a sharing group includes accounts A, B, C, and D, and accounts A and C have ongoing sessions using the sharing group balance. If an RAR is triggered for account C, ECE sends RAR notifications for account A's and C's ongoing sessions.

To do so, extend ECE at the prerating, post-charging, and post-rating extension points and use these ECE SDK sample programs: **SampleRarPostChargingExtension**, **SampleRarPostRatingExtension**, and **SampleRarPreRatingExtension**. See "Rating/Charging Extension – Triggering RAR Notifications" for more information.

Configuring ECE to Return Remaining-Balance Information in Usage Responses

You can configure ECE to return a customer's remaining-balance information in the usage response. 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 can return remaining-balance information for initiate and update usage requests. For DEBIT_UNIT usage requests, you can also configure whether ECE aggregates the currency and non-currency balance values returned in the response

The remaining-balance information that ECE returns pertains to all balances impacted by the usage request.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.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:

     ```
     remaining balance = sum of 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:

     ```
     remaining balance = {credit limit - sum of valid sub-balances of (current balance + consumed reserved amount)}
     ```
   - **Debit_Unit**: You can configure whether ECE returns aggregated or non-aggregated balance values for currency and non-currency balances in usage responses.
6. To configure the aggregation of balance values returned for DEBIT_UNIT usage requests, also set the following attributes:
   - **aggregateCurrencyBalanceForDebitUnit**: Specifies whether ECE aggregates currency balance values in **Debit_Unit** usage responses.
     - **true**: When set to true, ECE returns aggregated currency balance values. This is the default.
     - **false**: ECE returns non-aggregated currency balance values.
   - **aggregateNonCurrencyBalanceForDebitUnit**: Specifies whether ECE aggregates non-currency balance values in **Debit_Unit** usage responses.
     - **true**: ECE returns aggregated non-currency balance values.
     - **false**: ECE returns non-aggregated non-currency balance values. This is the default.

     Alternatively, you can configure these attributes in the *ECE_home***/config/management/charging-settings.xml** file and then restart ECE for the changes to take effect.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring Default Taxes in ECE

You configure taxes in Pricing Design Center (PDC). In PDC, you also define the following for each charge:

- When to apply taxes: at event time, during billing, or dynamically.
- How to apply taxes: with tax codes or tax selectors. Tax selectors apply tax codes based on account, service, event, and profile attributes.

See "About Calculating Taxes" in *BRM Calculating Taxes* for more information.

PDC publishes this tax configuration information to BRM and ECE. ECE honors these tax assignments during rating.

If a charge does not include a tax time or tax rate, ECE applies tax during billing and assigns the default tax code and rate that you configure in ECE.

> **Note**
>
> ECE supports only simple, fixed-rates, such as GST or VAT, for the default tax.

To configure ECE to apply a default tax rate, tax code, and general ledger ID (G/L ID) for a charge offer or a discount offer:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.taxation**.
4. Expand **Operations**.
5. Click **addTaxDetails**.
6. Specify values for the following required parameters:

> **Note**
>
> These parameters are mandatory. You must set all of them when configuring taxation.

   - **taxCode**: Enter the tax code for the associated charge offer or discount offer. Use the same tax code you defined in PDC for the related charge or discount offer.

     ECE uses this tax code to look up the tax rate when processing usage for the associated offer.
   - **taxRate**: Enter the tax rate to apply. For example, **0.20** applies a 20% tax to the total usage impact.
   - **taxGlId**: Enter the G/L ID to use for the tax impact.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> The **taxGlId** attribute specifies only the default tax G/L ID. To assign tax G/L IDs based on the rating period type or other event attributes, implement the logic in a custom extension at the post-rating extension point.

7. For each additional charge offer or discount offer that requires taxation, specify another set of values for **taxCode**, **taxRate**, and **taxGlId**.

Logging Prepaid Usage Overage

You can configure ECE to capture any overage amounts by prepaid customers during an active session, which can help you prevent revenue leakage. Suppose the network reports that the number of used units during a session exceeds a customer's available allowance and credit limit. In that case, ECE charges the customer up to the available allowance. It can then add information about the overage amount to an overage log file. You can create a custom solution to reprocess the overage amount later on.

> **Note**
>
> If your ECE system includes a Kafka Server, configure ECE to publish overage details to a Kafka topic instead. See "Understanding and Configuring ECE Prepaid Usage Overage".

To configure ECE to publish details about prepaid usage overage to its own log file:

1. Set the **OVERAGE_LOG_DIR** environment variable to the path where you want to create the overage log file.
2. In your *ECE_home***/config/ece.properties** file, add the following line:

   ```
   java.property.eceoveragelogdir=OverageLogDir
   ```

   where *OverageLogDir* is the absolute path to the directory where you want to create the overage log file.
3. In your *BRM_home***/sys/dm_kafka/log4j2.xml** file, add the following lines and set the XML elements to meet your business needs:

   ```
   <RollingFile name="OVERAGE" createOnDemand="true" fileName="${OVERAGE_LOG_DIR}/
   overage.log" filePattern="${OVERAGE_LOG_DIR}/overage.log.%i">
       <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5p] [%t] %m%n"/>
       <Policies>
           <SizeBasedTriggeringPolicy size="100MB"/>
       </Policies>
       <DefaultRolloverStrategy max="1"/>
   </RollingFile>
   ```

   For more information about the XML elements, see "Rolling file appenders" in the *Apache Log4j Manual*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Managing Direct Debit Data in ECE Cache

By default, ECE stores information about all direct debit events in its cache so it can validate any refund requests against one of the events.

However, some event types may not be eligible for refunds, so storing the direct debit information in the cache is unnecessary. To save space and improve performance, you can configure ECE not to store direct debit information in its cache for specific event types.

To prevent ECE from storing direct debit information in its cache for specific event types:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. In the **excludedEventsForDebitRefundSessions** attribute, list the event types separated by a comma. For example, enter **EventDelayedSessionTelcoGsm, EventDelayedSessionTelcoGprs**.

Configuring ECE to Redirect Subscriber Sessions to a Service Portal

You can configure ECE to redirect a customer session to a service portal when the customer is about to deplete the balance available for a service. A service portal is an external server where the customer can perform an action, such as adding funds to an account.

ECE evaluates redirection rules to determine whether to redirect a session. A rule specifies the service portal address, address type, and, optionally, the time during which the customer can access network resources while completing the action at the service portal.

A redirection rule can consider the following information:

- Whether final-unit indication (FUI) is present in the usage response.
- Whether the customer is roaming.
- The customer's business profile, such as a prepaid or postpaid payment type or a voice or data subscription type.
- The service type and attributes in the usage request.

Each rule can redirect a session to a different service portal. For example, you can redirect a roaming prepaid customer to a roaming top-up portal and a non-roaming prepaid customer to a home-network top-up portal.

When a credit-control client receives FUI in a usage response, it processes the session according to the Final-Unit-Action:

- REDIRECT redirects the customer to the service portal specified by ECE.
- TERMINATE ends the customer session.

If no configured redirection rule matches, ECE returns a Final-Unit-Action of TERMINATE.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

For example, the following redirection rule specifies that when a customer is roaming, the redirect address is **http://RedirectRoaming.com** and the redirect address type is **URL**:

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

This redirection rule specifies that when a customer is both postpaid and roaming, the redirect address is **http://RedirectRoaming.com**, the redirect address type is **URL**, and the maximum redirection time is 900 seconds:

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

Table 14-1 shows redirection-rule conditions that you can use to create redirection rules.

Table 14-1 ECE Redirection-Rule Conditions

ECE Redirection-Rule Conditions	Description
@fui	Checks in the charging result if the customer has insufficient funds (finds the Final Unit Indicator in the service context). @fui is required.
{business_profile([name:"*Business ProfileName*"])} == "true")	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. For example: `{business_profile([name:"POSTPAID"])} == "true" )`
@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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 14-1 (Cont.) ECE Redirection-Rule Conditions

ECE Redirection-Rule Conditions	Description

{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 14-1 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 website 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **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** 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 the match factor in discounting.

To enable the match factor in ECE:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **matchFactorEnabled** attribute to **true**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Customizing Consumption Order of Loan and Default Balances

By default, ECE includes valid loan balance items when rating usage requests. To use default balance items instead for specific service or event types, configure the pre-rating extension to set

```
extensionContext.setUseLoanBalanceItem(false)
```

.

When this setting is **false**, ECE excludes valid loan balance items for that usage request and rates the request using only eligible default balance items, according to the configured rating and consumption rules. ECE resets the setting to **true** for the next usage request. This setting does not create, grant, or modify loan balances.

You can use a pre-rating extension to configure selected services or events to exclude loan balance items for selected services or events before the loan balance.

In your extension logic, evaluate the service type, event type, or other usage-request attributes. For requests that must use the default balance first, call the **setUseLoanBalanceItem** method with a value of **false**. For example:

```
if (usePrincipalBalanceFirst) {
    extensionContext.setUseLoanBalanceItem(false);
}
```

The **setUseLoanBalanceItem** method supports the following values:

- **true**: ECE includes valid loan balance items during rating. This is the default.
- **false**: ECE excludes loan balance items during rating and uses valid default balance items according to the rating and consumption rules.

> **Note**
>
> - The value of **setUseLoanBalanceItem** applies only to the current usage request. ECE resets the value to true at the start of each usage request. Therefore, an override made for one request does not affect subsequent requests.
> - The value controls only whether existing valid loan balance items are eligible for rating. It does not create or grant a loan balance.

For more information, see "Customizing Rating"

Location-Based Charging

Location-based charging allows you to alter the rate based on your customer's location. For example, you could charge $5 per GB for customers located in Silicon Valley.

You configure location-based charging in PDC by:

- Creating a value map that associates a value map name with a set of cell IDs. For example, you could create a value map named **Silicon Valley** associating with cell IDs **209**, **408**, **415**, **650**, and **510**.

  See "Configuring Value Maps" in *PDC Creating Product Offerings* for more information.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Creating a charge selector or discount selector that maps a value map name to a specific charge or discount.

  For example, you could map the London value map to a 5 pound per GB charge and map the Manchester value map to a 6 pound per GB charge. See “Charge Selectors” in *PDC Online Help*.

- Adding the charge selector to a charge offer, or adding the discount selector to a discount offer.

When ECE rates an incoming usage request, it determines whether a cell ID in the request is included in a value map. If it is included, ECE checks whether:

- A charge selector associates the value map name to a charge amount.
- A discount selector associates the value map name to a discount amount.

If ECE finds a match, it applies the charge or discount associated with the value map name.

About Processing Conditional Balance Impacts

You can create conditional balance impacts when designing charge offers in Pricing Design Center (PDC). See "About Conditional Balance Impacts" in *PDC Creating Product Offerings*.

However, the way ECE processes conditional balance impacts is dependent on the pricing names assigned to each charge in an offering:

- **Apply Multiple Conditional Balance Impacts**: If you want multiple conditional balance impacts to apply for different types of charges within the same offer and day, use *different pricing names* for each charge. For example, to apply a charge for roaming in different countries, use unique pricing names for Zone 1 (Italy), Zone 2 (Austria), Zone 3 (Switzerland), and so on. This way, roaming in one zone won't affect the application of charges or grants in other zones. A customer in Italy consuming up to 1 GB of roaming data in a day is charged $5. However, a customer consuming data in both Italy and Austria on the same day is charged $10 (that is, $5 for 1 GB of roaming data in Italy and $5 for 1 GB of roaming data in Austria).
- **Apply a Single Conditional Balance Impact**: If you want only one conditional balance impact to apply on a given day for a specific offering, ensure that all charges in the offering use the *same pricing name*. This is useful when you have specific logic to choose between different pricing configurations but want only one to take effect. For example, you could apply $5 per day for 1 GB of data when roaming in one or more zones.

When ECE rates events, it determines whether to apply a conditional balance impact by checking whether it is the first time the customer has used a bundle and pricing combination (called a bundle identifier) on that day.

Bundle identifiers use the following format: *bundleName_pricingName*

where *bundleName* is the bundle name assigned in PDC, and *pricingName* is the name assigned to the charge in PDC. For example, a bundle identifier could be **Roaming_InternationalPricing** or **Roaming_Zone1Pricing**.

Customizing ECE to Apply Fixed Charge for Zero Usage

By default, when a customer has a subscription with a charge offer that contains a fixed charge and a usage charge, ECE does not apply any charges if the usage charge is zero. For example, if a customer makes a call and it disconnects immediately, the reported usage is zero and the total charge becomes zero even if there is a fixed charge.

Copyright © 2019, 2026, Oracle and/or its affiliates.

You can customize ECE to apply the fixed charge even if the usage charge is zero or rounded down to zero. To do this, extend ECE at the post-rating extension point to do the following:

- Identify the **pricingName** for subscriptions with zero usage and exclude the usage charge for it.
- Apply a fixed charge, by customizing the logic to create or modify the **quantityRated**, **timeRange**, **validity**, and **quantityRange** variables to set them to zero.

You can further let ECE re-calculate the NET_QUANTITY based on the newly created or modified data, applying the fixed charges.

For example:

```
for (ExtensionRatingPeriod erp : originalExtRatingPeriods) {
                if
(erp.getChargeRatePlanName().equals("RATE_PLAN_VOICE_001")) { /* condition to
detect 0 quantity case */
                    erp.markForDeletion();
                    if (erp.isFixedImpact()) {
                        LOG.debug("SamplePostRatingExtensionIJH.execute() :
mutating fixed impact => {}", erp);
                        // Drop and recreate with 0 quantity ...
                        erp.markForDeletion();

                        Range timeRange =
Range.make(erp.getTimeRange().start(), erp.getTimeRange().start());  // start
- start
                        UnitValue qty =
erp.getQuantity().subtract(erp.getQuantity());                          // 0.0
                        String aaResultTags = "";
                        if (erp.getAAResultTags() != null) {
                            aaResultTags =
Arrays.asList(erp.getAAResultTags()).toString();
                        }
                        Range validity = null;
                        if (erp.getValidityFromTime() != null &&
erp.getValidityToTime() != null) {
                            validity =
Range.make(erp.getValidityFromTime().toInstant().toEpochMilli(),
erp.getValidityFromTime().toInstant().toEpochMilli());
                        }
                        QuantityRange quantityRange =
QuantityRange.make(erp.getQuantityFrom(), erp.getQuantityFrom());

                        extensionContext.addRatingPeriod(
                           erp.getRatingPeriodId(),
                           timeRange,
                           erp.getImpact(),
                           erp.getRumName(),
                           erp.getCustomerId(),
                           erp.getBalanceId(),
                           erp.getProductId(),
                           erp.getBalanceElementId(),
                           erp.getBalanceElementCode(),
                           erp.getGeneralLedgerCode(),
                           erp.getTaxCode(),
                           erp.getTaxTime(),
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
                                erp.getTaxRate(),
                                erp.getOfferProfile(),
                                qty,
                                aaResultTags,
                                erp.getTimeTag(),
                                erp.getOrigZoneTag(),
                                erp.getPricingName(),
                                erp.getTimeModelName(),
                                erp.getZoneModelName(),
                                erp.getChargeRatePlanName(),
                                erp.getImpactCategory(),
                                erp.isFirstUsageGrant(),
                                validity,
                                quantityRange,
                                erp.getRatingPeriodType(),
                                erp.getCorrelationRatingPeriodId(),
                                new Pair<String, String>(erp.getItemType(),
erp.getItemIdentifier()),
                                erp.isRecurringBundleImpact(),
                                erp.getRecurringBundleDetails(),
                                erp.getOrigZoneTag(),
                                erp.getPurchasedProductOfferingId(),
                                erp.getOfferingId(),
                                erp.getProductOfferingExternalId(),
                                erp.getCombinedZoneResult(),
                                erp.isCounterImpact(),
                                erp.isFirstUsageOffer(),
                                erp.isFixedImpact(),
                                erp.isCurrencyImpact(),
                                erp.getInitialZoneResultTag(),
                                erp.getUsedZoneModelName(),
                                erp.getImpact(),
                                erp.getOriginalBalanceElementId(),
                                erp.getTaxJurisdiction(),
                                erp.getRateTag(),
                                erp.getLineage(),
                                erp.getDiscountInfo(),
                                erp.getAlterationRatePlanName(),
                                erp.getAlterationConfigurationName(),
                                erp.getAlterationPricingName());

                        } else {
                            // Drop scaled impacts ...
                            LOG.debug("SamplePostRatingExtensionIJH.execute() :
Dropping RATE_PLAN_VOICE_001 scaled impact => {}", erp);
                            erp.markForDeletion();
                        }
                    }
                }
```

Accessing ECE Configuration MBeans

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

1. Log in to the driver machine.

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Start the ECE charging servers (if they are not started).
3. Connect to the ECE charging server node enabled for JMX management.
   This is the charging server node set to **start CohMgt = true** in the *ECE_home***/config/eceTopology.conf** file, where *ECE_home* is the directory in which ECE is installed.
4. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
5. In the editor's MBean hierarchy, find the ECE configuration MBeans.

About Overload Protection in ECE

When the amount of incoming requests for ECE exceeds its capacity, it experiences an overload. This can lead to charging nodes becoming stuck or unresponsive. To avoid this, you can enable an overload protection mechanism.

When an overload happens, ECE starts rejecting the new incoming requests and informs the client about any submitted request that could not be processed. These requests can then be routed to some other alternative gateway or can be accepted by the same gateway once the overload condition is resolved.

You can configure ECE either to reject all incoming requests or to reject requests based on priority. For more information about how to configure ECE for overload protection, see "Configuring System Overload Protection" in *BRM System Administrator's Guide*.

> **Note**
>
> Priority-based overload protection is applicable only to HTTP Gateway and Diameter Gateway.

Redirecting Customer Sessions to a Service Portal

You can configure ECE to redirect a customer session to a service portal when the customer is about to deplete the balance available for a service. A service portal is an external server where the customer can perform an action, such as adding funds to an account.

ECE evaluates redirection rules to determine whether to redirect a session. A rule specifies the service portal address, address type, and, optionally, the time during which the customer can access network resources while completing the action at the service portal.

A redirection rule can consider the following information:

- Whether final-unit indication (FUI) is present in the usage response.
- Whether the customer is roaming.
- The customer's business profile, such as a prepaid or postpaid payment type or a voice or data subscription type.
- The service type and attributes in the usage request.

Each rule can redirect a session to a different service portal. For example, you can redirect a roaming prepaid customer to a roaming top-up portal and a non-roaming prepaid customer to a home-network top-up portal.

When a credit-control client receives FUI in a usage response, it processes the session according to the Final-Unit-Action:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **REDIRECT** redirects the customer to the service portal specified by ECE.
- **TERMINATE** ends the customer session.

If no configured redirection rule matches, ECE returns a Final-Unit-Action of TERMINATE.

Returning FUI in Initiate Responses

Before configuring a redirection rule that uses **@fui** for an Initiate request, configure ECE to return final-unit indication (FUI) in the Initiate response. In the charging settings, set the **enableFuiForInitiate** parameter to true.

When **enableFuiForInitiate** is set to true and a Requested-Service-Unit (RSU) leaves the balance exactly at the effective ceiling, ECE classifies the result as **CREDIT_CEILING_REACHED**. The relevant RSU flow creates the special ceiling-reached service event, and response processing converts that event to the **FINAL_UNIT_INDICATOR** reason code. The response remains successful, and the CCA can contain FUI. The response does not include the **CREDIT_CEILING_BREACH** reason.

> **Note**
>
> The FUI does not indicate that the Initiate request failed. ECE grants the remaining available units and directs the network to perform the final-unit action after the units are consumed.

If the available balance is already zero, the Initiate request fails regardless of the **enableFuiForInitiate** setting. The failure response can include reasons such as **NO_RATING_GRAPH_CONFIGURED** or **NO_RATED_QUANTITY**.

Configure the **enableFuiForInitiate** parameter using JConsole or by editing the **charging-settings.xml** file.

To configure the parameter using JConsole:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "".
2. Expand the ECE Configuration node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **enableFuiForInitiate** attribute to **true**.

To configure the parameter in the **charging-settings.xml** file:

1. Open the **ECE_home/*config*/management/charging-settings.xml** file.
2. Set the **enableFuiForInitiate** parameter to **true**:

   ```
   enableFuiForInitiatetrue/enableFuiForInitiate
   ```

3. Save the file.
4. Restart ECE to apply the change.

Configuring Redirection Rules

ECE derives the service portal address from configurable instructions that you define in redirection rules. You can base a redirection rule on one or more customer conditions:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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, such as a BRM business profile for prepaid or postpaid customers or voice or data subscriptions.

Each redirection rule can redirect a session to a different service portal. For example, you can configure rules for the following business scenarios:

- Given a prepaid customer who is roaming, redirect the subscriber to **http://***roamingURL*.
- Given a prepaid customer who is not roaming, redirect the subscriber to **http://** *roamingURL*.

First, determine the content of your redirection rules. Then configure the completed rules in ECE using JConsole or by editing the charging-settings.xml file.

To determine the content of the redirection rules:

1. Create a temporary file in a text editor.
2. Create each redirection rule on a single line with no line breaks. The examples in this topic include line breaks only for readability.
3. Create additional rules as required. Separate multiple rules with semicolons and enter the complete configuration on one line when you set the redirectionRule parameter.

Redirection-Rule Syntax

Use the following syntax:

```
(redirection_condition1 [AND redirection_condition2 ... AND redirection_conditionN]) =>
[redirect_type:"redirectType",redirect_address:"redirectAddress",redirect_validity:"redir
ectValidity"];
```

where:

- **redirection_condition** specifies a condition that must be met for ECE to send the specified redirect information in the usage response. See Table 14-2, "Conditions Allowed in Redirection Rules."
- **redirect_type** specifies the service portal address type, such as **URL**.
- **redirect_address** specifies the service portal address.
- **redirect_validity** optionally specifies, in seconds, how long the customer can access network resources while completing the action at the service portal. If omitted, ECE uses the default reservation validity time.

Table 14-2 Conditions Allowed in Redirection Rules

Condition	Description
@fui	Checks whether final-unit indication is present in the charging result. This condition is required for redirection.
{business_profile([name:"BusinessProfileName"])} == "true"	Checks whether the customer has the specified BRM business-profile attribute set to **true**.
@roamingRequest	Checks whether the request is for a roaming customer. Use **!** **@roamingRequest** to check for a non-roaming request.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 14-2 (Cont.) Conditions Allowed in Redirection Rules

Condition	Description
{request_attribute([name:"FieldName"])}	Checks for the presence of a usage-request attribute. To check a value, compare the expression with that value; for example, `{request_attribute([name:"CELL_ID"])} == "1234"` .
@productType	Retrieves the service type associated with the request.

To configure the redirection rules using JConsole:

1. Complete the redirection-rule configuration in a text editor.
2. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
3. Expand the ECE Configuration node.
4. Expand **charging.redirectionConfiguration**.
5. Expand **Attributes**.
6. Set the **redirectionRule** attribute to the text rules you created.

The default value is an empty string. If no rule is configured or no rule matches, ECE does not redirect the session and returns a Final-Unit-Action of **TERMINATE**.

To configure the redirection rules in the **charging-settings.xml** file:

1. Complete the redirection-rule configuration in a text editor.
2. Open the **ECE_home/*config*/management/charging-settings.xml** file.
3. Set the **redirectionRule** attribute to the text rules you created.
4. Save the file.
5. Restart ECE to apply the change.

   ECE begins using the updated redirection rules.

> **Note**
>
> - If a rule has invalid syntax, ECE logs a Rule Evaluation Failed error in the **charging-server** log files and leaves the redirection information empty in the usage response.
> - Edit the configuration in the text file where you created it, then copy the completed one-line configuration into JConsole or **charging-settings.xml**. This helps prevent errors when a rule is too long to view in full in JConsole.

Copyright © 2019, 2026, Oracle and/or its affiliates.

15

Managing Session Start and End Times

You can configure how Oracle Communications Elastic Charging Engine (ECE) sets the start and stop times for your customers' sessions.

Topics in this document:

- Using Session Connect Time for Charging
- Optimizing Network Signaling
- Configuring ECE to Align Validity Start and End of Conditional Balance Impacts and Charge Offers

Using Session Connect Time for Charging

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

You can configure ECE to instead use the session connect time, which is the time the session actually begins, as the session start time for calculating usage charges. You can do this by setting the **connectionTimeEnabled** entry in the *ECE_home*/**config/management/charging-settings.xml** file to **true**.

To use the session connect time for calculating usage charges:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.reservationConfig**.
4. Expand **Operations**.
5. For each product that you offer, do the following:
    a. Select **enabledOrDisableConnectionTime**.
    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).
        - **enableOrDisable**: Enter **true** to use the session connect time.
    c. Click the **enabledOrDisableConnectionTime** button.

Optimizing Network Signaling

Many times, bundles expire at midnight on a particular day. If your customers are using bundles at that time, the renewal and request messages that are transmitted at midnight can cause an undue network load. To prevent this, you can configure ECE to randomize the validity times for a service so that renewal requests do not occur simultaneously.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To optimize network signaling by randomizing validity times:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.
4. Expand **Attributes**.
5. Set **subscriptionCycleForwardMode** to **true**.

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 in a JMX editor, such as JConsole. See "Accessing 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

16

Managing Reservations for Online Sessions

You can configure how Oracle Communications Elastic Charging Engine (ECE) reserves your customers' resources for online sessions.

Topics in this document:

- Configuring Reservation Expiration and Validity
- Configuring a Minimum Quantity for Reservation
- Configuring Reservation Quota for Services
- Managing Dynamic Quotas for Online Sessions

Configuring Reservation Expiration and Validity

In an online session, the network sends the following to ECE:

- **Usage updates**: Keeps 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 reauthorization is needed.
- **Reauthorization requests**: Requests an extension of the session. In response, ECE determines whether the customer's balance is sufficient and, if so, reauthorizes the call.

ECE determines how many resources to reserve for a usage session and when the usage session expires by using the following information passed in the requests:

- **Reservation duration**: This amount is used to calculate the amount of resources to reserve for the usage session. For example, if the duration is 20 minutes and the rate is $2 per minute, ECE reserves $40 for the usage session. At the end of the duration, the client must ask for a reauthorization to extend the usage session.

  You do not want the duration 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 expires.
- **Validity time:** This specifies how long the reservation is valid. When the validity time ends, the client must send a usage report to ECE. If the network mediation client does not communicate the used service units (USU) within the validity time, 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.

  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.

ECE calculates when the reservation expires by adding together the reservation duration and validity time. For example, if the usage request specifies a reservation duration of 240 seconds and a validity time of 600 seconds, the reservation expires in 840 seconds.

When ECE receives a usage request that does not specify a validity time or reservation duration, ECE uses the default values specified.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To configure the default values for reservation expiration and validity:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.reservationConfig**.
4. Expand **Attributes**.
5. Specify the defaults for the following attributes:
   - **validityTime** (Long data type): Enter the amount of time, in seconds, that a reservation remains valid. The default is one hour.
   - **reservationDuration** (Long data type): Enter the amount of time, in seconds, that is used to calculate the amount of resources to reserve. The default is one hour.

   For detailed information, see **ReservationConfigMBean** in the **BizParamConfigMBean** package of *Elastic Charging Engine Java API Reference*.

Configuring a Minimum Quantity for Reservation

You can configure a minimum reservation quantity for charging events. 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.

> **Note**
>
> 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.

To configure the minimum quantity for reservation:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> 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 in a JMX editor, such as JConsole. See "Accessing 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.

      > **Note**
      >
      > 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 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 "Configuring Dynamic Quota" in *PDC Creating Product Offerings*.

If dynamic quota selector rules are configured for a service-event combination, ECE does one of the following depending on whether the network passes a requested service unit (RSU) in a usage request:

- **An RSU is passed in the usage request**: ECE evaluates and applies the dynamic quota selector rules to the usage request to derive the allocated quota. ECE returns the derived quota (in the GSU) and the following quota attributes in the usage response to the network:
  - **Quota holding time.** Specifies how long a granted quota can be idle before releasing the reservation.
  - **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.

  > **Note**
  >
  > When the dynamic quota validity time spans a tariff boundary and triggers Tariff Time Change (TTC) processing, the usage response includes the dynamic quota validity time and the applicable TTC information. When no TTC occurs, ECE returns the dynamic quota validity time without TTC information.

- **An RSU is not passed in the usage request**: ECE does not return a quota in its response to the network.

If dynamic quota selector rules are *not* configured for a service-event combination, ECE uses the default quota configuration for deriving the quota and quota attributes.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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".

Copyright © 2019, 2026, Oracle and/or its affiliates.

17

Managing Rounding and Consumption Rules

You can configure how Oracle Communications Elastic Charging Engine (ECE) rounds balance impacts and the order in which it consumes sub-balances.

Topics in this document:

- Configuring Rounding for a Resource
- Configuring Rounding for Reverse Rating on Multiple RUMs
- Configuring Systemwide Consumption Rules for Balances
- Configuring the Consumption Order for Offers with the Same Priority

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, see "Adding Rounding Rules for Specific Events" in *PDC Online Help*.

However, you can configure system-wide 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 USD.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. 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 website:

     https://docs.oracle.com/javase/8/docs/api/java/math/RoundingMode.html

     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, the authorized resource quantity may include a remaining balance amount that is less than the configured increment quantity. You can configure a systemwide rounding rule to allow ECE to consume this leftover amount.. For information about rounding in PDC, see "Specifying Rounding in Balance Impacts" in *PDC Creating Product Offerings.*

To configure whether ECE can consume a leftover authorized balance quantity that is less than the configured increment quantity:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **reverseRateUseAllBalances** attribute to one of the following values:
   - To allow ECE to consume leftover available balance amounts that are less than the configured increment quantity, enter **true**.

     This option allows customers to use all of their available balances. When an increment quantity is set for charging, ECE can consume the maximum available balance even when the remaining available balance is less than the increment quantity. ECE grants only up to the available balance, based on the requested unit.
   - To prevent ECE from consuming leftover available balance amounts that are less than the configured increment quantity, enter **false**.

     This option allows ECE to consume the available balance only for complete increment quantities. A leftover balance that is less than the increment quantity remains ungranted and unconsumed.

Copyright © 2019, 2026, Oracle and/or its affiliates.

The default is **false**.

Configuring Systemwide Consumption Rules for Balances

When more than one validity-based sub-balance 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, see "Specifying the Order in Which Sub-Balances Are Consumed" in *PDC Creating Product Offerings*.
- Configuring consumption rules in BRM, see "Specifying the Order in Which Resource Sub-Balances Are Consumed" in *BRM Setting Up Pipeline Pricing*.

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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **systemConsumptionRule** attribute to one of the following systemwide consumption rules:
   - **EARLIEST_START**
   - **LATEST_START**
   - **EARLIEST_EXPIRATION**
   - **LATEST_EXPIRATION**
   - **EARLIEST_START_LATEST_EXPIRATION**
   - **EARLIEST_START_EARLIEST_EXPIRATION**
   - **LATEST_START_LATEST_EXPIRATION**
   - **LATEST_START_EARLIEST_EXPIRATION**
   - **EARLIEST_EXPIRATION_EARLIEST_START**
   - **EARLIEST_EXPIRATION_LATEST_START**
   - **LATEST_EXPIRATION_EARLIEST_START**
   - **LATEST_EXPIRATION_LATEST_START**
   - **NONE**: When the attribute is set to **NONE**, the default consumption rule is not configured, and the order for consuming balances is undefined.

   By default, this attribute is set to **EARLIEST_START_EARLIEST_EXPIRATION**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Configuring the Consumption Order for Offers with the Same Priority

ECE consumes allowances from offers based on their configured priority, from the highest priority to the lowest priority. If multiple offers have the same priority, ECE determines which offer to consume allowances from first based on the value of the **offerSelectionModeOnEquiPriorityOffers** entry in your **charging-settings.xml** file or your **oc-cn-ece-helm-chart/values.yaml** file:

- **START_TIME**: ECE consumes the allowance from the offer with the earliest validity start date. This is the default.
- **END_TIME**: ECE consumes the allowance from the offer with the earliest validity end date.

For example, assume a customer has purchased the following offers, which have the same priority:

- Offer A grants 1 GB of data with a validity period from Jan 1 through Jan 30.
- Offer B grants 200 MB of data with a validity period from Jan 10 through Jan 20.

On Jan 15, the customer downloads 100 MB of data. If the entry is set to **START_TIME**, ECE first consumes the data allowance from Offer A. If the entry is set to **END_TIME**, ECE first consumes the data allowance from Offer B.

During ECE runtime, you configure the consumption order as follows:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Select the **offerSelectionModeOnEquiPriorityOffers** attribute and set it to **START_TIME** or **END_TIME**.

Configuring Balance Impact Rounding in ECE

You can configure rounding for a currency and noncurrency balance impact by specifying rules for how Oracle Communications Elastic Charging Engine (ECE) rounds the rating impact amounts it calculates. The rules you specify are applied at the system level so the rounding of the rating impact amounts apply across all charges, discounts, and chargeshares. The rounding is also applied to reverse rating calculations and taxation calculations.

The rules you can specify for rounding are as follows:

- Scale

  You can specify a rule for how many digits to the right of the decimal point to allow. The default scale is **2**.

- Rounding mode

  You can specify a rule for the rounding behavior according to the Java math rounding enum. The default is **HALF_UP**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

See the Java SE API Reference documentation for information about using the Java math rounding enum.

For information about configuring rounding for currency and noncurrency balance impacts, see "Configuring Rounding for a Currency Balance Impact " and "Configuring Rounding for a Noncurrency Balance Impact".

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 USD.

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 points 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.

Configuring Rounding for a Currency Balance Impact

You can configure rounding for a currency balance impact by specifying rules for how ECE rounds the rating impact amounts it calculates.

The rules you specify are applied as a systemwide configuration for rounding the rating impact amounts of all charges, discounts and chargeshares. The rounding is also applied to all reverse rating calculations and taxation calculations.

See "Configuring Rounding for a Noncurrency Balance Impact" for information about configuring rounding for noncurrency balance impacts.

To configure rounding for a currency balance impact:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Specify values for the following attributes:
   - **currencyScale**: 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**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **currencyRoundingMode**: Enter the rounding mode that determines the rounding behavior by entering the string representation of the Java math rounding enum.

  See the Java SE API Reference documentation for information about using the Java math rounding enum.

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

  The default is **HALF_UP**.

6. Save your changes.

Configuring Rounding for a Noncurrency Balance Impact

You can configure rounding for a noncurrency balance impact. The rules you specify are applied as a systemwide setting and apply for rounding the rating impact amounts of all charges, discounts, and chargeshares. The rounding is also applied to all reverse rating calculations and taxation calculations.

To configure rounding for a noncurrency balance impact:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Specify values for the following attributes:
   - **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**.
   - **nonCurrencyRoundingMode**: Enter the rounding mode that determines the rounding behavior by entering the string representation of the Java math rounding enum.

     See the Java SE API Reference documentation for information about using the Java math rounding enum.

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

     The default is **HALF_UP**.
6. Save your changes.

For information about configuring rounding for currency balance impacts, see "Configuring Rounding for a Currency Balance Impact ".

Configuring Rounding When Authorizing Multiple RUMs

When ECE receives an online charging request for a specified amount of usage, ECE uses the customers product offerings to determine if the customer's balance amount is sufficient for the requested amount. If the request applies to multiple RUMs; for example, occurrence and duration, fractional values of the authorized balance quantity might result. You can enable ECE to round up the authorized balance quantities to the nearest whole number.

Copyright © 2019, 2026, Oracle and/or its affiliates.

If your business requires that your customers must be able to use *all* of their balances, configure ECE to round up the authorized balance quantity. However, rounding up the authorized balance quantity may result in customers exceeding their credit limits.

You configure whether to round up the fractional value of the impacted balance quantity as a systemwide setting. ECE authorizes an additional RUM unit when the quantity is a fraction.

To configure rounding when authorizing multiple RUMs:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole.
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. 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**.
6. Save your changes.

> **Note**
>
> When discounting custom events, you may encounter the following ECE exception:
>
> ```
> RatingMessageBundle-6000: The rate plan failed to evaluate. The cause
> is probably due to a misconfiguration in the rate plan graph.
> ```
>
> This may occur because the discount offer contains two rating periods of different RUMs and RUM units, and the total quantity cannot be calculated across them. In this scenario, ensure that you have a configuration to filter out the RUM and pass in the rating periods of the same RUM into the trigger. To discount both RUMs, use the filter to create separate configurations for each RUM.

Copyright © 2019, 2026, Oracle and/or its affiliates.

18

Managing Granted Allowances

You can configure how Oracle Communications Elastic Charging Engine (ECE) manages allowances granted to customers.

Topics in this chapter:

- Processing Granted Allowances Before Applying Usage Charges
- About Automatically Renewing Granted Allowances
- About Allowance Renewals
- Configuring Automatic Renewals

Processing Granted Allowances Before Applying Usage Charges

ECE processes a customer's granted allowances and usage charges in the order they appear in the product offerings defined in PDC. For example, assume you configure a product offering in Pricing Design Center (PDC) that grants 5 GB of free data per month and then charges $5 per additional GB of data. If a customer purchases the product offering and uses 8 GB of data in a month, ECE would first consume the 5 GB of free data and then apply a $15 charge for the remaining 3 GB of data usage.

To configure charges in PDC to process granted allowances before applying usage charges, see "Configuring Pricing to Consume Granted Allowances Before Charging" in *PDC Creating Product Offerings*.

> **Note**
>
> ECE handles grant values as LONG values instead of INTEGER values. This prevents overflow when ECE processes large granted allowance values.

About Automatically Renewing Granted Allowances

A renewable offer grants specific resource for a fixed period. For example, it could grant a data allowance for one month. Once the resource is fully consumed, it automatically renews according to predefined settings, if automatic renewals are enabled in PDC.

For example, you can create an auto-renewal offer that grants 2 GB of data for $10, which can automatically renew up to two times in a single day of usage. In this case, the data would be charged as follows:

- The customer pays $10 and receives the first 2 GB of data.
- After consuming the first 2 GB, the customer is charged another $10 and receives an additional 2 GB.
- After consuming the second 2 GB, the customer is charged a further $10 and receives the final 2 GB.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- If the customer consumes all 6 GB, no additional data is granted, and the customer is not charged further, as the maximum number of renewals has been reached.

For more information, see "Automatically Renewing Granted Allowances" in *PDC Creating Product Offerings*.

> **Note**
>
> When you create a pricing tier in PDC, ensure that each tier is set to renew automatically. See "Automatically Renewing Granted Allowances" in *PDC Online Help* and "Configuring the Automatic Renewal of Allowances" in *PDC Creating Product Offerings* for more information.

About Allowance Renewals

The way ECE processes automatically renewable offers differs between online and offline charging. See the following:

- About Allowance Renewals During Online Charging
- About Allowance Renewals During Offline Charging

About Allowance Renewals During Online Charging

ECE manages the automatic renewal of granted allowances during the authorization process for online charging as follows:

1. PDC shares the validity of **renewalResource** and replicates it across other impacts for renewable tiers.
2. When a customer purchases a package that includes an automatically renewable allowance, ECE creates a tracker balance for the current cycle. This tracker balance includes:
   - **The Consumption amount**: The volume the user must consume before receiving a new allowance, such as 1 GB.
   - **The Renewal tracker amount**: The number of times the allowance can be granted to the customer, such as three times.
3. During the charging process, ECE checks if the following conditions apply:
   - The allowance is configured for automatic renewal.
   - The customer consumed the granted amount during the session.
   - The allowance can be granted to the customer again.
4. If all of these conditions are met, ECE automatically renews the granted allowance for the customer.

> **Note**
>
> If the consumption amount is exceeded at the end of a session, ECE does not grant a new allowance within that session. Instead, the new allowance is granted at the start of the next session.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To configure ECE to perform automatic renewals during online charging:

1. Configure ECE to automatically grant allowances. See “Configuring Granted Allowances”.
2. (Optional) Configure ECE to generate notifications when automatic renewals occur. See “Enabling Notifications for Automatically Renewed Granted Allowances”.

About Allowance Renewals During Offline Charging

Oracle Communications Offline Mediation Controller processes CDRs from your network devices, converts them, and sends them as usage requests to ECE for rating. ECE then manages automatic renewals of granted allowances during usage rating as follows:

1. ECE determines the rating mode in the received usage request. If it is offline, ECE proceeds to the next step.
2. ECE returns the original quantity to rate and grant Renewal Free Units (RFUs). If the total amount consumed is greater than the granted allowance, ECE renews the allowance.
3. ECE sets the new allowance's validity start date to the **requestStartTime** and leaves the end time unchanged. The **requestStartTime** is pulled from the **usageRequest**. For example, if the usage request was received on April 30 at 15:00, the allowance's validity start time is set to 2025-04-30T15:00:00.
4. ECE generates the rated output, generating two **TERMINATE** events: one for the grant (and any associated charge) and one for the consumption. ECE only splits the rated events if it has granted any renewal, recurring, or non-counter grant when processing an offline usage request.
5. ECE completes rating and applies charges using the renewed balances.

> **Note**
>
> - Out-of-sequence CDRs can trigger duplicate reload attempts or fallback to PayGo, which can cause duplicate reloads or incorrect charging. To prevent this, implement ordering and ingestion controls.
> - You cannot re-rate events for offline allowance renewals.

Configuring Automatic Renewals

Use the following process to set up automated renewals:

1. (Offline only) Configure Offline Mediation Controller to automatically renew granted allowances. See "Setting Up Automatic Renewals for Offline Charging".
2. Configure ECE to automatically grant allowances. See “Configuring Granted Allowances”.
3. (Optional) Configure ECE to generate notifications when automatic renewals occur. See “Enabling Notifications for Automatically Renewed Granted Allowances”.

Setting Up Automatic Renewals for Offline Charging

To set up your system to automatically renew granted allowances during offline charging, in Offline Mediation Controller, indicate the CDR is for offline charging by setting **REQUEST_MODE_TYPE** to **OFFLINE**. in your NPL rule file.

Copyright © 2019, 2026, Oracle and/or its affiliates.

You do not have to set up configurations to automatically renew granted allowances during online charging.

Configuring Granted Allowances

To configure ECE to automatically renew granted allowances for online and offline charging, set the following ECE Configuration MBeans. You can set these attributes using the **charging-settings.xml** file, a JMX editor, or your **override-values.yaml** file for **oc-cn-ece-helm-chart**.

- **nonLinearRating = true**
- **tariffTimeChangeSupported = true**
- **checkUsedUnitsAfterFUI = false**
- **enableFuiForInitiate = true**
- **alignRecurringImpactsToOffer = true**
- **reverseRateUseAllBalances = false**
- **failBalanceUpdatesOnCreditCeilingBreach = true**
- **remainingBalanceCalcMode = CURRENT_BALANCE**
- **grantRenewalDuringAuthorization = true**
- **rolloverConsumptionRule = ALWAYS_FIRST**
- **saveNonCounterGrantDuringOngoingSession = true**
- **rateZeroQuantity = false**

Enabling Notifications for Automatically Renewed Granted Allowances

You can configure ECE to send notifications when an allowance is automatically renewed or fully consumed. This keeps downstream systems updated in real time. For example, ECE may send a notification such as "Renewal 3 of 5 triggered" with relevant details. When all renewals are used, another notification is sent summarizing the total renewals consumed.

To enable these notifications for both online and offline charging, set the following ECE Configuration MBeans using the **charging-settings.xml** file, a JMX editor, or your **override-values.yaml** file for **oc-cn-ece-helm-chart**:

- Set **renewalGrantActivationNotificationMode** to **ASYNCHRONOUS**. See "Enabling Specific Notification Types".
- Set **enableFuiForInitiate** to **true**.
- By default, the notification includes the message "Reload x out of y", where *x* is the number of reloads that have been granted and *y* is the maximum number of renewals allowed. You can customize the message included in the notification by using the ECE post-charging extension. You can use the *ECE_home***/ocecesdk/source/oracle/communication/brm/charging/sdk/extensions/SamplePostChargingExtension** ECE SDK sample program as a starting point. For more information, see "Post-Charging Extension – Modifying the Renewal Message".
  The following shows a sample renewable grant notification:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification version="1.0.0.0.0">
   <RenewalGrantActivationNotification>
      <NotificationType>RENEWAL_GRANT_ACTIVATION_NOTIFICATION_EVENT</
NotificationType>
      <BalanceId>12345678</BalanceId>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
      <BalanceElementCode>RenewalResource</BalanceElementCode>
      <ActiveReservation>20.00</ActiveReservation>
      <ConsumedReservation>0</ConsumedReservation>
      <CurrentBalance>-40.00</CurrentBalance>
      <OriginalNumberOfReloadAvailable>5.0</OriginalNumberOfReloadAvailable>
      <CurrentNumberOfReloadAvailable>4.00</CurrentNumberOfReloadAvailable>
      <ValidFrom>2015-01-05T18:34:00Z</ValidFrom>
      <ValidTo>2015-02-01T08:00:00Z</ValidTo>
      <TotalGrantedAmount>-40.00</TotalGrantedAmount>
      <ReloadCharge>10.00</ReloadCharge>
      <DateOfGrant>2015-01-05T10:34-08:00</DateOfGrant>
      <OfferName>Renewable_Bundle_1</OfferName>
      <OfferId>3a914eeb-f330-4dcb-881f-12e351bfd8d3</OfferId>
      <PublicUserIdentity>123456789012</PublicUserIdentity>
      <ExtensionData>
         <Message>Reload 1.00 out of 5.0</Message>
      </ExtensionData>
   </RenewalGrantActivationNotification>
</Notification>
```

In the sample notification, **<OriginalNumberOfReloadAvailable>** is the maximum number of renewals allowed and **<CurrentNumberOfReloadAvailable>** is the number of renewals remaining for the customer.

Copyright © 2019, 2026, Oracle and/or its affiliates.

19

Managing Charging During System Downtimes

You can configure how Oracle Communications Elastic Charging Engine (ECE) handles charging when the system is down.

Topics in this document:

- Configuring Diameter Gateway to Bypass Rating During ECE Downtime
- Configuring How ECE Manages Active Sessions When Network Elements Fail

Configuring Diameter Gateway to Bypass Rating During ECE Downtime

During a planned maintenance activity or in the event of unplanned downtime of an ECE node, you can configure Diameter Gateway to continue receiving CCRs and responding to the service network without real-time rating of the CCRs.

When you configure Diameter Gateway to bypass rating, it persists the Diameter CCRs to the Oracle NoSQL database. Later, when ECE nodes are restored, you can replay the persisted CCRs to the ECE charging servers for rating and updating balance impacts. With this functionality, services can be delivered to your subscribers without any interruption.

To configure Diameter Gateway to bypass rating, perform the following procedures:

1. Enable the **ocsBypassExtension** charging extension.
    a. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
    b. Expand the **ECE Configuration** node.
    c. Expand **charging.extensions**.
    d. Expand **Attributes**.
    e. Specify the fully qualified class name of the extension for the **ocsBypassExtension** attribute.
2. Start the persistence of the diameter messages. See "Managing the Persistence of Usage Requests During ECE Downtime".
3. After the ECE nodes are restored, use Diameter Replayer to replay the CCRs to ECE for rating and updating balance impacts. See "Replaying Persisted Requests into ECE".

Managing the Persistence of Usage Requests During ECE Downtime

When Diameter Gateway is configured to bypass rating during a planned maintenance activity or in an unplanned downtime of ECE, Diameter Gateway receives CCRs and persists them to the Oracle NoSQL Database. Later, when the ECE nodes are restarted, you stop the bypass rating extension and replay the persisted messages to the ECE charging servers for rating and updating balance impacts.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Managing the persistence of diameter requests involves starting and stopping the persistence of the requests. Before persisting the incoming diameter requests, ensure that the **ocsBypassExtension** charging extension is enabled. See "Configuring Diameter Gateway to Bypass Rating During ECE Downtime" for information on configuring Diameter Gateway to bypass rating.

For planned maintenance activities, you start the persistence of usage requests before the ECE nodes become unavailable. During an outage, the persistence of diameter requests starts only if the **ocsBypassExtension** charging extension is enabled. If only the extension is enabled and bypass rating is not started, the usage requests do not flow to the ECE Charging Server nodes. In such a scenario, Diameter Gateway returns Error 5012 for the requests.

> **Note**
>
> The bypass rating functionality is supported for Gy diameter messages only.

To persist incoming diameter requests:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **DiameterGateway** node.
3. Expand **BFTTask**.
4. Expand **Operations**.
5. Click **startByPass**.

   The Diameter Gateway starts persisting the incoming requests.

In planned maintenance activities or in unplanned downtime, once the ECE nodes become available and start running, you must stop the persistence of the requests. Otherwise, messages are persisted even after the ECE nodes are up and running, which results in a large volume of requests that need to be replayed.

To stop persisting the requests, click **stopByPass**. Before doing this, you can check if the persistence of requests is running by clicking **BFTRunning**. If bypassing rating is enabled, this field shows **True**. Otherwise, it shows **False**.

Replaying Persisted Requests into ECE

When you configure Diameter Gateway to bypass rating during planned maintenance or unplanned downtime of ECE, Diameter Gateway persists the incoming CCRs in the Oracle NoSQL Database. Later, during a non-peak period, you replay the persisted CCRs to the ECE charging server when the ECE server is restored and is ready to process real-time requests. When you replay the persisted CCRs, the requests are passed to the ECE charging server, which then rates the CCRs and updates the balance impacts. You can plan when to start replaying the persisted messages, considering that replaying them can have performance impacts while real-time requests are also being processed. Before replaying the persisted requests, ensure that bypass rating extension is disabled and the ECE nodes are up and running. To check if bypass rating is stopped, click **BFTRunning** in ECE MBeans.

To replay the persisted requests into ECE:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **DiameterGateway** node.

Copyright © 2019, 2026, Oracle and/or its affiliates.

3. Expand **BFTTask**.
4. Expand **Operations**.
5. Click **persistedMessageCount** to view the number of requests that have been persisted and are ready for replay.
6. Click **startBFTReplayer**.

   The replayer starts replaying the persisted messages to the ECE charging servers. As messages are replayed to the servers, the number of **persistedMessageCount** keeps decreasing until it becomes **0**.

   The **replayedMessageCount** field shows the number of requests that are being replayed. As requests are replayed, the number of **replayedMessageCount** keeps increasing until it matches the initial count of **persistedMessageCount**.

You stop replaying the messages once all the persisted messages are replayed into the ECE charging servers.

To stop replaying the persisted messages:

1. Check the status of the replayer by clicking **BFTReplayerRunning**. Ensure that this field shows **Running**.
2. Check the status of the **replayedMessageCount** field. Ensure that this field shows **0**, which indicates that all the persisted messages are replayed into the ECE charging servers.
   - **replayedMessageCount** indicates the number of Diameter messages that are replayed to the ECE charging server by the current instance of Diameter Replayer.
   - **persistedMessageCount** indicates the number of Diameter messages that have been persisted to the Oracle NoSQL database, but are yet to be replayed.
3. Click **stopBFTReplayer**.

   Replaying of messages is stopped.

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 in a JMX editor, such as JConsole. See "Accessing 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 with an Initiated state are terminated when an accounting on/off request is processed.
   - **CANCEL**: Active sessions with an Initiated state are canceled when an accounting on/off request is processed.

Copyright © 2019, 2026, Oracle and/or its affiliates.

20

Using Impact-Based Candidate Selection

Learn how Oracle Communications Elastic Charging Engine (ECE) selects candidates for charge selectors and generic selectors when configured for impact-based selection.

Topics in this document:

- How Impact-Based Candidate Selection Works
- Selecting Candidates Across Rating Periods

How Impact-Based Candidate Selection Works

During ECE rating, charge selectors and generic selectors normally evaluate matching rules or result branches in configured order and select the first candidate that matches. You can instead use impact-based candidate selection to evaluate all matching candidates and select the one whose rating impact is most appropriate for a configured currency or noncurrency balance element.

Use impact-based candidate selection when the configured rate alone does not show the actual effect of rating. For example, candidate impacts can differ because rating consumes free units, crosses tiers, or produces multiple balance impacts. This feature applies only to charge selectors and generic selectors during ECE rating.

You configure the candidate selection mode and candidate selection balance element in PDC. See "Impact-Based Candidate Selection" in *PDC Creating Product Offerings*.

The following candidate selection modes are supported:

- **FIRST_MATCH:** Selects the first matching candidate in configured order. This is the default.
- **MIN_IMPACT:** Evaluates all matching candidates and selects the candidate with the lowest signed immediate impact to the configured balance element.
- **MAX_IMPACT:** Evaluates all matching candidates and selects the candidate with the highest signed immediate impact to the configured balance element.

Existing selectors that do not specify **CandidateSelectionMode** continue to use **FIRST_MATCH**. Existing generic selectors that do not specify candidate-selection fields continue to use the first matching result branch.

If you use a DSL selector extension, see "Customizing a DSL Selector Extension for Impact-Based Selection" in *ECE Implementing Charging*.

Selecting Candidates Across Rating Periods

For **MIN_IMPACT** and **MAX_IMPACT**, ECE uses the same rating scope for candidate evaluation and final rating. With total-period rating, ECE evaluates each candidate across the full usage period, including a period that crosses a time-of-day tariff boundary. With split-cut rating, ECE evaluates candidates separately for each rating cut. As a result, different cuts can select different candidates when the rate, tier, or balance impact changes.

Copyright © 2019, 2026, Oracle and/or its affiliates.

For example, an event runs from 09:55 to 10:05 and the tariff changes at 10:00. Candidate A produces impacts of -0.20 and -0.80, totaling -1.00. Candidate B produces impacts of -0.40 and -0.10, totaling -0.50. With total-period rating and **MIN_IMPACT**, Candidate A is selected because -1.00 is lower than -0.50. If ECE splits the event at 10:00, Candidate B can be selected for the first cut and Candidate A for the second cut.

ECE uses that same scope when it simulates and compares candidates, selects a candidate, reevaluates the selected branch, and applies the final charge.

Selecting a Charge Selector Candidate

For **MIN_IMPACT** and **MAX_IMPACT**, ECE evaluates all matching charge selector rules in configured order. It evaluates each candidate separately without updating subscriber balances.

ECE creates an overlap group when two or more matching candidates can rate the same portion of a request for the same purchased offering, customer, and balance scope. Candidates in different purchased offerings do not compete because they belong to different overlap groups. Within each group, ECE compares the qualifying candidates, retains one winner, removes the other candidates from that group, reevaluates the applicable RUMs for the winner, and then performs final rating. Candidates outside an overlap group continue through normal rating.

A candidate qualifies only when at least one of its rate plans declares an impact to **CandidateSelectionBalanceElementCode** and the candidate produces an immediate impact to that balance element during evaluation. If no candidate in an overlap group qualifies, the request follows the existing runtime failure path. ECE does not select a default candidate, return a zero-rate result, or apply a charge selector-specific fallback.

When charge selector candidates produce the same comparable impact, a fully rated candidate is selected over a partially rated candidate. If the tie remains, the candidate associated with the earlier matching charge selector rule is selected.

Selecting a Generic Selector Candidate

Impact-based candidate selection for a generic selector occurs within a single Charge Rate Plan and does not create offer-level overlap groups. For **MIN_IMPACT** and **MAX_IMPACT**, ECE evaluates each matching `results` branch separately and compares its immediate impact to **CandidateSelectionBalanceElementCode**.

Before comparison, ECE adds all fixed and scaled immediate impacts to the configured balance element. If a matching branch does not produce an impact to that balance element, ECE uses an impact of 0. Impacts to other balance elements do not affect selection, but they still apply if ECE selects the branch. When a request uses multiple RUM paths, ECE performs selection separately within each applicable generic selector path.

After selecting a branch, ECE reevaluates only the selected branch using the original rating input and then continues with normal rating. The comparison does not update subscriber balances. If multiple branches have the same comparable impact, ECE selects the first matching `results` branch. If no branch matches, the Charge Rate Plan follows the existing no-applicable-rating path. Impact-based candidate selection does not add a default branch or zero-rate fallback.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part V

Integrating with External Systems

This part provides information about integrating external applications with Oracle Communications Elastic Charging Engine (ECE). It contains the following chapters:

- Creating Kafka Topics for ECE Integration
- Connecting ECE to a 5G Client
- Generating CDRs for External Systems
- Connecting ECE to a Diameter Client
- Connecting ECE to a RADIUS Client
- Configuring Policy-Driven Charging

Copyright © 2019, 2026, Oracle and/or its affiliates.

21

Creating Kafka Topics for ECE Integration

You create Apache Kafka topics so Oracle Communications Elastic Charging Engine (ECE) can support sending Notifications with HTTP Gateway, Diameter Gateway, RADIUS Gateway and BRM Gateway.

Topics in this document:

- Setting Up ECE and Gateways for Apache Kafka Using Topic-Based Routing
- Migrating Existing ECE Systems Gateways to Apache Kafka Using Topic-Based Routing
- Setting Up ECE and Gateways for Apache Kafka Using Partition-Based Routing
- Connecting ECE and Gateways to Kafka Topics
- Recording Failed ECE Usage Requests
- Understanding and Configuring ECE Prepaid Usage Overage

Setting Up ECE and Gateways for Apache Kafka Using Topic-Based Routing

You can use the enhanced Kafka integration in Oracle Communications Billing and Revenue Management (BRM) for improved real-time data processing, particularly in managing high data volumes. A dedicated message processor ensures efficient and reliable system functionality.

> **Note**
>
> This is the recommended configuration for Kafka integration.

The message processor handles data in a controlled and sequential manner. Its checkpointing mechanism enables reliable recovery and uninterrupted data processing during system interruptions. If a gateway goes out of service, the system automatically redirects its partitions to another gateway without manual intervention. When the original gateway returns, traffic is automatically rerouted back.

To set up the Apache Kafka architecture using topic-based routing, follow these steps:

> **Note**
>
> For new multisite deployments, install and configure all sites at the same time, since there is no traffic in the system yet.

1. Set the **kafkaCustomPartitionAssignorEnabled** ECE configuration MBean to **true**. See "Enabling the Topic-Based Kafka Architecture".

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Wait until all ECS instances or pods have been restarted and any rebalancing has completed.
3. Perform a rolling restart of the Diameter Gateway, RADIUS Gateway, HTTP Gateway, EM Gateway, and BRM Gateway so that the consumers are restarted, and they subscribe to the new Kafka topics.
4. On multisite deployments, repeat steps 2 through 4 until all sites have been configured to use the Kafka topic-based routing feature.
5. On multisite deployments, restart the **ReplicatedFederatedCache** service between the sites in the Coherence cluster and verify that the ECE Configuration **AppConfiguration** MBeans in all the sites are in sync.

Enabling the Topic-Based Kafka Architecture

By default, ECE uses the Apache Kafka partition-based architecture to communicate with network clients, but you can configure ECE to use the topic-based architecture instead.

To enable the topic-based Kafka architecture without restarting ECE:

1. Do one of the following based on your system type:
   - **Cloud native systems**: Connect to a JMX editor, such as JConsole. See "JMX Connection to ECE Using JConsole" in *BRM Cloud Native System Administrator's Guide*.
   - **On-premises systems**: Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.kafkaConfigurations**.
4. Expand **Attributes**.
5. Set the **kafkaCustomPartitionAssignorEnabled** attribute to **true**.

Alternatively, you can use the **charging-settings.xml** file or your **override-values.yaml** file for **oc-cn-ece-helm-chart** to set the **kafkaCustomPartitionAssignorEnabled** attribute. If you make the change by updating the file, you must perform a rolling restart of your ECE application instances or pods for the change to take effect.

Generating Kafka Topics with Topic-Based Routing

You use the **create_all_kafka_topics.sh** script to create the required Kafka topics. When running the script, use the Kafka broker details for all sites.

To generate Kafka topics with topic-based routing:

1. Ensure you have installed the Apache Kafka command-line tools, such as **kafka-topics.sh**, and that their location is in the PATH environment variable or the KAFKA_HOME and KAFKA_BIN_DIR environment variables.
2. Do one of the following based on your system type:
   - **Cloud native systems**: Create an **override-values.yaml** file for **oc-cn-ece-helm-chart** and configure the keys under **kafkaConfigurationList** to meet your system requirements.
   - **On-premises systems**: In your **charging-settings.xml** file, configure the **kafkaConfigurationList** entries to meet your system requirements.

Copyright © 2019, 2026, Oracle and/or its affiliates.

3. In your configuration file, configure the Kafka brokers to allow automatic topic creation. Set the default number of partitions for new topics to either 20 or the product of the number of Kafka broker instances and the number of gateways that will consume notifications for a topic, whichever is greater.

   For example, if you have 3 Kafka brokers and 8 Diameter Gateways, set the default to 24 (3 x 8) partitions. This configuration ensures that there are more partitions than consumers, allowing each consumer to have its own partition and distributing partitions evenly across Kafka brokers for optimal resource utilization.

   > **Note**
   >
   > For multisite deployments, the configuration file should have the details for all sites. This ensures that the **create_all_kafka_topics.sh** script creates all necessary topics for all Kafka brokers across all sites, supporting automatic high availability and disaster recovery.

4. Run the **create_all_kafka_topics.sh** script:

   ```
   create_all_kafka_topics.sh overrideValuesFile --bootstrap-server kafkaHost:port --
   partitions numPartitions [--ssl enabled | disabled] [--truststore-location
   trustStorePath ] [--replication-factor replicationNum]
   ```

   where:

   - *overrideValuesFile* specifies the name and path to the **override-values.yaml** file you created in step 2.
   - *kafkaHost* and *port* specify the host and port that the Kafka client will connect to in a bootstrap Kafka cluster the first time it starts.
   - *numPartitions* specifies the number of partitions to create in the topic.
   - *trustStorePath* specifies the path to the TrustStore file. This option is required only if SSL is enabled.
   - *replicationNum* specifies how many copies of each topic's partition are stored across different brokers in the Kafka cluster. The default is **1**, which disables the replication.

Enter the *trustStorePassword* when prompted by the script, where *trustStorePassword* is the password for the TrustStore. This behavior applies when SSL is enabled.

Migrating Existing ECE Systems Gateways to Apache Kafka Using Topic-Based Routing

To migrate from the existing integration architecture to the Apache Kafka topic-based routing:

1. For multisite deployments, stop the federation between the site about to be migrated and the other sites.
2. Set the **kafkaCustomPartitionAssignorEnabled** ECE configuration MBean to **true**. See "Enabling the Topic-Based Kafka Architecture".

   This updates the **charging-settings.xml** file in the cache and updates the ECE **charging-settings-namespace** ConfigMap.
3. During the ECE postinstallation process, run the **create_all_kafka_topics.sh** script to create the required Kafka topics. See "Generating Kafka Topics with Topic-Based Routing".

Copyright © 2019, 2026, Oracle and/or its affiliates.

Wait until all ECS instances or pods have been restarted and any rebalancing has completed.

4. Drain the remaining notifications from the Kafka brokers at the site being migrated. Using the existing site's gateways, any remaining notifications in the old **EceNotifications** topics at the Kafka broker will be consumed.

   Run this step until all notifications on the old topics are consumed. Wait until the old topics have been drained of all remaining notifications so that the consumers are restarted and subscribe to the new Kafka topics.

5. Perform a rolling restart of the Diameter Gateway, RADIUS Gateway, HTTP Gateway, EM Gateway, and BRM Gateway so that the consumers are restarted, and they subscribe to the new Kafka topics.

> **Note**
>
> Ensure that BRM is pointed to another site's EM Gateways prior to bringing down the EM Gateway at the site being migrated.

6. On multisite deployments, repeat 2 through 6 for all the sites until they have been migrated to using the new Kafka redesign feature.

7. Restart the configuration federation of the **ReplicatedFederatedCache** service between the sites in the Coherence cluster and verify that the ECE Configuration **AppConfiguration** MBeans in all the sites are in sync.

> **Note**
>
> For ECS pods, run the helm upgrade with the job to update app configuration enabled.

Setting Up ECE and Gateways for Apache Kafka Using Partition-Based Routing

> **Note**
>
> This configuration is recommended for backward compatibility only.

To set up ECE with your Kafka topics:

1. When you install ECE:
   - Specify to use Apache Kafka topics and enter the details for your ECE notification topics, Suspense topic, ECE failure topic, and ECE overage topic.
   - (HTTP Gateway Only) Enable Network Repository Function (NRF) registration in one of your HTTP Gateway servers.
   - (BRM Gateway Only) Specify the details for connecting to the BRM Gateway.

   For more information, see "Installing Elastic Charging Engine" in *ECE Installation Guide*.

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. During the ECE postinstallation process:
   - Run the **kafka_post_install.sh** script to create your ECE notification topics, Suspense topic, ECE failure topic, and ECE overage topic. See "Creating Kafka Topics for ECE" in *ECE Installation Guide*.
   - Run the **post_Install.sh** script and choose to create only the Acknowledgment queue. See "Creating WebLogic JMS Queues for BRM" in *ECE Installation Guide*.

   > **Note**
   >
   > It is strongly recommended that you use the **kafka_post_install.sh** script to create all ECE notification topics in advance instead of relying on topic auto-creation. This ensures that ECE notification consumers will be able to subscribe to topics even before the first notification message is produced.
   >
   > All ECE consumer groups are prefixed with Ece. Your consumers should never use a consumer group that starts with this prefix to avoid collisions.

3. Configure Kafka partition assignments for gateways. See "Configuring Kafka Partition Assignments for Gateways".
4. (Optional) Configure ECE to send information about failed usage requests to the ECE failure topic. See "Recording Failed ECE Usage Requests".
5. (Optional) Configure ECE to generate CDRs for any prepaid usage overage and send them to the ECE overage topic. See "Understanding and Configuring ECE Prepaid Usage Overage".

Configuring Kafka Partition Assignments for Gateways

Configure Kafka partition assignments in ECE AppConfiguration; do not configure them in **kafka_post_install.sh**. For cloud native deployments, set **kafkaPartition** and **kafkaPartitionForInternalExternal** in your **override-values.yaml** file for **oc-cn-ece-helm-chart**. For on-premises deployments, set the attributes in the ECE Configuration MBeans by using a JMX editor such as JConsole.

To configure Kafka partition assignments for gateways:

1. For each Diameter Gateway and BRM Gateway configuration, set **kafkaPartition** to one dedicated Kafka partition ID.
2. For each HTTP Gateway configuration, set **kafkaPartition** to a comma-separated list of dedicated Kafka partition IDs.

   Configure at least one partition ID for each HTTP Gateway replica that might run.

   For example, if **httpgw1** is configured with **1**,**2**,**3**, **httpgw1-0** consumes notifications from partition **1**, **httpgw1-1** consumes notifications from partition **2**, and **httpgw1-2** consumes notifications from partition **3**.
3. Set **kafkaPartitionForInternalExternal** to the dedicated partition ID for notifications configured for internal and external publishing.
4. Ensure that every partition ID specified by **kafkaPartition** or **kafkaPartitionForInternalExternal** is unique across the ECE deployment.

   Also ensure that the ECE notification topic contains enough partitions for all of the partition IDs that you assign. See "Connecting ECE and Gateways to Kafka Topics."
5. Do one of the following:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- For cloud native deployments, after you update the Kafka partition application configuration values, run the **chargingConfigurationReloader** job to reload the application configuration. See "Reloading ECE Application Configuration ChangesReloading ECE Application Configuration Changes" in *BRM Cloud Native System Administrator's Guide*.
- For on-premises deployments, update the **kafkaPartition** and **kafkaPartitionForInternalExternal** attributes by using a JMX editor such as JConsole.

Determining the Number of Partitions Required

Use this procedure only for Kafka partition-based routing. Determine the minimum number of partitions in the ECE notification topic before you create the topic and assign partition IDs to gateways.

To determine the minimum number of partitions:

1. Identify the maximum number of dedicated gateway notification consumers that might run in the deployment:
   - Diameter Gateway instances
   - HTTP Gateway replicas across all HTTP Gateway configurations
   - BRM Gateway instances
   - One consumer for notifications configured by **kafkaPartitionForInternalExternal**
2. If you assign consecutive partition IDs starting with 0, calculate the minimum number of partitions by using this formula:

   [(Max Diameter Gateway instances) + (Max HTTP Gateway replicas) + (Max BRM Gateway instances) + (1 for kafkaPartitionForInternalExternal)]

   For example, if you have 4 Diameter Gateway instances, 3 HTTP Gateway replicas, and 1 BRM Gateway instance, create at least [(4 + 3 + 1 + 1) = 9] partitions. This example assumes that the partition IDs are 0 through 8.
3. Compare the calculated minimum with the highest partition ID that you plan to assign. The number of partitions must be greater than the highest partition ID configured in **kafkaPartition** or **kafkaPartitionForInternalExternal**.
4. Configure the **partitions** value to the larger of the calculated minimum and the highest partition ID plus 1.

> **Note**
>
> If the ECE External Notification topic is enabled, configure its partition count separately by using **externalPartitions**. You can set **partitions** higher than the minimum to allow future gateway partition IDs without changing the Kafka broker.

Connecting ECE and Gateways to Kafka Topics

You connect ECE to the following Kafka topics so that ECE can publish notifications, failed usage requests, and CDRs with usage overage information to them.

- **ECE notification topics**: Stores notifications from ECE.
- **Suspense topic**: Stores failed notifications from ECE.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **ECE failure topic**: (HTTP Gateway and Diameter Gateway only) Stores details about failed ECE usage requests, such as the user ID and request payload.
- **ECE overage topic**: (HTTP Gateway and Diameter Gateway only) Stores overage records, which contain details about usage overage amounts for prepaid customers.

You can also connect the HTTP Gateway, Diameter Gateway, and RADIUS Gateway to these Kafka topics so they can retrieve notifications, failed usage requests, and CDRs with usage overage information from them.

> **Note**
>
> If you are using an ECE system with a topic-based Kafka architecture, ECE splits the ECE notifications topic into multiple granular topics based on factors such as cluster, gateway, notification type, and client.

ECE creates each notification topic using this naming convention:

`Notify.`*`clusterName`*`.`*`consumerGroup`*`[.`*`notificationName`*`][.`*`networkClientId`*`][.`*`eceGatewayId`*`]`

where:

- *clusterName* is the string that represents the site name from your **charging-settings.xml** file. The allowable values are BRM1, BRM2, ECE1, ECE2, and so on.
- *consumerGroup* is the name of the consumer group. The possible values are **DiamNetwork**, **RadNetwork**, **ChfNetwork**, **Brm**, **Ext**, **Suspense**, and **Overage**.
- *notificationName* is the notification event type. This is included for **DiamNetwork**, **RadNetwork**, **ChfNetwork**, **Brm**, and **Ext** consumer groups.
- *networkClientId* is used to differentiate the CHF Network (for the HTTP Gateway) from the Diameter Client (for the Diameter Gateway). This is included for **DiamNetwork** and **ChfNetwork** consumer groups.
- *eceGatewayId* is the value taken from the Diameter Gateway ID. This is included for **DiamNetwork** consumer groups.

To connect ECE and gateways to your Kafka topics:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.kafkaConfigurations**.
4. Expand **Attributes**.
5. Specify the Kafka configuration values for the attributes in Table 21-1.

Table 21-1 kafkaConfiguration Properties

Property Name	Description
name	The name of your ECE cluster.
hostname	The host name and port number of the machine in which Apache Kafka is installed. If it contains multiple Kafka brokers, create a comma-separated list.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 21-1 (Cont.) kafkaConfiguration Properties

Property Name	Description
topicName	The name of the Kafka topic where ECE will publish notifications. **Note:** This is for partition-based routing only.
suspenseTopicName	The name of the Kafka topic where failed notifications are published. **Note:** This is for partition-based routing only.
failureTopicName	The name of the Kafka topic where ECE will publish details about failed usage requests. **Note:** This is for partition-based routing only.
overageTopicName	The name of the Kafka topic where ECE will publish overage records with information about your prepaid customer's usage overage during online sessions. **Note:** This is for partition-based routing only.
partitions	The number of partitions in the ECE notification topic for Kafka partition-based routing. Configure enough partitions for all gateway notification consumers that use dedicated partitions, plus the partition configured by **kafkaPartitionForInternalExternal**. For the calculation, examples, and validation rule, see "Determining the Number of Partitions Required." **Note:** Note: This is for partition-based routing only.
externalPartitions	The total number of Kafka partitions in the ECE External Notification topic. **Note:** This is for partition-based routing only.
externalTopicEnabled	The parameter that specifies, whether ECE External Notification topic is created and in use. Set to **true** to enable.
externalTopicName	The name of the ECE External Notification topic. **Note:** This is for partition-based routing only.
failurePartitions	The number of Kafka partitions in your ECE failure topic. **Note:** This is for partition-based routing only.
kafkaCustomPartitionAssignorEnabled	The parameter to enable topic-based routing for Kafka notifications which uses a custom partition assignor for automated support for high availability and disaster recovery. If this parameter is set to **false**, the partition-based routing will be used.
kafkaProducerReconnectionInterval	The amount of time, in milliseconds, the ECE Notification Publisher waits before attempting to reconnect to a Kafka topic.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 21-1 (Cont.) kafkaConfiguration Properties

Property Name	Description
kafkaProducerReconnectionMax	The maximum amount of time, in milliseconds, the ECE Notification Publisher waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaProducerReconnectionInterval** will increase exponentially for each consecutive connection failure, up to this maximum.
kafkaHTTPReconnectionInterval	The amount of time, in milliseconds, HTTP Gateway waits before attempting to reconnect to the Kafka topic.
kafkaHTTPReconnectionMax	The maximum amount of time, in milliseconds, HTTP Gateway waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaHTTPReconnectionInterval** will increase exponentially for each consecutive connection failure, up to this maximum.
kafkaDGWReconnectionInterval	The amount of time, in milliseconds, Diameter Gateway waits before attempting to reconnect to the Kafka topic.
kafkaDGWReconnectionMax	The maximum amount of time, in milliseconds, Diameter Gateway waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaDGWReconnectionInterval** will increase exponentially for each consecutive connection failure, up to this maximum.
kafkaBRMReconnectionInterval	The amount of time, in milliseconds, BRM Gateway waits before attempting to reconnect to the Kafka topic.
kafkaBRMReconnectionMax	The maximum amount of time, in milliseconds, BRM Gateway waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaBRMReconnectionInterval** will increase exponentially for each consecutive connection failure, up to this maximum.
kafkaFederationIssuesTopicName	The Kafka topic name for federation-related issues.
kafkaFederationIssuesReconnectionInterval	The reconnection interval, in milliseconds, of the Kafka producer to the federation issues topic.
kafkaFederationIssuesReconnectionMax	The maximum reconnection time, in milliseconds, of the Kafka producer to the federation issues topic
kafkaRGWTopicName	The Kafka topic that is used for Radius Gateway notifications.
kafkaRGWReconnectionIntervalMillis	The amount of time, in milliseconds, RADIUS Gateway waits before attempting to reconnect to the Kafka topic.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 21-1 (Cont.) kafkaConfiguration Properties

Property Name	Description
kafkaRGWReconnectionMaxMillis	The maximum amount of time, in milliseconds, RADIUS Gateway waits before attempting to reconnect to a broker that has repeatedly failed to connect. The **kafkaRGWReconnectionIntervalMillis** will increase exponentially for each consecutive connection failure up to this maximum.
walletLocation	The path of the wallet where certificates are stored when connecting to the Kafka brokers using SSL.

Recording Failed ECE Usage Requests

ECE may occasionally fail to process usage requests. For example, a data usage request could fail because a customer has insufficient funds. You can configure ECE to publish details about failed usage requests, such as the user ID and request payload, to the ECE failure topic in your Kafka server. Later on, you can reprocess the usage requests or view the failure details for analysis and reporting.

To enable the recording of failed ECE usage requests:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole.
2. Expand the **ECE Configuration** node.
3. Expand **charging.kafkaConfigurations**.
4. Expand **Attributes**.
5. Set the **persistFailedRequestsToKafkaTopic** property to **true**.

> **Note**
>
> You can also examine the gateway log files to determine why a usage request failed. See "Troubleshooting Failed Usage Requests" in *BRM System Administrator's Guide* for more information.

Understanding and Configuring ECE Prepaid Usage Overage

ECE can generate overage records when prepaid usage exceeds the amount that was granted, reserved, or available for the session. These overage records help account for usage that ECE cannot fully charge during the active session and help prevent revenue leakage.

When the network reports that the number of Used Service Units (USUs) is greater than the number of Granted Service Units (GSUs), ECE treats the excess usage as usage beyond the granted or reserved amount. In this flow, ECE applies built-in credit-protection behavior:

1. ECE charges the customer only up to the available allowance.
2. ECE generates an overage record for the excess usage.
3. ECE sends the overage record to the ECE Overage Kafka topic.

Copyright © 2019, 2026, Oracle and/or its affiliates.

For example, assume that a customer has a prepaid balance of 100 minutes, but uses 130 minutes during a session. If the network reports 130 minutes of usage after only 100 minutes were granted, ECE charges the customer for 100 minutes, creates an overage record for the remaining 30 minutes of usage, and writes the overage record to the ECE Overage Kafka topic.

> **Note**
>
> If your system does not contain a Kafka server, you can configure ECE to publish overage usage amounts to a log file instead. See "Logging Prepaid Usage Overage".

You can also configure ECE to create overage records during post-rating validation by using the **checkReservationOverImpact** attribute. When this attribute is enabled and the rated balance impact exceeds the customer's available balance, ECE applies the balance impact only up to the available balance and creates an overage record for the remaining amount.

Use **checkReservationOverImpact** when you want ECE to capture overage during post-rating validation.

> **Note**
>
> The built-in credit-protection behavior is not controlled by the **checkReservationOverImpact** attribute. Therefore, even when **checkReservationOverImpact** is set to **false**, ECE can still generate overage records when the network reports USUs are greater than GSUs.

To configure ECE to perform post-rating validation for prepaid usage overage:

1. Ensure that you have created an ECE Overage Kafka topic and connected ECE to your Kafka server. See "Connecting ECE and Gateways to Kafka Topics".
2. Enable ECE to perform post-rating validation and capture overage when the rated balance impact exceeds the available balance:
   a. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
   b. Expand the **ECE Configuration** node.
   c. Expand **charging.server**.
   d. Expand **Attributes**.
   e. Set the **checkReservationOverImpact** attribute to **true**. The default is **false**.

You can also customize ECE to include any existing unrated quantity in a CDR or notification using the ECE SDK post-charging extension (**postChargingExtension**). Use the **OverageDetailsImpl** class to include the unrated quantity. The extension can then capture the unrated quantity by using the **getOverageDetailsMap()** method.

Copyright © 2019, 2026, Oracle and/or its affiliates.

22

Connecting ECE to a 5G Client

You can set up 5G network integration for online and offline charging by using the Oracle Communications Elastic Charging Engine (ECE) HTTP Gateway.

Topics in this document:

- About the HTTP Gateway
- Integrating HTTP Gateway with 5G Networks
- Using the ECE REST API

> ⚠ **Caution**
>
> Deploying charging for 5G with HTTP Gateway (5G CHF) requires a cloud native deployment of ECE and BRM components. The HTTP Gateway can be used only on an ECE cloud native system.

About the HTTP Gateway

You integrate 5G clients with ECE by using the HTTP Gateway. The HTTP Gateway sends usage requests to ECE for online and offline charging and then sends responses to the 5G network.

The HTTP Gateway supports the 5G service-based architecture and does the following:

- Receives ECE REST API requests from 5G clients and then translates them into batch request server (BRS) requests.
- Publishes details about inbound ECE REST API requests that failed to the ECE failure Kafka topic.
- Determines whether a usage request requires online or offline charging. See "About Determining the Charging Type" for details.
- Submits BRS requests to the ECE server.
- Receives responses from the ECE server and then translates them into REST API responses.
- Responds to the 5G clients.
- Consumes notifications from the ECE notification topic and then notifies the 5G clients by making a REST call to the URL stored in the system.
- Publishes details about failed notifications to the corresponding Kafka dead-letter queue. For example: Ece.ChfNetwork.RarNotificationEvent.ChfNet001.Dlq.

When configured to do so, the HTTP Gateway can also send ECE REST API requests from 5G clients to the CDR Gateway for generating CDRs. For more information, see "Generating CDRs for External Systems".

Copyright © 2019, 2026, Oracle and/or its affiliates.

About Determining the Charging Type

ECE can perform online charging and offline charging. When HTTP Gateway receives a usage request, it determines whether to route the request to online or offline charging as follows:

- For INITIATE requests, based on the **multipleUnitUsage** block. If the block is present, the request needs online charging. If the block is missing, the request needs offline charging.
- For UPDATE requests, based on the value of the **quotaManagementIndicator** field in the request. If the value is set to **ONLINE_CHARGING**, the request needs online charging. If the field is missing or the value is set to **OFFLINE_CHARGING**, the request needs offline charging.
- For TERMINATE requests, based on the value of the **quotaManagementIndicator** field in the request. If the value is set to **ONLINE_CHARGING**, the request needs online charging. If the field is missing or the value is set to **OFFLINE_CHARGING**, the request needs offline charging.

For more information about these properties, see the Nchf Converged Charging endpoints and Nchf Offline-Only Charging endpoints in *REST API Reference for Elastic Charging Engine*.

About Sending Notifications to HTTP Gateway

You can configure ECE to send the following notifications to the ECE notification topics, where they are retrieved by the HTTP Gateway:

- Spending Limit Notification (SNR): Specifies that a subscriber has reached a spending threshold or limit.
- Reauthorization Request (RAR): Specifies that a reauthorization request is needed.

Configure HTTP Gateway to consume SNR and RAR notifications from the ECE notification topics. When configured to do so, HTTP Gateway listens on the ECE notification topics and filters notifications based on the notification type in the header. Depending on the notification type, HTTP Gateway does the following:

- Retrieves SNR notifications from the Kafka topic and makes a REST API call to **notifUri** to post the notification.
- Retrieves RAR notifications from the Kafka topic and makes a REST API call to post the notification to **notifUri** in the usage request.

Integrating HTTP Gateway with 5G Networks

For information about conformance with industry standards, see *ECE 5G CHF Protocol Implementation Conformance Statement*.

To integrate HTTP Gateway with your 5G network:

1. Create Apache Kafka topics and connect them to ECE. See "Creating Kafka Topics for ECE Integration".
2. Configure the NRF registration details for each HTTP Gateway instance in your system. See "Configuring Registration Details for the HTTP Gateway Server".
3. Configure one or more NF services. See "Configuring NF Services".
4. Configure the HTTP Gateway to send usage requests to ECE Server for convergent charging. See "Configuring HTTP Gateway for Convergent Charging".

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. Define any custom service-event mappings in Pricing Design Center. See "Enabling Charging for Custom Events" in *PDC Creating Product Offerings*.
6. Edit your mediation specification file and load it into ECE. The mediation specification enables the HTTP Gateway to associate each HTTP request with its respective usage-request builder. See "Editing the HTTP Gateway Mediation Specification File".
7. Map nCHF request fields to ECE payload attributes by updating the HTTP request specification files and loading them into ECE. See "Mapping nCHF Request Fields to ECE Payload Attributes".
8. Configure HTTP Gateway to terminate requests that have timed out at the client. See "Checking for Client-Side Timeouts in HTTP Gateway".
9. Configure ECE to send notifications to HTTP Gateway through the ECE notification topics. See "Configuring ECE to Publish Notifications to HTTP Gateway".
10. Enable OAuth 2.0 authentication in HTTP Gateway. See "Enabling OAuth 2.0 Authentication in HTTP Gateway" in *BRM Security Guide*.
11. (Optional) Enable SSL-based communications between the CHF and the NRF, PCF, and SMF. See "Securing Communication Between HTTP Gateway and the NRF, PCF, and SMF".

> **Note**
>
> Skip this step if you enabled SSL-based communication between the CHF and the NRF, PCF, and SMF during the ECE installation process. See "Installing an ECE Integrated System" in *ECE Installation Guide*.

12. (Optional) Configure your charging function (CHF) operations to route communication through a Services Communication Proxy (SCP). See "Configuring Communication Through SCP".
13. Start your HTTP Gateway. See "Starting the HTTP Gateway".

After the HTTP Gateway is set up, your 5G clients can start:

- Submitting ECE REST API requests to HTTP Gateway for online or offline charging by ECE. See "Using the ECE REST API".
- Sending unrated 5G usage events to HTTP Gateway so they can be converted into CDRs and sent to roaming partners, data warehousing system, and legacy billing systems. See "Generating CDRs for External Systems".

Configuring Registration Details for the HTTP Gateway Server

You can register:

- One or more HTTP Gateway server instances with an NRF registration server.
- One HTTP Gateway server instance with multiple NRF registration servers.
- One HTTP Gateway server instance with both primary and secondary NRF registration servers. In this case, when a primary NRF server goes down, the HTTP Gateway server registers with the secondary NRF server. See "Configuring Multiple Primary and Secondary NRF Registration Servers" for more information.

You do so by configuring the registration details in JConsole and then starting the HTTP Gateway instance.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To build an NRF registration request for an HTTP Gateway server instance:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.nfProfileConfigurations**.
4. Expand **Attributes**.
5. Specify the registration values for the fields in Table 22-1.

Table 22-1 Fields for Configuring NRF Registration

Attribute Name	Mandatory	Description
name	Yes	The name of the HTTP Gateway instance.
clusterName	Yes	The name of the cluster that the HTTP Gateway belongs to.
allowedNfDomains	No	The network function (NF) domains that are allowed to access the HTTP Gateway server. Enter a regular expression for the domains according to the ECMA-262 dialect [8]. If not provided, any NF domain is allowed to access the HTTP Gateway server.
allowedNfTypes	No	The type of NFs that are allowed to access the HTTP Gateway server, such as AMF or SMF. If not provided, any NF type is allowed to access the HTTP Gateway server.
allowedNssaisSd	No	The S-NSSAI Slice Differentiator (SD) ID of the network slices that are allowed to access the HTTP Gateway server. If not provided, any slice is allowed to access the HTTP Gateway server.
allowedNssaisSst	No	The S-NSSAI Slice/Service Type (SST) ID of the network slices that are allowed to access the HTTP Gateway server. If not provided, any slice is allowed to access the HTTP Gateway server.
allowedPlmnsMcc	No	The Mobile Country Code (MCC) of the PLMNs that are allowed to access the HTTP Gateway server. If not provided, any PLMN is allowed to access the HTTP Gateway server.
allowedPlmnsMnc	No	The Mobile Network Code (MNC) of the PLMNs that are allowed to access the HTTP Gateway server. If not provided, any PLMN is allowed to access the HTTP Gateway server.
capacity	No	The static capacity information in the range of 0-65535, expressed as a weight relative to other NF instances of the same type. The default is **0**. If the capacity is also present in the **nfServiceList** parameters, those will have precedence over this value.
customInfo	No	The specific data for custom NFs.
defaultNotificationSubscriptions CallbackUri	No	The callback URI for the default notification type.
defaultNotificationSubscriptions N1MessageClass	No	The information element (IE) that is used to identify the class of the N1 message type.
defaultNotificationSubscriptions N2InformationClass	No	The information element (IE) that is used to identify the class of the N2 message type.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-1 (Cont.) Fields for Configuring NRF Registration

Attribute Name	Mandatory	Description
defaultNotificationSubscriptions NotificationType	No	The type of notification for the corresponding callback URI.
fqdn	Yes	The FQDN of the HTTP Gateway server. For AMF, the FQDN registered with the NRF is the AMF name. For example: **chf-demo.novalocal**.
gpsiRangeListEnd	No	The ending range for the list of GPSIs that can be served by the CHF instance. The default is **100008**. If not provided, the CHF can serve any GPSI.
gpsiRangeListPattern	No	The pattern for the list of GPSI ranges that can be served by the CHF instance. The default is **^extid-.+@oracle1.com$**. If not provided, the CHF can serve any GPSI.
gpsiRangeListStart	No	The starting range for the list of GPSIs that can be served by the CHF instance. The default is **10000**. If not provided, the CHF can serve any GPSI.
heartBeatTimer	No	The time, in seconds, between two consecutive heartbeat messages from an NF Instance to the NRF.
httpGatewayName	Yes	The name of the HTTP Gateway that this property configuration belongs to.
interPlmnFqdn	No	The FQDN that is used for inter-PLMN routing as specified in 3GPP 23.003 [12]. This is required if the HTTP Gateway needs to be discoverable by other NFs in a different PLMN.
ipv4Addresses	No	The IPv4 addresses of the HTTP Gateway.
ipv6Addresses	No	The IPv6 addresses of the HTTP Gateway.
load	No	The current load percentage of the HTTP Gateway, ranging from 0 to 100.
locality	No	Operator-defined information about the location of the HTTP Gateway instance, such as the geographic location.
nfProfileChangesInd	No	Whether the IE is absent in the request to the NRF and may be included by the NRF in the NFRegister or NFUpdate response: • **true**: The NF Profile contains changes. • **false**: The complete NF Profile. This is the default.
nfProfileChangesSupportInd	No	Whether the IE may be present in the NFRegister or NFUpdate request and will be absent in the response: • **true**: The NF Service Consumer supports receiving NF Profile Changes in the response. • **false**: The NF Service Consumer does not support receiving NF Profile Changes in the response. This is the default.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-1 (Cont.) Fields for Configuring NRF Registration

Attribute Name	Mandatory	Description
nfProfileJsonRequestPayload	No	The complete NRF NF profile JSON used for NF registration. This field accepts either inline JSON or an absolute path to a JSON file accessible to the HTTP Gateway. When a file path is used, the JSON file must contain a top-level object with an entry set to the HTTP Gateway name, such as **httpgateway1**. The gateway reads only the entry matching its own name. For an example JSON payload file, see "Example NRF JSON Payload File". **Note:** When this field is set, HTTP Gateway uses the NRF NF Profile JSON for registration and ignores the settings under **charging.nfServiceConfigurations**. It also ignores all settings under **charging.nfProfileConfig**, except for the following attributes: • **nfProfileJsonRequestPayload** • **nrfApiVersionInUri** • **nrfHttp2Enable** • **nrfRestEndPointUrl**
nfServicePersistence	No	If present, and set to **true**, it indicates that the different instances of an NF Service in this NF instance, supporting the same API version, can persist their resource state in shared storage. Thus, these resources are available after a new NF service instance supporting the same API version is selected by a NF Service Consumer (see 3GPP 23.527 [27]). Otherwise, it indicates that the NF Service Instances of the same NF Service cannot share a resource state inside the NF Instance. Possible values are **true** or **false**.
nfSetId	No	A globally unique identifier of a set of equivalent and interchangeable NFs from an SNPN or PLMN network that provide distribution, redundancy, and scalability. This attribute is required if your system uses an SCP. See "Configuring Communication Through SCP". The default is empty. **Note:** You can include multiple instances of CHF under one NFSet ID. If one of the CHF instances is not available, SCP sends the request to another instance belonging to the same NFSet ID.
nfStatus	Yes	The status of the HTTP Gateway server: • **REGISTERED** • **SUSPENDED** • **UNDISCOVERABLE** The default is **REGISTERED**.
nfType	Yes	The type of NF. Set this to **CHF**. **Note:** ECE supports only **CHF**.
nrfApiVersionInUri	No	The API version segment, such as v1 or v2, used in nnrf-nfm endpoint URIs. If you use a JSON file to provide the complete NRF NF profile JSON used for NF registration, ensure this value matches the one for **apiVersionInUri** in the NRF JSON payload.
nrfHttp2Enable	No	Whether the endpoint URL of the NRF registration server (**nrfRestEndPointUrl**) uses the HTTP/2 protocol: • **true**: The HTTP/2 protocol is used. This is the default. • **false**: The HTTP/1 protocol is used.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-1 (Cont.) Fields for Configuring NRF Registration

Attribute Name	Mandatory	Description
nrfRestEndPointUrl	Yes	The endpoint URL of the NRF registration server. To register this HTTP Gateway instance with multiple NRF registration servers, list the endpoint URLs separated by a comma. For example: **http://localhost:8080,http://localhost:8081**. If this field is missing, NRF URL registration will not occur.
nrfSecondarySiteRestEndPointUrls	No	The endpoint URL of one or more secondary NRF servers. The list can include NRF servers on the same site and on different sites. See "Configuring Multiple Primary and Secondary NRF Registration Servers" for more information.
nsiList	No	The list of Network Slice Instances (NSIs) of the HTTP Gateway. If not provided, the HTTP Gateway can serve any NSI.
perPlmnSnssaiListPlmnIdMcc	No	The Mobile Country Code (MCC) value for **plmnSnssai**.
perPlmnSnssaiListPlmnIdMnc	No	The Mobile Network Code (MNC) value for **plmnSnssai**.
perPlmnSnssaiListSd	No	The Slice Differentiator (SD) value for **plmnSnssai**.
perPlmnSnssaiListSst	No	The Slice/Service Type (SST) value for **plmnSnssai**.
plmnListMcc	No	The MCC value for **plmnList**.
plmnListMnc	No	The MNC value for **plmnList**.
plmnRangeListEnd	No	The ending range for the list of PLMNs (including the PLMN IDs of the CHF instance) that can be served by the CHF instance. The default is **333333**. If not provided, the CHF can serve any PLMN.
plmnRangeListPattern	No	The pattern for the list of PLMNs (including the PLMN IDs of the CHF instance) that can be served by the CHF instance. If not provided, the CHF can serve any PLMN.
plmnRangeListStart	No	The starting range for the list of PLMNs that can be served by the CHF instance. The default is **100000**.
priority	No	The priority of this NF service relative to other services of the same type, where lower values indicate higher priority. The value must be in the range of 0-65535. The default is **1**. **Note**: The NRF may overwrite the received priority value when exposing an NFProfile.
recoveryTime	No	The timestamp for when the NF was started or restarted, in DateTime format.
snssaisSdl	No	The S-NSSAIs of the NF. When present, this IE represents the list of S-NSSAIs supported in all the PLMNs listed in the plmnList IE. If not provided, the NF can serve any S-NSSAI.
snssaisSst	No	The S-NSSAIs of the NF. When present, this IE represents the list of S-NSSAIs supported in all the PLMNs listed in the plmnList IE. If not provided, the CHF can serve any S-NSSAI.
supiRangeListEnd	No	The end of a list of GPSI ranges that can be served by the CHF instance, such as 1009. The default is **10008**. If not provided, the CHF can serve any GPSI.
supiRangeListPattern	No	The pattern for a list of GPSI ranges that can be served by the CHF instance. The default is **^nai-450081.+@.+org$**. If not provided, the CHF can serve any GPSI.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-1 (Cont.) Fields for Configuring NRF Registration

Attribute Name	Mandatory	Description
supiRangeListStart	No	The start of a list of GPSI ranges that can be served by the CHF instance. The default is **10000**. If not provided, the CHF can serve any GPSI.

Configuring Multiple Primary and Secondary NRF Registration Servers

You can configure an HTTP Gateway server instance to register with multiple primary and secondary NRF registration servers. When the heartbeat to a primary NRF server fails, HTTP Gateway performs the following actions:

1. HTTP Gateway retries the heartbeat up to the number of attempts specified by **nrfHeartBeatRetryCount**.
2. If the heartbeat retry attempts are exhausted, HTTP Gateway initiates registration with the secondary NRF server configured for that primary NRF server.
3. If registration with a secondary NRF server fails, HTTP Gateway retries registration with that NRF server up to the number of attempts specified by **nrfRetryCount** before attempting to register with a new NRF site. The **nrfRetryIntervalInSecond** attribute specifies the time to wait before retrying a connection to an NRF instance.
4. If the retry attempts are exhausted for a secondary NRF server, HTTP Gateway initiates registration with the next secondary NRF server in the configured order.

For descriptions and default values of these attributes, see Table 22-4 fields for an HTTP Gateway Instance.

When the primary NRF server becomes available again, HTTP Gateway re-registers with the primary NRF server. It then starts the heartbeat to the primary NRF server and deregisters from the secondary NRF server.

To configure an HTTP Gateway instance to register with both primary and secondary NRF registration servers:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.nfProfileConfigurations**.
4. Expand **Attributes**.
5. Specify the value for the following attributes:
   - **nrfRestEndPointUrl**: Set this to the endpoint URL of the primary NRF server.

     To configure multiple primary NRF servers, list the endpoint URLs separated by a comma (**,**).
   - **nrfSecondarySiteRestEndPointUrls**: Set this to the endpoint URL of the secondary NRF server.
     - To configure multiple secondary NRF servers for one primary NRF server, list the endpoint URLs, in order, separated by a semicolon (**;**). HTTP Gateway uses the secondary NRF servers in the order listed.
     - If you have multiple primary NRF servers and want to map each to a different secondary NRF server, separate the secondary endpoint URLs by a comma (**,**).

Copyright © 2019, 2026, Oracle and/or its affiliates.

  - If a secondary NRF server is not available for one of the primary NRF servers, enter **NA** or **None**.

Example: Configure One Primary with No Secondaries

This example specifies to configure a primary NRF server (**primaryNRF_1**) with no secondary NRF servers.

```
nrfRestEndPointUrl="www.primaryNRF_1.com"
nrfSecondarySiteRestEndPointUrls=""
```

Example: Configure One Primary with Two Secondaries

This example specifies to configure a primary NRF server (**primaryNRF_1**) with two secondary NRF servers (**secondaryNRF_A** and **secondaryNRF_B**).

```
nrfRestEndPointUrl="www.primaryNRF_1.com"
nrfSecondarySiteRestEndPointUrls="www.secondaryNRF_A.com;www.secondaryNRF_B.com"
```

Example: Configure One Primary with Two Secondaries, and Another Primary with One Secondary

This example specifies to configure:

- A primary NRF server (**primaryNRF_1**) with two secondary NRF servers (**secondaryNRF_A** and **secondaryNRF_B**)
- A primary NRF Server (**primaryNRF_2**) with one secondary NRF server (**secondaryNRF_C**)

```
nrfRestEndPointUrl="www.primaryNRF_1.com,www.primaryNRF_2.com"
nrfSecondarySiteRestEndPointUrls="www.secondaryNRF_A.com;www.secondaryNRF_B.com,www.secondaryNRF_C.com"
```

Example: Configure Three Primaries, Two with a Secondary and One with No Secondary

This example specifies to configure:

- A primary NRF server (**primaryNRF_1**) with one secondary NRF server (**secondaryNRF_A**)
- A primary NRF server (**primaryNRF_2**) with no secondary NRF server (**None** or **NA**)
- A primary NRF Server (**primaryNRF_3**) with one secondary NRF server (**secondaryNRF_B**)

```
nrfRestEndPointUrl="www.primaryNRF_1.com,www.primaryNRF_2.com,www.primaryNRF_3.com"
nrfSecondarySiteRestEndPointUrls="www.secondaryNRF_A.com,None,www.secondaryNRF_B.com"
```

Example NRF JSON Payload File

If you use a JSON file to provide the complete NRF NF profile JSON used for NF registration, ensure the contents of the file adhere to the following:

- The top-level JSON object includes an entry keyed by the HTTP Gateway name, such as **httpgateway1**. The gateway reads only the entry matching its own name.

  The HTTP Gateway name must match the **httpGatewayName** in your **charging-settings.xml** file.
- **nfServiceList** must be a JSON object keyed by the service instance ID, such as CLCHF1, not an array.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- The value of the **apiVersionInUri** JSON field matches that of the **nrfApiVersionInUri** entry in your **charging-settings.xml** file.

The following shows sample contents for an NRF JSON payload file:

```
{
   "httpgateway1": {
      "nfInstanceId": "cb350178-afab-45e6-b7f3-745532b2b04c",
      "nfType": "CHF",
      "nfStatus": "REGISTERED",
      "heartBeatTimer": 120,
      "plmnList": [
         {
            "mcc": "440",
            "mnc": "54"
         }
      ],
      "sNssais": [
         {
            "wildcardSd": false
         }
      ],
      "ipv4Addresses": [
         "10.5.192.9"
      ],
      "priority": 100,
      "capacity": 65535,
      "load": 0,
      "chfInfo": {
         "supiRangeList": [
            {
               "start": "440542150034380",
               "end": "440547000000899"
            }
         ]
      },
      "nfServicePersistence": false,
      "nfServiceList": {
         "CLCHF1": {
            "serviceInstanceId": "CLCHF1",
            "serviceName": "nchf-spendinglimitcontrol",
            "versions": [
               {
                  "apiVersionInUri": "v1",
                  "apiFullVersion": "1.R15.1.0",
                  "expiry": "2034-12-31T23:59:59.871Z"
               }
            ],
            "scheme": "http",
            "nfServiceStatus": "REGISTERED",
            "ipEndPoints": [
               {
                  "ipv4Address": "10.5.192.9",
                  "transport": "TCP",
                  "port": 8080
               }
            ],
            "priority": 100,
            "capacity": 100,
            "load": 0
         }
      },
      "lcHSupportInd": false,
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
        "olcHSupportInd": false
      }
    }
```

Configuring NF Services

> **Note**
>
> If the nfProfileJsonRequestPayload field is set, HTTP Gateway ignores the configuration in this section and instead uses the NRF NF Profile JSON file for registration.

You must configure at least one NF service for HTTP Gateway. By default, HTTP Gateway includes one, but you can add more.

To add or remove an NF service configuration, do this:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole.
2. Expand the **ECE Configuration** node.
3. Expand **charging.nfServiceConfigurations**.
4. Expand **Operations**.
5. Do one of the following:
   - To add an NF service configuration, enter its name and then click **addNfServiceConfiguration**.
   - To remove an NF service configuration, enter its name and then click **removeNfServiceConfiguration**.

To configure an NF service, do this:

1. Access the ECE configuration MBeans.
2. Expand the **ECE Configuration** node.
3. Expand **charging.nfServiceConfigurations.***Instance_Name*.
4. Expand **Attributes**.
5. Specify the NF service configuration values for the attributes in Table 22-2.

Table 22-2 NF Service Configuration Attributes

Attribute Name	Mandatory	Description
allowedNfDomains	No	The network function (NF) domains that are allowed to access the service instance. Enter a regular expression for the domains according to the ECMA-262 dialect. If not provided, any NF domain is allowed to access the service instance.
allowedNfTypes	No	The type of NFs that are allowed to access the service instance. If not provided, any NF type can access the service instance.
allowedNssaisSd	No	The S-NSSAI Slice Differentiator (SD) ID of the network slices that are allowed to access the service instance. If not provided, any slice can access the service instance.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-2 (Cont.) NF Service Configuration Attributes

Attribute Name	Mandatory	Description
allowedNssaisSst	No	The S-NSSAI Slice/Service Type (SST) ID of the network slices that are allowed to access the service instance. If not provided, any slice can access the service instance.
allowedPlmnsMcc	No	The Mobile Country Code (MCC) of the PLMNs that are allowed to access the service instance. If not provided, any PLMN can access the service instance. When included, the **allowedPlmns** attribute does not need to include the PLMN IDs registered in the **plmnList** attribute of the NF Profile.
allowedPlmnsMnc	No	The Mobile Network Code (MNC) of the PLMNs that are allowed to access the service instance.
apiFullVersion	Yes	The full version number of the API as specified in 3GPP 29.501.
apiPrefix	No	The optional path segments used to construct the {apiRoot} variable of the different API URIs.
apiVersionInUri	Yes	The version of the service instance to be used in the URI for accessing the API.
capacity	No	The static capacity information in the range of 0-65535, expressed as a weight relative to other services of the same type. The default is **50**. The capacity and priority parameters, if present, are used for NF selection and load balancing.
defaultNotificationSubscriptions CallbackUri	No	The callback URI for the default notification type.
defaultNotificationSubscriptions N1MessageClass	No	The information element (IE) that is used to identify the class of the N1 message type.
defaultNotificationSubscriptions N2InformationClass	No	The information element (IE) that is used to identify the class of the N2 message type.
defaultNotificationSubscriptions NotificationType	No	The type of notification for the corresponding callback URI.
expiry	No	The expiration date and time of the NF service. The default is **2020-12-01T18:55:08.871Z**.
fqdn	No	The FQDN of the NF service instance.
interPlmnFqdn	No	The FQDN that is used for inter-PLMN routing as specified in 3GPP 23.003. This is required if the service instance needs to be discoverable by other NFs in a different PLMN.
ipv4Address	No	The IPv4 address.
ipv6Address	No	The IPv6 address.
load	No	The current load percentage of the NF service, ranging from 1 through 100. The default is **5**.
name	Yes	The name of the NF service.
nfServiceStatus	Yes	The status of the NF service instance: • **REGISTERED** • **SUSPENDED** • **UNDISCOVERABLE** The default is **REGISTERED**.
port	No	The port number. The default is **0**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-2 (Cont.) NF Service Configuration Attributes

Attribute Name	Mandatory	Description
primaryChfServiceInstance	No	The specific data for a CHF service instance. Include this IE if the CHF service instance serves as a secondary CHF instance of another primary CHF. When present, set it to the **serviceInstanceId** of the primary CHF service instance.
priority	No	The priority used for service selection (relative to other services of the same type), ranging from 0 through 65535. Lower values indicate a higher priority. The default is **1**. The NRF may overwrite the received priority value when exposing an NFProfile with the Nnrf_NFDiscovery service.
recoveryTime	No	The timestamp when the NF service was started or restarted. For example, 2019-08-03T18:55:08.871Z. The format should be of type DateTime.
scheme	Yes	The URI scheme, such as **http** or **https**. The default is **http**.
secondaryChfServiceInstance	No	Include this IE if the CHF service instance serves as a primary CHF instance of another secondary CHF. When present, set it to the **serviceInstanceId** of the secondary CHF service instance. Do not set this IE when **primaryChfServiceInstance** is present.
serviceInstanceId	Yes	The unique ID of the service instance within a given NF instance. The default is **chf1**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-2 (Cont.) NF Service Configuration Attributes

Attribute Name	Mandatory	Description
serviceName	Yes	The name of the service instance: • **nnrf-nfm** • **nnrf-disc** • **nudm-sdm** • **nudm-uecm** • **nudm-ueau** • **nudm-ee** • **nudm-pp** • **namf-comm** • **namf-evts** • **namf-mt** • **namf-loc** • **nsmf-pdusession** • **nsmf-event-exposure** • **nausf-auth** • **nausf-sorprotection** • **nausf-upuprotection** • **nnef-pfdmanagement** • **npcf-am-policy-control** • **npcf-smpolicycontrol** • **npcf-policyauthorization** • **npcf-bdtpolicycontrol** • **npcf-eventexposure** • **npcf-ue-policy-control** • **nsmsf-sms** • **nnssf-nsselection** • **nnssf-nssaiavailability** • **nudr-dr** • **nlmf-loc** • **n5g-eir-eic** • **nbsf-management** • **nchf-spendinglimitcontrol** • **nchf-convergedcharging** • **nnwdaf-eventssubscription** • **nnwdaf-analyticsinfo** The default is **nchf-convergedcharging**.
supportedFeatures	No	The supported features of the NF Service instance.
transport	No	The transport protocol. The default is **TCP**.
httpGatewayName	Yes	The name of the HTTP Gateway that this property configuration belongs to.
clusterName	Yes	The name of the cluster that the HTTP Gateway belongs to.

Configuring HTTP Gateway for Convergent Charging

Configure the HTTP Gateway to send usage requests to ECE Server for convergent charging, and to consume SNR and RAR notifications from the ECE notification topics.

To configure the HTTP Gateway for convergent charging:

Copyright © 2019, 2026, Oracle and/or its affiliates.

1. To allow secure communication between the CHF and the PCF, NRF, SMF, generate a self-signed SSL certificate for the CHF.
2. Access the ECE configuration MBeans in a JMX editor, such as JConsole.
3. Expand the **ECE Configuration** node.
4. Expand **charging.httpGatewayConfigurations**.
5. Expand **Attributes**.
6. Specify values for the fields in Table 22-3.

Table 22-3 Fields for HTTP Gateway

Name	Default	Description
serverSslKeyStore	"httpGatewayServer.jks"	The name of the KeyStore file for ECE server.
serverSslKeyStoreType	"@ECE_HTTPGATEWAY_KEYSTORE_TYPE@"	The SSL KeyStore type for ECE server, such as **JKS** or **PKCS12**.
serverSslKeyStoreAlias	"@ECE_HTTPGATEWAY_KEYSTORE_ALIAS@"	The alias name for the ECE server SSL KeyStore.
walletLocation	"@ECE_WALLET_LOCATION@"	The path to the ECE wallet.
snrHttp2Enable	"true"	Whether HTTP Gateway uses the HTTP/2 protocol: • **true**: The HTTP/2 protocol is used. • **false**: The HTTP/1 protocol is used.
retryIntervalInMillis	"5000"	The amount of time, in milliseconds, between reconnection attempts to ECE server.
httpIdentityKeystore	"@HTTP_IDENTITY_KEYSTORE@"	The path to the Identity KeyStore certificate file. The default is **httpIdentityKeystore.p12**.
httpIdentityKeystoreType	"@HTTP_IDENTITY_KEYSTORE_TYPE@"	The type of Identity KeyStore: **SSO** or **PKCS12**. The default is **PKCS12**.
httpTruststore	"@HTTP_TRUSTSTORE@"	The path to the TrustStore file. The default is **httpTruststore.p12**.
httpTruststoreType	"@HTTP_TRUSTSTORE_TYPE@"	The type of TrustStore: **SSO** or **PKCS12**. The default is **PKCS12**.
includeTrailingForwardSlash	"true"	Whether HTTP Gateway adds a trailing forward slash (*/*) to the RAR notification URL (**true**) or not (**false**). The default is **true**.
ecsClusterHealthStatus	""	Specifies the ECS cluster health status value detection. This is a read-only value that the HTTP Gateway updates automatically.

7. Expand **charging.HttpGatewayConfigurations.***name*, where *name* is the name of the HTTP Gateway instance.
8. Expand **Attributes**.
9. Specify values for the fields in Table 22-4.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-4 Fields for an HTTP Gateway Instance

Name	Default	Description
name	"@HTTPGATEWAY_NAME@"	The name of the HTTP Gateway instance.
clusterName	"@CLUSTER_NAME@"	The name of the cluster that the HTTP Gateway belongs to.
serverHttp2Enabled	"@HTTPGATEWAY_HTTP2_ENABLED@"	Whether ECE Server uses the HTTP/2 protocol: • **true**: The HTTP/2 protocol is used. • **false**: The HTTP/1 protocol is used.
serverPort	"@HTTPGATEWAY_SERVER_PORT@"	The HTTPS port number of the server on which HTTP Gateway resides.
serverHttpPort	"@HTTPGATEWAY_SERVER_HTTP_PORT@"	The HTTP port number of the server on which HTTP Gateway resides.
nrfHeartBeatRetryCount	"4"	The number of times the heartbeat retries if it fails during registration.
serverSslEnabled	"@HTTPGATEWAY_SERVER_SSL_ENABLED@"	Whether SSL communication is enabled between your HTTP client and the HTTP Server (**true**) or not (**false**).
processingThreadPoolSize	"200"	The number of threads used by the HTTP Gateway instance to process a set of incoming usage requests.
processingQueueSize	"32768"	The number of incoming usage requests that can be processed simultaneously by the HTTP Gateway.
kafkaBatchSize	"10"	The size of the Kafka batch.
externalTrafficInfo	""	The location of the header URL information.
nrfRetryIntervalInSecond	"60"	The wait interval between NRF retry attempts.
nrfRetryCount	"3"	The number of registration attempts for each NRF instance before attempting to register on a new NRF site.
kafkaPartition	“5,6,7,8,9”	A comma-separated list of dedicated Kafka partition IDs consumed by HTTP Gateway replicas for Kafka partition-based notification routing. For the required number of partition IDs, replica-to-partition assignment order, and uniqueness requirements, see "Configuring Kafka Partition Assignments for Gateways." To calculate the total number of partitions required in the ECE notification topic, see "Determining the Number of Partitions Required" and "Creating Kafka Topics for ECE Integration." **Note:** For Kafka topic-based routing, this attribute is ignored.
pcfSslEnabled	"@PCF_SSL_ENABLED@"	Whether SSL communication is enabled between the CHF and the PCF (**true**) or not (**false**). The default is **false**.
nrfSslEnabled	"@NRF_SSL_ENABLED@"	Whether SSL communication is enabled between the CHF and the NRF (**true**) or not (**false**). The default is **false**.
smfSslEnabled	"@SMF_SSL_ENABLED@"	Whether SSL communication is enabled between the CHF and the SMF (**true**) or not (**false**). The default is **false**.
httpSSLType	"@HTTP_SSL_TYPE@"	The type of SSL connection between the CHF and PCF, NRF, or SMF: • **oneway**: The client must authenticate the server's identity. This is the default. • **twoway**: The client and server must authenticate each others identity.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-4 (Cont.) Fields for an HTTP Gateway Instance

Name	Default	Description
oauth2Enabled	"@OAUTH2_ENABLED@"	Whether incoming ECE API requests must have the access token attached to the header (**true**) or not (**false**). For information about setting up OAuth 2.0, see "ECE REST API Security" in *BRM Security Guide*.
ingressRequestTimeoutMs	"3000"	Specifies how long the 5G client waits for a response, in milliseconds, before timing out. After this amount of time, the HTTP Gateway terminates the request. The default is **3000**. **Note**: Set this to a slightly lower value than the client's timeout to account for the time spent traversing the network.
enableNetworkExpiration	"true"	Whether to calculate the ingress request timeout using the time stamp contained in the payload (**true**) or using the time the message arrived (**false**).
chfNetworkId	""	The CHF Network ID for the HTTP Gateway. **Note:** This is used by Kafka topic-based routing when creating topic names.
ccnNotificationAgeLimitInSec	"300"	The time limit, in seconds, after which any Converged Charging Notify (CCN) notification consumed by the HTTP Gateway is considered expired and delivery attempts for the N40 notification to the N40 client are halted.
slcnNotificationAgeLimitInSec	"300"	The time limit, in seconds, after which any Spending Limit Control Notify (SLCN) notification consumed by HTTP Gateway is considered expired and delivery attempts for the N28 notification to the N28 client are halted.

Editing the HTTP Gateway Mediation Specification File

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

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

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

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

- ProductType (service)
- EventType
- Version

To edit the mediation specification:

1. Create a mediation specification file or edit an existing one.

Copyright © 2019, 2026, Oracle and/or its affiliates.

A sample mediation specification file is available at *ECE_home*/**sample_data/config_data/specifications/ece_end2end/http_mediation.spec**.

It is recommended to create only one mediation specification file. 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:
   - **Service-Identifier**: The Service-Identifier is a placeholder.
   - **Rating-Group**: The Rating-Group value sent in the ECE REST API request.
   - **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 this information for each event definition object defined in the mediation table.

   For each received request, HTTP Gateway correlates the Rating-Group, and Event-Timestamp 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 HTTP Gateway to recognize a newly deployed event definition object.

   For example, to have requests processed according to a new specification on December 16, 2020, you would enter:

   ```
   |   ValidFrom
    | "2020-12-16T12:01:01"
   ```

   You can also specify a time zone. For example,

   ```
   |   ValidFrom
    | "2020-12-16T12:01:01 PST"
   ```

   If a time zone is not sent, the **ValidFrom** field is set to UTC.
4. Save the **http_mediation.spec** file in the directory where you save your configuration data.
5. Verify that the directory specified in the *ECE_home*/**config/management/migration-configuration.xml** file is the directory where you save your configuration data.
6. Run the **configLoader** utility:

   ```
   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, HTTP Gateway re-creates its in-memory usage-request builder map and begins using the mapping definitions (to send requests that adhere to the specifications) provided that the **validFrom** date is reached.
7. Restart the HTTP Gateway.
8. Load the mediation specification file into the ECE server by using the **configLoader** utility.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Mapping nCHF Request Fields to ECE Payload Attributes

HTTP Gateway uses request specification XML files to map fields from incoming nCHF charging requests to ECE payload attributes. The payload attributes are used during charging, rating, extension processing, and call detail record (CDR) generation.

> **Note**
>
> A request specification differs from the HTTP mediation specification. The HTTP mediation specification determines the service, event, and request builder used to process an incoming request. The request specification defines how request fields are mapped to payload attributes for the selected event.

To configure request field mappings, create or modify a request specification file and load it into ECE:

1. Create a request specification file or open the request specification file used by your deployment.

   ECE provides sample request specification files in the *ECE_home***/sample_data/config_data/specifications/ece_http** directory. The sample file names end with **_Specification.xml**.
2. Update the request specification to add or modify the required request field mappings and payload attributes for the applicable event.
3. Save the specification file in the directory specified by the **configObjectsDataDirectory** parameter in the *ECE_home***/config/management/migration-configuration.xml** file.
4. Run the **configLoader** utility to load the updated request specification into ECE:

   ```
   start configLoader
   ```

   For more information, see "configLoader" in *BRM System Administrator's Guide*.

Retaining IPv4 and IPv6 PDP Address Information in the ECE Payload

HTTP Gateway supports mapping IPv4-only, IPv6-only, and dual-stack Protocol Data Unit (PDU) address information from incoming nCHF charging requests into ECE payload attributes. This enables ECE to retain IPv4 and IPv6 PDU address information independently in the GPRS_INFO payload for downstream processing, such as custom extension processing and CDR generation.

To retain both IPv4 and IPv6 address information, update the DATA_USAGE event-attribute specification used by your deployment. ECE provides a sample *ECE_home***/sample_data/config_data/specifications/ece_http/Data_Usage_Specification.xml** file that you can use as a guideline. In a production deployment, update the appropriate ***_Specification.xml** file used for the DATA_USAGE event in the directory configured by the **configObjectsDataDirectory** parameter in the *ECE_home***/config/management/migration-configuration.xml** file.

Table 22-5 shows the supported pduAddress request fields and the corresponding ECE payload attributes.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 22-5 Supported pduAddress Request Fields and ECE Payload Attributes

nCHF Request Field	ECE Payload Attribute	Type
pduIPv4Address	PDP_ADDRESS (legacy mapping)	String
pduIPv4Address	PDP_ADDRESS_IPV4	String
pduIPv6AddresswithPrefix	PDP_ADDRESS_IPV6	String
pduAddressprefixlength	PDP_ADDRESS_PREFIX_LENGTH	Integer
iPv4dynamicAddressFlag	PDP_ADDRESS_IPV4_DYNAMIC_FLAG	Integer
iPv6dynamicPrefixFlag	PDP_ADDRESS_IPV6_DYNAMIC_FLAG	Integer

> **Note**
>
> For **iPv4dynamicAddressFlag** and **iPv6dynamicPrefixFlag**, HTTP Gateway converts Boolean values to integers: **true** becomes **1**, and **false** becomes **0**.

To modify the DATA_USAGE event-attribute specification:

1. Open the request specification file that defines the DATA_USAGE event-attribute specification used by your deployment.
2. Update the DATA_USAGE event-attribute specification to add or modify the required payload attributes and request field mappings:

```
<complexAttributeItem>
    <name>GPRS_INFO</name>
    <description>GPRS_INFO</description>
    <type>STRUCT</type>
    <simpleAttributeItem>
        <name>PDP_ADDRESS</name>
        <description>PDP_ADDRESS</description>
        <type>STRING</type>
        <networkAttributeItem></networkAttributeItem>

<networkAttributeItem5g>pDUSessionChargingInformation.pduSessionInformation
.pduAddress.pduIPv4Address</networkAttributeItem5g>
    </simpleAttributeItem>
    <simpleAttributeItem>
        <name>PDP_ADDRESS_IPV4</name>
        <description>PDP_ADDRESS_IPV4</description>
        <type>STRING</type>
        <networkAttributeItem></networkAttributeItem>
        <networkAttributeItem5g>
pDUSessionChargingInformation.pduSessionInformation.pduAddress.pduIPv4Addre
ss</networkAttributeItem5g>
            </simpleAttributeItem></complexAttributeItem>
```

3. Save the specification file in the directory configured by **configObjectsDataDirectory**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

4. Run the **configLoader** utility to load the updated request specification into ECE:

   ```
   start configLoader
   ```

   For more information, see "configLoader" in *BRM System Administrator's Guide*.

5. Verify that the updated request specification is loaded successfully before processing nCHF charging requests.

The following example shows a part of an incoming nCHF charging request with a dual-stack pduAddress object:

```
"pduAddress": {
    "pduIPv4Address": "198.51.100.254",
    "pduIPv6AddresswithPrefix": "2001:0db8:0000:0000:0000:8a2e:0370:7334.",
    "pduAddressprefixlength": 64,
    "iPv4dynamicAddressFlag": true,
    "iPv6dynamicPrefixFlag": true
}
```

After processing the request, HTTP Gateway populates the GPRS_INFO payload with the corresponding attribute values.

```
Block "GPRS_INFO" {
    String  "PDP_ADDRESS"                    192.168.10.2
    String  "PDP_ADDRESS_IPV4"               192.168.10.2
    Integer "PDP_ADDRESS_IPV4_DYNAMIC_FLAG"  1
    String  "PDP_ADDRESS_IPV6"               2001:db8:85a3::8a2e:370:7334/64
    Integer "PDP_ADDRESS_IPV6_DYNAMIC_FLAG"  1
    Integer "PDP_ADDRESS_PREFIX_LENGTH"      64
}
```

> **Note**
>
> When pduIPv4Address is present in the incoming nCHF request, ECE continues to populate the legacy PDP_ADDRESS payload attribute from the IPv4 address to maintain compatibility with existing customizations. Use PDP_ADDRESS_IPV4 and PDP_ADDRESS_IPV6 when customizations need to distinguish between IPv4 and IPv6 addresses.

Checking for Client-Side Timeouts in HTTP Gateway

5G clients can contain their own request timeouts that activate when ECE takes too long to respond. When this occurs, the 5G client abandons the request, but ECE may continue to process the request and send a response, which the client discards. This extra processing is unnecessary and can be detrimental to ECE production systems approaching overload conditions. To improve performance, you can configure the HTTP Gateway to terminate requests that have timed out on the client side.

To configure the HTTP Gateway to check for client-side timeouts:

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

3. Expand **charging.HttpGatewayConfigurations.***name*, where name is the name of the HTTP Gateway instance.
4. Expand **Attributes**.
5. Specify timeout values for the HTTP Gateway instance:
   - **ingressRequestTimeoutMs**: Specifies how long the 5G client waits for a response, in milliseconds, before timing out. After this amount of time, the HTTP Gateway terminates the request. The default is **3000**.

     > **Note**
     >
     > Set this to a slightly lower value than the client's timeout to account for the time spent traversing the network.

   - **enableNetworkExpiration**: Specifies whether to calculate the ingress request timeout using the time stamp contained in the request payload (**true**) or the time the message arrived at the HTTP Gateway (**false**). The default is **true**.
6. Expand **charging.brsConfigurations.***name*, where *name* is the name of the BRS instance.
7. Expand **Attributes**.
8. Specify timeout values for the BRS instance:
   - **requestMaxRetry**: Specifies the maximum number of times the BRS retries sending the ingress request to ECS. Use a value between 1 and 5. The default is 3.
   - **requestRetryWaitTimeMs**: Specifies the amount of time, in milliseconds, between each retry attempt. The default is 5.

When a request is terminated, the HTTP Gateway logs an error and sets the **ResponseConsumerFailureCode** field to **Expired** in the response to the client.

Configuring ECE to Publish Notifications to HTTP Gateway

You can configure ECE to publish Nchf_SpendingLimitControl_Notify and Nchf_ConvergedCharging_Notify notifications to the ECE notification topics. The HTTP Gateway retrieves these notifications from the topics.

To configure ECE to publish notifications to HTTP Gateway:

1. Configure HTTP Gateway to consume notifications from Kafka.

   For Kafka partition-based routing, when **kafkaCustomPartitionAssignorEnabled** is set to **false**, configure the **kafkaPartition** attribute for each HTTP Gateway configuration. Specify a unique, comma-separated list containing at least one dedicated partition ID for each HTTP Gateway replica that might run. See "Configuring HTTP Gateway for Convergent Charging."

   For Kafka topic-based routing, when **kafkaCustomPartitionAssignorEnabled** is set to **true**, the **kafkaPartition** attribute is ignored.
2. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
3. Expand the **ECE Configuration** node.
4. Expand **charging.server**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. Expand **Attributes**.
6. Set **kafkaEnabledForNotifications** to **true**.
7. Set the **policyCounterNotificationAggregationEnabled** property to **true** to enable policy counter notification aggregation. When enabled, ECE aggregates the latest policy label during request processing and sends a single SNR for each applicable policy counter. By default, this property is set to **false**.
8. Expand **charging.httpGatewayConfigurations**.
9. Expand **Attributes**.
10. Set **notificationListenerConnectionPoolSize** to the number of threads that HTTP Gateway uses to consume notifications from the ECE notification topics and send notification requests to the configured notification URI. The default is **10**.
11. Expand **charging.notification**.
12. Expand **Attributes**.
13. Set **rarNotificationMode** and **spendingLimitNotificationMode** to one of the following values:
    - **NONE**: ECE does not send this type of notification.
    - **ASYNCHRONOUS**: ECE sends notifications asynchronously to the ECE notification topics.

      If configured, HTTP Gateway consumes notifications from the ECE notification topics and dispatches them through REST API calls.

Securing Communication Between HTTP Gateway and the NRF, PCF, and SMF

You can enable secure communication between the HTTP Gateway (CHF) and the NRF, PCF, and SMF during the ECE installation process. See "Installing an ECE Integrated System" in *ECE Installation Guide* for more information.

If you did not enable it during the ECE installation process, you can secure communication during a maintenance window.

To secure communication between HTTP Gateway and the NRF, PCF, and SMF:

1. Generate your SSL TrustStore and Identity KeyStore certificates for the NRF, PCF, and SMF.
2. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
3. Expand the **ECE Configuration** node.
4. Expand **charging.httpGatewayConfigurations**.
5. Expand **Attributes**.
6. Specify values for the following fields:
    - **httpIdentityKeystore**: Specify the path to the Identity KeyStore certificate files.
    - **httpIdentityKeystoreType**: Specify the type of SSL Identity KeyStore: **PKCS12** or **SSO**.
    - **httpTruststore**: Specify the path to the SSL TrustStore files.
    - **httpTruststoreType**: Specify the type of SSL TrustStore file: **PKCS12** or **SSO**.

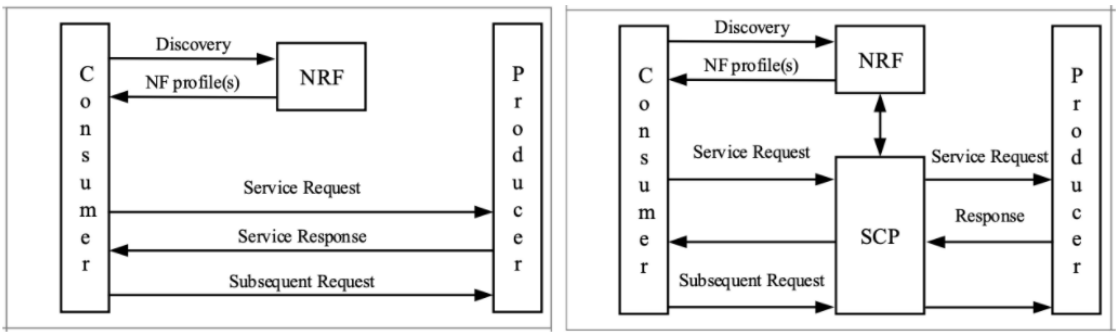
Copyright © 2019, 2026, Oracle and/or its affiliates.

7. Expand **charging.httpGatewayConfigurations.***name*, where *name* is the name of the HTTP Gateway instance.
8. Expand **Attributes**.
9. Specify values for the following fields:
   - **pcfSSLEnabled**: Set this to **true** to enable SSL-based communication between the CHF and PCF.
   - **nrfSSLEnabled**: Set this to **true** to enable SSL-based communication between the CHF and NRF.
   - **smfSSLEnabled**: Set this to **true** to enable SSL-based communication between the CHF and SMF.
   - **httpSSLType**: Specify the type of SSL communication: **oneway** or **twoway**.
   - **scpAuthorities**: Enter the URL of the primary and secondary SCP authority, delimited by commas. For example: scpAuthorities="scp1.example.com,scp2.example.com".
10. If your system routes communication between the CHF and other network functions through an Oracle Services Communications Proxy (SCP), also set this attribute:
    - **scpAuthorities**: Enter the URL of the primary and secondary SCP authority, delimited by commas. For example: scpAuthorities="scp1.example.com,scp2.example.com".

Configuring Communication Through SCP

HTTP Gateway supports the communication models shown in Figure 22-1 for the charging function (CHF) operations.

Figure 22-1 Supported Communication Models for CHF Operations



Configuring your system to route communication between the CHF and other NFs through an Oracle Services Communications Proxy (SCPs) involves the following high-level tasks:

1. Deploy two or more interchangeable CHF instances within the same PLMN or SNPN network and assign them the same NFSet ID.
2. Include the NFSet ID in the HTTP Gateway's registration request to the NRF registration server.
3. Provide the SCP KeyStore credentials to HTTP Gateway.
4. Supply the SCP connection details to each instance of the HTTP Gateway.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> NFSet IDs enable the SCP to reroute charging requests to another CHF instance within the same NFSet if one of the CHF instances becomes unavailable.

To configure HTTP Gateway to communicate through an SCP:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.nfProfileConfigurations**.
4. Expand **Attributes**.
5. Set **nfSetId** to a unique identifier for a set of equivalent and interchangeable NFs in a PLMN or SNPN network:
   - For PLMN networks, use this format:

     set*setID*`.chfset.5gc.mnc`*threeDigitMnc*`.mcc`*threeDigitMcc*

     For example: **set12.chfset.5gc.mnc007.mcc427**.
   - For SNPN networks, use this format:

     set*setID*`.chfset.5gc.nid`*threeDigitNid*`.mnc`*threeDigitMnc*`.mcc`*threeDigitMcc*

     For example: **set10.chfset.5gc.nid283.mnc012.mcc345**.

   where:
   - *setID* consists of alphabetic characters, digits (0-9), and, optionally, a hyphen (-). The characters are not case specific.
   - *threeDigitNid* is a three-digit code for a Network Interface Device (NID). The NID code is assigned by an SNPNs during deployment.
   - *threeDigitMnc* is a three-digit code for a Mobile Network Code (MNC). If there are only two significant digits, start the code with a zero.
   - *threeDigitMcc* is a three-digit code for a Mobile Country Code (MCC).
6. Expand the **ECE Configuration** node.
7. Expand **charging.httpGatewayConfigurations**.
8. Expand **Attributes**.
9. Configure the following attributes:
   - Select **httpIdentityKeystore** and enter the name and path to the SCP-related KeyStore certificate file.
   - Select **httpTruststore** and enter the name and path to the SCP SSL TrustStore file.
10. Expand the **ECE Configuration** node.
11. Expand **charging.HttpGatewayConfigurations.***name*, where *name* is the name of the HTTP Gateway instance.
12. Expand **Attributes**.
13. Configure the following attributes:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- Select **scpAuthorities** and enter the URL of the primary and secondary SCP authority, delimited by commas. For example: scpAuthorities="scp1.example.com,scp2.example.com".
- Select **nrfHeartBeatRetryCount** and enter the maximum number of tries for a heartbeat call. If all retries have been exhausted, HTTP Gateway starts registering with a new NRF. The default is **4**.
- Select **nrfRetryCount** and enter the number of retries for NRF registration. The default is **3**.

Starting the HTTP Gateway

When the HTTP Gateway starts, it automatically joins the Coherence cluster and gains access to ECE caches and invocation services that it uses to send requests to ECS. At startup, HTTP Gateway checks for notifications in the ECE notification topics.

To start the HTTP Gateway:

1. Ensure that the HTTP Gateway server and other components are started.
2. Start ECC:

   ```
   ./ecc
   ```

3. Start the HTTP Gateway:

   ```
   ecc:000> start httpGateway
   ```

Using the ECE REST API

After you set up the HTTP Gateway on your system, your 5G clients can submit requests to the ECE REST API. For more information about the ECE REST API operation types, see *REST API Reference for Elastic Charging Engine*.

You make calls to the ECE REST API through Swagger at this URL:

https://*hostname*:*serverhttpsPort*/openapi-ui/index.html

where *hostname* is the host name of the machine on which HTTP Gateway is running, and *serverhttpsPort* is the port number on the HTTP Gateway server.

If your system's communication model includes an SCP:

- You must include the **Authority** header in all HTTP requests to the CHF operations. Set the **Authority** header to the host name and port number of the SCP Authority Server. For example:

  ```
  Authority: example.com:1534
  ```

- CHF responses include a **3gpp-Sbi-Binding** header to indicate binding preferences to SCP and NF consumers. The header appears in the following format:

  ```
  3gpp-Sbi-Binding: bl=nf-set; nfset=NFSet_ID; recoverytime=timestamp
  ```


Copyright © 2019, 2026, Oracle and/or its affiliates.

23

Generating CDRs for External Systems

You can configure Oracle Communications Elastic Charging Engine (ECE) to generate call detail records (CDRs) for unrated 5G usage events. ECE does not use CDRs for convergent charging, so it does not generate them by default. You might want CDRs for roaming partners, data warehousing, and legacy billing systems.

Topics in this document:

- About Using the HTTP Gateway
- About Generating CDRs
- About Saving CDR Files to Disk
- About the CDR Generation Process
- Setting Up ECE to Generate CDRs
- About Trigger Types

> ⚠ **Caution**
>
> Generating CDRs for unrated 5G events requires a cloud native deployment of ECE and BRM components. The HTTP Gateway and CDR Gateway can be used only on an ECE cloud native system.

About Using the HTTP Gateway

By default, HTTP Gateway sends all 5G usage requests to ECE Server for online and offline charging.

You can configure HTTP Gateway to convert some of the usage requests into CDRs based on the charging type by enabling CDR generation. You can then send the CDRs to roaming partners, a data warehousing system, or legacy billing systems for rating.

When CDR generation is enabled, HTTP Gateway routes usage requests to one of the following, depending on your configurations and the charging type:

- ECE Server for charging. See "Using the ECE REST API".
- The CDR Gateway for generating CDRs. See "About Generating CDRs".

You specify where to route online charging requests and offline charging requests when you configure the HTTP Gateway. See "Configuring HTTP Gateway for CDR Generation".

About Generating CDRs

You can configure ECE to generate CDR files for unrated 5G usage events. You might want CDR files for roaming partners, data warehousing, and legacy billing systems. ECE can publish CDR files to your file system or a Kafka messaging service. In both cases, the files are

Copyright © 2019, 2026, Oracle and/or its affiliates.

in JSON format. For details about the CDR format, see "CHF-CDR Format" in *ECE 5G CHF Protocol Implementation Conformance Statement*.

When configuring ECE to generate CDRs, you can specify whether to:

- Generate CDRs for online charging requests, offline charging requests, or both.
- Generate individual CDRs or aggregate multiple CDRs together according to the trigger criteria you specify.
- Purge completed CDRs from the CDR store that are older than a specified number of days.
- Purge orphaned CDRs from the store that are older than a specified number of seconds.
- Close open CDR sessions when a terminate request does not arrive within a configurable amount of time.

If you set up your ECE system for disaster recovery, you can also specify to:

- Mark partially processed CDRs in the CDR Store as incomplete.
- Mark when a duplicate usage update occurs in a CDR.
- Support a custom value for why a CDR session was closed.

For more information, see "About CDR Generator in an Active-Active System" in *BRM System Administrator's Guide*.

About Saving CDR Files to Disk

If you configure CDR Gateway to save JSON-formatted CDR files to disk, it stores the files to the directory you specify using the following file naming format:

```
ClusterName_StartTimeStamp_EndTimeStamp_SequenceNumber.Extension
```

For example: BRM_1654514133000_1654514134000_1.out

You set the file name extension, the maximum number of CDRs that can be written to a CDR output file, and the directory in which to store CDR files when you configure the CDR Formatter Plug-In. See "Configuring the CDR Formatter Plug-in".

About the CDR Generation Process

ECE generates CDR files in the Charging Function component (CHF) using this process:

1. The HTTP Gateway sends a request to the CDR Gateway. The requests can be for online charging, offline charging, or both.
2. The CDR Gateway does the following:
   a. Generates individual CDR records for each request or aggregates multiple requests into a CDR record according to the trigger criteria you specify.
   b. Stores CDR records in the CDR database. You can use either Oracle NoSQL Database or Oracle Database.
3. The CDR Formatter does the following:
   a. Extracts CDR records from the database and passes them to the CDR Formatter plug-in module for processing. You can use the default plug-in included with ECE or create a custom plug-in.

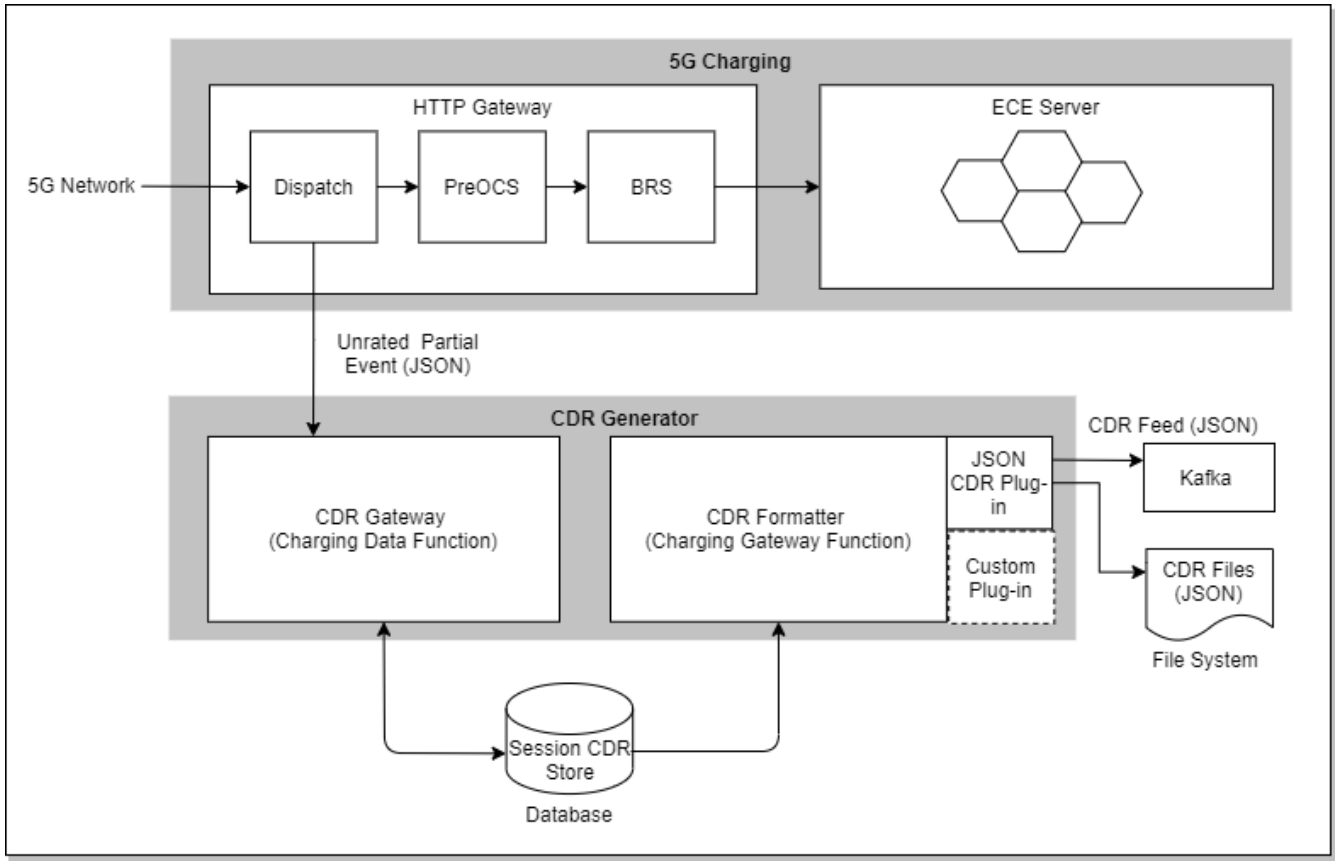
Copyright © 2019, 2026, Oracle and/or its affiliates.

b. Purges CDR records from the database after a specified amount of time. You can purge both processed and incomplete CDR records based on your configuration.

4. The CDR Formatter plug-in module generates CDR files. Depending on your configuration, it stores them on the disk or sends them to the Kafka messaging service.

Figure 23-1 shows the process flow for generating CDRs.

Figure 23-1 CDR Process Flow



Setting Up ECE to Generate CDRs

To set up ECE to generate CDRs:

1. Connect your 5G client to HTTP Gateway. See "Connecting ECE to a 5G Client".
2. Configure HTTP Gateway to route usage requests to the CDR Gateway. See "Configuring HTTP Gateway for CDR Generation".
3. Configure the CDR Gateway to generate CDRs and store them in the database. See "Configuring the CDR Gateway".
4. Configure the CDR Formatter to extract unrated CDRs from the database. See "Configuring the CDR Formatter".
5. Specify the plug-in to use for creating JSON formatted CDR files. See "Configuring the CDR Formatter Plug-in".

Copyright © 2019, 2026, Oracle and/or its affiliates.

Accessing ECE Configuration MBeans

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

1. Log on to the driver machine.
2. Start the ECE charging servers (if they are not started).
3. Connect to the ECE charging server node enabled for JMX management.
   This is the charging server node set to **start CohMgt = true** in the *ECE_home*/**config/eceTopology.conf** file, where *ECE_home* is the directory in which ECE is installed.
4. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
5. In the editor's MBean hierarchy, find the ECE configuration MBeans.

Configuring HTTP Gateway for CDR Generation

You can configure HTTP Gateway to send usage requests to the CDR Gateway by enabling CDR generation.

You can also configure HTTP Gateway to route usage requests to ECE Server for rating or the CDR Gateway based on the type of charging request:

- **Offline Charging Requests**: To send offline charging requests to ECE Server, set the **rateOfflineCDRinRealtime** attribute to **true**. To send them to the CDR Gateway, set the attribute to **false**.
- **Online Charging Requests**: To send online charging requests to ECE Server, set the **generateCDRsForOnlineRequests** attribute to **false**. To send them to the CDR Gateway, set the attribute to **true**.

To configure the HTTP Gateway:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.httpGatewayConfigurations**.
4. Expand **Attributes**.
5. Specify values for the fields in Table 23-1.

Table 23-1 Fields for Configuring CDR Generation

Name	Default	Description
cdrGenerationEnabled	"false"	If set to **true**, ECE generates CDRs according to your configuration. If set to **false**, ECE uses convergent charging for all charging requests. If set to **false**, no other settings in this area are relevant.
cdrGenerationStandaloneMode	"false"	If set to **true**, HTTP Gateway doesn't send any requests to ECE, but it still generates CDRs.
cdrGatewayList	"localhost:8084"	Specifies one or more servers for the CDR Gateway. Use a comma-separated list for multiple servers.
cdrGatewayRetry	"3"	The number of attempts for sending requests to CDR Gateway before giving up.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 23-1 (Cont.) Fields for Configuring CDR Generation

Name	Default	Description
rateOfflineCDRinRealtime	"false"	If set to **true**, HTTP Gateway sends offline charging requests to ECE for rating.
generateCDRsForOnlineRequests	"true"	If set to **true**, HTTP Gateway generates CDRs for all online charging requests.

Configuring the CDR Gateway

You configure the CDR Gateway to connect to your CDR storage database. You can also configure the CDR Gateway to generate either individual CDRs or aggregate multiple CDRs together according to trigger criteria that you specify.

To configure the CDR Gateway:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.cdrGatewayConfigurations**.
4. Expand **Attributes**.
5. Specify values for the fields in Table 23-2.

Table 23-2 Fields for Configuring CDR Gateway

Name	Default	Description
name	"cdrGateway1"	The name of this CDR Gateway instance. You can specify multiple CDR Gateways when configuring the HTTP Gateway.
primaryInstanceName	"cdrGateway1"	The name of the primary CDR Gateway instance.
schemaNumber	"1"	The number of the database schema used to store CDRs. Different CDR Gateways can write to different schemas.
isNoSQLConnection	"true"	The type of database used for storing CDRs: • **true**: Oracle NoSQL Database • **false**: Oracle Database
noSQLConnectionName	"@NO_SQL_CONNECTION@"	The connection name for the NoSQL database. This attribute applies only if you are using Oracle NoSQL Database for storing CDRs.
connectionName	"@ORACLE_PERSISTENCE_CONNECTION_NAME@"	The connection name of the Oracle database. This attribute applies only if you are using Oracle Database for storing CDRs.
cdrPort	"8084"	The port number for the CDR Gateway server.
cdrHost	"localhost"	The IP address, host name, or fully qualified domain name (FQDN) for the CDR Gateway server. **cdrHost** and **cdrPort** are also included in the **cdrGatewayList** field for the HTTP Gateway.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 23-2 (Cont.) Fields for Configuring CDR Gateway

Name	Default	Description
individualCdr	"false"	The type of CDR generation: • **true**: ECE generates an individual CDR for each event. • **false**: ECE aggregates requests until a trigger takes effect to write out the partial CDR or terminate the request. For more information, see "About Trigger Types".
cdrServerCorePoolSize	"32"	The number of threads in the CDR server pool.
cdrServerMaxPoolSize	"256"	The maximum number of threads allowed in the CDR server pool.
enableIncompleteCdrDetection	"false"	Whether to mark partially processed CDRs in the CDR Store as incomplete: • **true**: The CDR Formatter marks partially processed CDRs as incomplete. It also adds the **incompleteCDR** field to the CDR's **recordExtension** element. • **false**: Partially processed CDRs are not marked.
retransmissionDuplicateDetectionEnabled	"false"	Whether to detect and mark CDRs generated for retransmitted 5G CHF usage requests as possible duplicates: • **false**: The CDR Gateway does not mark possible duplicate usage. • **true**: When a usage request contains the **retransmissionIndicator** field set to true, ECE adds the **possibleDuplicateUsage** field set to true in the CDR's **recordExtension** element. If this attribute is set to true, the seqStore table must include the columns listed in Table 23-3

Table 23-3 seqStore Table

Columns	Data Type
SESSION_ID	**VARCHAR2(4000)**
LOCAL_SEQ_NO	**BLOB**
RECORD_SEQ_NO	**BLOB**
INCOMPLETE_CDR	**NUMBER**
INVOCATION_SEQ_NO	**BLOB**
PARTIAL_CLOSURE_TIMESTAMP	**NUMBER**

Configuring the CDR Formatter

You can configure the CDR Formatter to do the following:

- Retrieve completed CDRs from the CDR Store and pass them to a specified plug-in module for processing.
- Purge completed CDRs from the CDR Store that are older than a specified number of days.
- Purge orphan CDRs from the CDR Store. Orphan CDRs are incomplete CDRs that are older than a specified number of seconds. The CDR Gateway can create orphan CDRs if your ECE system goes down due to maintenance or failure.

Copyright © 2019, 2026, Oracle and/or its affiliates.

For example, if it is 12:00:00 and the configurable duration is 200 seconds, the CDR Formatter would purge from the CDR Store all incomplete CDRs that were last updated today at 11:56:40 or earlier.

- Mark partially processed CDRs as incomplete. This can occur in active-active disaster recovery systems when a site goes down while processing CDR sessions. See "About CDR Generator in an Active-Active System" in *BRM System Administrator's Guide* for more information.

To configure the CDR Formatter:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **cdrFormatters**.
4. Expand **Attributes**.
5. Specify values for the fields in Table 23-4.

Table 23-4 Fields for Configuring the CDR Formatter

Name	Default	Description
name	"cdrFormatter1"	The name of a CDR Formatter instance. You should name CDR Formatter instances consistently and uniquely (for example, cdrFormatter1, cdrFormatter2, and so on).
primaryInstanceName	"cdrFormatter1"	The name of the primary CDR Formatter instance.
schemaNumber	"1"	The number of the database schema for processing CDRs.
isNoSQLConnection	"true"	The type of database used for storing CDRs: • **true**: Oracle NoSQL Database • **false**: Oracle Database
noSQLConnectionName	"@NO_SQL_CONNECTION@"	The connection name for the NoSQL database. This attribute applies only if you are using Oracle NoSQL Database for storing CDRs. This is the same connection you use for CDR Gateway.
connectionName	"@ORACLE_PERSISTENCE_CONNECTION_NAME@"	The connection name of the Oracle database. This attribute applies only if you are using Oracle Database for storing CDRs. This is the same connection you use for CDR Gateway.
threadPoolSize	"6"	The number of threads used by the CDR Formatter instance to process a set of CDRs for each time range defined by **checkPointInterval**. Valid values are greater than zero and up to any number the system resources allow. Tune this value to the expected workload in the deployed environment.
retainDuration	"0"	The duration in seconds that processed CDRs are retained in the CDR Store before they can be purged. The default is **0**, which means that CDRs are purged immediately after being processed.
ripeDuration	"60"	The duration in seconds that CDRs must be stored in the CDR Store before the CDR Formatter can read them. Delaying the processing of CDRs up to the **ripeDuration** time allows time for resolving any duplicate CDRs that may have been persisted to the CDR Store.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 23-4 (Cont.) Fields for Configuring the CDR Formatter

Name	Default	Description
checkPointInterval	"6"	The time interval in seconds that the CDR Formatter instance waits before reading a batch of CDR information. This value must be: • Less than or equal to the value of **ripeDuration** • Evenly divisible by the number of threads configured for **threadPoolSize** The CDR Formatter doesn't read CDR information when the time interval is less than the **ripeDuration** interval.
pluginPath	"ece-cdrformatter.jar"	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.
pluginType	"oracle.communication.brm.charging.cdr.formatterplugin.internal.SampleCdrFormatterCustomPlugin"	The type of plug-in used to format CDRs.
pluginName	"cdrFormatterPlugin1"	The class name with the package path for the formatter plug-in object to be called by the CDR formatter. You can write a custom plug-in and specify it here.
noSQLBatchSize	"25"	The number of CDR records to be read from the NoSQL Database in a single read operation.
cdrStoreFetchSize	"2500"	The number of CDR records to retrieve from the CDR Store and hold in memory at a time.
cdrOrphanRecordCleanupAgeInSec	"200"	The amount of time, in seconds, at which an incomplete record in the CDR Store is considered an orphan. The CDR Formatter purges orphan records from the CDR Store that are this age or older.
cdrOrphanRecordCleanupSleepIntervalInSec	"200"	The sleep interval, in seconds, between each call to purge orphan CDRs from the CDR Store.
enableIncompleteCdrDetection	"false"	Whether to mark partially processed CDRs in the database as incomplete: • **true**: The CDR Formatter marks partially processed CDRs as incomplete. It also adds the **incompleteCDR** field to the CDR's **recordExtension** element. • **false**: Partially processed CDRs are not marked.

Configuring the CDR Formatter Plug-in

You can configure the CDR Formatter plug-in to create JSON-formatted CDR files and then store them in your file system or send them to your Kafka messaging service.

To configure the CDR Formatter plug-in:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **cdrFormatterPlugins**.
4. Expand **Attributes**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

5. Specify values for the fields in Table 23-5.

Table 23-5 Fields for Configuring the CDR Formatter Plug-in

Name	Default	Description
name	"cdrFormatterPlugin1"	Name of the formatting plug-in. **cdrFormatterPlugin1** is the default plug-in that comes with ECE, but you can also specify a custom plug-in.
tempDirectoryPath	"/tmp/tmp"	Path for CDR files being formatted.
doneDirectoryPath	"/tmp/done"	Path for formatted CDR files.
doneFileExtension	".out"	Extension for CDR files.
enableKafkaIntegration	"false"	Whether to send records to Kafka. Because ECE uses Kafka for notifications, you set it up during installation and configure the HTTP Gateway to use it. See "Creating Kafka Topics for ECE" in *ECE Installation Guide* and "Connecting ECE and Gateways to Kafka Topics".
enableDiskPersistence	"true"	Whether to save records as JSON files in the path specified for **doneDirectoryPath**.
maxCdrCount	"20000"	The maximum number of CDRs that can be written to a CDR output file.
enableStaleSessionCleanupCustomField	"false"	Whether to allow the plug-in to add a custom reason value to a CDR's **causeForRecordClosing** field. This field specifies why a CDR session was closed: • **true**: The custom value is allowed. It also adds the **staleSessionCleanup** field to the CDR's **recordExtension** element. • **false**: The custom value is not supported.
staleSessionCauseForRecordClosingString	"PARTIAL_RECORD"	The name of the custom reason for closing a CDR session. This value will be added to the CDR's **causeForRecordClosing** field.

About Trigger Types

If you configured ECE to aggregate events in the same CDR, ECE uses triggers to determine when a CDR remains open and when it closes. See:

- Triggers for Convergent Charging Events
- Triggers for Roaming Events

Triggers for Convergent Charging Events

When an event meets any of the trigger conditions in Table 23-6, the event is added to the open CDR. The CDR remains open with the same sequence number.

> **Note**
>
> If the request type is TERMINATE, the CDR is closed regardless of the trigger type.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 23-6 Triggers for Adding to an Open CDR

Category	Trigger Type
Change of Charging Conditions	• QOS_CHANGE • USER_LOCATION_CHANGE • SERVING_NODE_CHANGE • CHANGE_OF_UE_PRESENCE_IN_PRESENCE_REPORTING_AREA • CHANGE_OF_3GPP_PS_DATA_OFF_STATUS • HANDOVER_CANCEL • HANDOVER_START • HANDOVER_COMPLETE
Limit per Rating Group	• TIME_LIMIT (The expiration of the data time limit per rating group) • VOLUME_LIMIT (The expiration of the data volume limit per rating group) • EVENT_LIMIT (The expiration of the data event limit per rating group)
Quota Management Triggers	• QUOTA_THRESHOLD (The validity time, volume, or unit threshold has been reached) • QUOTA_EXHAUSTED (The validity expiration time, volume, or unit quota is exhausted) • FORCED_REAUTHORISATION (Reauthorization request by CHF)

When an event meets any of the trigger conditions in Table 23-7, it is added to the open CDR and then the CDR is closed.

Table 23-7 Triggers for Closing a CDR

Category	Trigger Type
Change of Charging Conditions	• UE_TIMEZONE_CHANGE • PLMN_CHANGE PLMN • RAT_CHANGE RAT • SESSION_AMBR_CHANGE • REMOVAL_OF_UPF • MANAGEMENT_INTERVENTION • ADDITION_OF_ACCESS **Note:** If the Change of Charging condition is true for any trigger that comes at the rating group level, its information is added into a new usedUnitContainer block. It is not aggregated into the existing usedUnitContainer block.
Limit per PDU Session	• TIME_LIMIT (The expiration of data time limit per PDU session) • VOLUME_LIMIT (The expiration of data volume limit per PDU session) • EVENT_LIMIT (The expiration of data event limit per PDU session) • MAX_NUMBER_OF_CHANGES_IN_CHARGING_CONDITIONS (The maximum number of charging condition changes has expired)

The following sample CDR shows a trigger type of **MANAGEMENT_INTERVENTION**, which causes ECE to add the information to the open CDR and then close the CDR:

```
"quotaManagementIndicator": "ONLINE_CHARGING",
"triggers": [
   {
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
      "triggerType": "MANAGEMENT_INTERVENTION",
      "triggerCategory": "IMMEDIATE_REPORT"
   }
],
```

Triggers for Roaming Events

When a roaming event meets any of the trigger conditions in Table 23-8, it is added to the open CDR.

Table 23-8 Triggers for Adding to an Open CDR (Roaming Event)

Category	Trigger Type
Change of Charging Conditions	• QOS_CHANGE • USER_LOCATION_CHANGE • SERVING_NODE_CHANGE • CHANGE_OF_UE_PRESENCE_IN_PRESENCE_REPORTING_AREA • CHANGE_OF_3GPP_PS_DATA_OFF_STATUS • TARIFF_TIME_CHANGE
Limit per QoS Flow	• TIME_LIMIT (Expiration of data time limit per QoS Flow) • VOLUME_LIMIT (Expiration of data volume limit per QoS Flow)
Others	• TIME_LIMIT (Expiration of data time limit per QoS Flow) • VOLUME_LIMIT (Expiration of data volume limit per QoS Flow)

When a roaming event meets any of the trigger conditions in Table 23-9, it is added to the open CDR, and the CDR is then closed.

Table 23-9 Triggers for Closing a CDR (Roaming Event)

Category	Trigger
Change of Charging Conditions	• UE_TIMEZONE_CHANGE • PLMN_CHANGE • RAT_CHANGE • SESSION_AMBR_CHANGE • REMOVAL_OF_UPF • MANAGEMENT_INTERVENTION
Limit per PDU Session	• TIME_LIMIT (The expiration of data time limit per PDU session) • VOLUME_LIMIT (The expiration of data volume limit per PDU session) • EVENT_LIMIT (The expiration of data event limit per PDU session) • MAX_NUMBER_OF_CHANGES_IN_CHARGING_CONDITIONS (The maximum number of charging condition changes has expired)

The following sample CDR for a roaming event shows a trigger type of **QOS_CHANGE**, which causes ECE to add the information to the open CDR and keep the CDR open:

```
"quotaManagementIndicator": "ONLINE_CHARGING",
"triggers": [
   {
      "triggerType": "QOS_CHANGE",
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
            "triggerCategory": "IMMEDIATE_REPORT"
        }
    ],
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

24

Connecting ECE to a Diameter Client

You can set up network integration for online charging by using the Oracle Communications 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
- Configuring Diameter Gateway Nodes for Online Charging
- Checking for Client-Side Timeouts in Diameter Gateway
- Configuring ECE to Publish Notifications to Diameter Gateway
- Adding Custom AVPs for Usage Requests
- Using Incremental or Cumulative Accounting for Usage Requests
- Configuring WebLogic Queues for Notifications
- Including Loan Sub-Balance in Balance Queries
- Enriching Balance Query Responses with Subscriber Preferences
- About Handling Requests When Charging Servers Are Unavailable
- Cleaning Sy Sessions from the Cache

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 notifications from the ECE Notification queue or topic, create Diameter notification messages from them, and send the notification messages to the appropriate Diameter clients.

For information about conformance with industry standards, see *ECE Diameter Gateway Protocol Implementation Conformance Statement*.

To implement Diameter Gateway:

1. Create Apache topics and connect ECE to them. See "Creating Kafka Topics for ECE Integration".

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. For all of the 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. Configure Diameter Gateway to retrieve Sp and Sy information. See "Network Integration for Sp and Sy Interface (Policy) Requests".
4. For Gy interface Diameter requests, ensure that you have done the following:
   - Defined any custom service-event mappings in PDC.
   - Edited 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".
5. Configure the Diameter Gateway nodes for online charging. See "Configuring Diameter Gateway Nodes for Online Charging".
6. Configure the Diameter Gateway to terminate requests that have timed out at the client. See "Checking for Client-Side Timeouts in Diameter Gateway".
7. Configure ECE to send your desired notification types to Diameter Gateway. See "Configuring ECE to Publish Notifications to Diameter Gateway".
8. (Optional) Configure Diameter Gateway to add custom AVPs to usage requests. See "Adding Custom AVPs for Usage Requests".
9. (Optional) Configure whether Diameter Gateway uses incremental accounting logic, cumulative accounting logic, or both. See "Using Incremental or Cumulative Accounting for Usage Requests".
10. (Optional) Configure ECE and Diameter Gateway to use WebLogic queues, rather than Kafka topics, for notifications. See "Configuring WebLogic Queues for Notifications".
11. (Optional) Configure Diameter Gateway to include both the main and loan balance information in balance query responses. See "Including Loan Sub-Balance in Balance Queries".
12. (Optional) Configure ECE to include subscriber preferences, such as product lifecycle state, in balance query responses returned to Diameter Gy clients. See "Enriching Balance Query Responses with Subscriber Preferences".
13. 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 start up, the Diameter Gateway instances read from the ECE Notification queue or topic 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).

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

Sp/Sh:

- User-Data-Request (UDR)
- Subscribe-Notifications-Request (SNR)
- Push-Notification-Request (PNR)

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 WebLogic Queues for Notifications". 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 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

  - Cancel
- Price inquiry
- 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 one service. Contain the following Gy interface request types in an MSCC group:

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

  > **Note**
  >
  > When Diameter Gateway receives a CCR-U message with **Reporting-Reason** set to **QHT**, it processes the request as an Update operation. The QHT reporting reason does not terminate the active session or release reservations. The session remains active until a later CCR-U message updates the session or a CCR-T message terminates it.

  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.

Termination processing and reservation release occur when Diameter Gateway receives a terminating request, such as CCR-T. They do not occur when Diameter Gateway receives a CCR-U with Reporting-Reason set to QHT.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

> **Note**
>
> - For Diameter Gy usage requests, ECE derives the request time zone from the 3GPP-MS-TimeZone AVP (MS-TimeZone-3GPP in ECE Diameter Gateway configuration and message syntax). ECE uses the AVP UTC offset, including the encoded daylight saving time adjustment, to calculate time-zone-sensitive balance validity, such as calendar-day and daily bucket validity. This includes daily buckets granted through conditional RUMs.
> - Ensure that the 3GPP-MS-TimeZone AVP specifies the expected UTC offset for the subscriber or network request. If the AVP is absent, Diameter Gateway uses UTC. The TZ environment settings of the ECE, BRM, and PDC servers do not override the value in the AVP for Diameter Gy usage requests.

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, 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**

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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*/**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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

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

You can also specify a time zone. For example,

```
|   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***/config/management/migration-configuration.xml** file is the directory where you save your configuration data.
6. Run the **configLoader** utility:

   ```
   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 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***/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

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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***/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 multiple 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, and 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**.

   The file is in the directory specified by the **configObjectsDataDirectory** parameter in the *ECE_home***/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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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].

  – **"END_USER_NAI"**

    The identifier is in the form of a Network Access Identifier, as defined in [NAI].

  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.

  ```
  DiameterMediationTable {
      Service-Context-Id | Service-Identifier | Rating-Group | ProductType |
  EventType | Version | Subscription-Id-Type | ValidFrom |
      "gy.service@example.com" | "1" | "10" | "VOICE" | "V_USAGE" | 1.0 |
  "END_USER_NAI" | "2012-12-31T12:01:01 PST" |
      "gy.service@example.com" | "1" | "11" | "DATA" | "D_USAGE" | 1.0 |
  "END_USER_IMSI" | "2012-12-31T12:01:01 PST" |
  }
  ```

- 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.

  ```
  DiameterMediationTable {
      Service-Context-Id | Service-Identifier | Rating-Group | ProductType |
  EventType | Version | Subscription-Id-Type | ValidFrom |
      "gy.service@example.com" | "1" | "12" | "DATA" | "D_USAGE" | 1.0 |
  "END_USER_SIP_URI, END_USER_IMSI" | "2012-12-31T12:01:01 PST" |
  }
  ```


Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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.

```
DiameterMediationTable {
    Service-Context-Id | Service-Identifier | Rating-Group | ProductType |
EventType | Version | Subscription-Id-Type | ValidFrom |
    "gy.service@example.com" | "1" | "13" | "SMS" | "S_USAGE" | 1.0 | "" |
"2012-12-31T12:01:01 PST" |
}
```

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).

Configuring Diameter Gateway Nodes for Online Charging

To configure Diameter Gateway node to process online charging requests:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.diameterGatewayConfigurations.***Instance_Name*, where *Instance_Name* is the name of the instance to configure.
4. Expand **Attributes**.
5. Specify values for all the attributes required to configure the instance.

   See Table 24-1 for attribute descriptions and default values.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 24-1 Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
name	"diameterGateway1"	The name of the Diameter Gateway instance. Name Diameter Gateway node instances consistently and uniquely (for example, **diameterGateway1**, **diameterGateway2**, and so on. If you change the name of an existing instance, you must update the name in the *ECE_home***/config/eceTopology.conf** file.
clusterName	""	The cluster name of the Diameter Gateway instance. ECE uses both **name** and **clusterName** to identify a Diameter Gateway instance. You must specify a different cluster name if you are configuring Diameter Gateway nodes with the same name. **Note**: Ensure that this cluster name matches the cluster name in the *ECE_home***/config/charging-coherence-override-secure-prod.xml** file.
diameterTrafficPort	"3868"	The port (on the physical host computer running the Diameter Gateway node instance) the Diameter Gateway instance listens on to handle Diameter messages. When adding new Diameter Gateway instances, choose a port number that is not used by another application. When multiple Diameter Gateway instances run on the same physical host computer, each instance must use a different port number.
ioThreadPoolSize	"10"	The number of threads used by the network I/O thread pool. The valid values are greater than zero and up to any number the system resources allow.
responseTimeout	"10"	The maximum duration, in seconds, the Diameter Gateway instance waits for a response from the Diameter client. If the Diameter Gateway instance does not receive a response from the Diameter client within the specified duration, the Diameter Gateway instance stops waiting for a response and removes the notification from the JMS queue. The valid values are greater than zero and up to any number the system resources allow.
watchDogInterval	"30"	The duration in seconds that the Diameter Gateway instance waits before it issues a Device-Watchdog-Request message (DWR).
originHost	No default value Setting a value for this field is mandatory.	The Origin-Host attribute-value pair (AVP) to be sent in the Diameter request. You assign this unique identifier to your Diameter Gateway server on its host. It can be any string value.
originRealm	No default value Setting a value for this field is mandatory.	The Origin-Realm AVP to be sent by the Diameter Gateway in outgoing Diameter requests. You assign this signalling realm (domain) to your Diameter Gateway server. You must set the same origin realm value for all Diameter Gateway instances in the same ECE topology.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 24-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
diameterTrafficHost	""	The network interface (on the physical or virtual host computer running the Diameter Gateway node instance) that the Diameter Gateway node binds to and listens on for Diameter messages. The value can be an IP address or a host name. The value can also be an empty string (the default). In that case, the Diameter Gateway instance listens for Diameter messages on all network interfaces available on the server.
diameterTrafficHostSctp	""	When SCTP is used, the network interface (on the physical or virtual host computer running the Diameter Gateway node instance) that the Diameter Gateway node binds to and listens on for Diameter messages (sent from Diameter clients). The value can be: • One or more SCTP IP addresses. For multihoming systems, separate multiple IP addresses with a comma (,). For example: `10.240.179.147,10.240.182.149` • One or more host names. • An empty string, which disables the SCTP transport. Your operating system must support SCTP to use this configuration. If not, install the SCTP system package for your operating system version.
loopback	"false"	Specifies the loopback setting for performance testing. The valid values are: • **True** specifies that the Diameter Gateway instance does not send the credit-control request to ECE. Instead, the Diameter Gateway instance returns the success result code to the network element. • **False** specifies that the Diameter Gateway instance sends the credit-control request to ECE.
requestProcessorThreadPool Size	"10"	The number of threads used by the request-processor thread pool. The request-processor thread pool is a Diameter Gateway thread pool dedicated to processing Diameter requests handed off to it from the I/O thread pool. The valid values are greater than zero and up to any number the system resources allow.
requestProcessorBatchSize	"10"	The batch size of the Diameter requests handed off by the network I/O thread pool to the request-processor thread pool. The valid values are greater than zero and up to any number the system resources allow.
notificationThreadPoolSize	"10"	The number of threads used by the Diameter Gateway instance to process notification messages. The 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 24-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
maxNotificationCommitSize	"100"	The maximum number of dequeued notification messages from the JMS topic that can remain uncommitted. If the number of dequeued notification messages from the JMS topic exceeds this number, the Diameter Gateway instance stops reading messages until the read messages are committed.
maxWaitTimeNotificationCommit	"3600000"	The maximum wait time for which the uncommitted messages can remain in the notification JMS topic. **Note:** This only applies for deployments using WebLogic Server for Notifications (not Kafka).
ccFailover	"FAILOVER_SUPPORTED"	Indicates if the Diameter Gateway instance operates in a cluster that supports failover. The valid values are: • **FAILOVER_SUPPORTED** • **FAILOVER_NOT_SUPPORTED** The value set here is the value the Diameter Gateway instance sends for the CC-Session-Failover AVP in all credit-control answers (CCAs) it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.4
creditControlFailureHandling	"RETRY_AND_TERMINATE"	Indicates how the Diameter client should proceed if a CCA is not received before the Tx timeout. The valid values are: • **TERMINATE** • **CONTINUE** • **RETRY_AND_TERMINATE** The value set here is the value the Diameter Gateway instance sends for the Credit-Control-Failure-Handling AVP in all CCAs it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.14
directDebitingFailureHandling	"TERMINATE_OR_BUFFER"	Indicates how the Diameter client should proceed if a Direct Debit CCA is not received before the transaction timeout. The valid values are: • **TERMINATE_OR_BUFFER** • **CONTINUE** The value set here is the value the Diameter Gateway instance sends for the Direct-Debiting-Failure-Handling AVP in all credit-control answers (CCAs) it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.15
allowMultipleConnectionsPerPeer	"false"	Allows a Diameter Gateway to accept multiple connections from the same diameter peer identity for high availability. Any outgoing diameter requests for the peer will be load balanced across the multiple connections.
sctpMaxInStream	"0"	The maximum number of incoming streams allowed per association. The valid range is **0** to **65535**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 24-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
sctpMaxOutStream	"0"	The maximum number of outgoing streams allowed per association. The valid range is **0** to **65535**.
sctpSendBufferSize	"0"	The buffer size for sending data (in bytes). The valid range is **0** to **2147483647**. This attribute determines the maximum amount of data that can be sent in a single transaction. You can increase or decrease the size based on your requirements.
sctpReceiveBufferSize	"0"	The buffer size for receiving data (in bytes). The valid range is **0** to **2147483647**. This attribute determines the SCTP receiver window size. You can increase or decrease the size based on your requirements. For example, you can increase the size for high-volume connections or decrease the size if you want to limit the incoming data backlog.
notificationListenerConnectionPoolSize	"10"	The connection pool size for notification listeners.
kafkaPartition	"4"	The dedicated Kafka partition ID consumed by the Diameter Gateway for Kafka partition-based notification routing. The partition ID must be unique across gateways and cannot overlap with HTTP Gateway replica partitions. **Note:** For Kafka topic-based routing, this attribute is ignored.
dgwNotifNetworkResponseTimeoutAdder	"200"	The interval, in milliseconds, the Diameter Gateway instance waits for a response from the notification topic.
ingressRequestTimeoutMs	"3000"	How long the client waits for a response, in milliseconds, before timing out. After this amount of time, the Diameter Gateway terminates the request. The default is **3000**. **Note:** Set this to a slightly lower value than the client's timeout to account for the time spent traversing the network.
enableNetworkExpiration	"true"	Whether to calculate the ingress request timeout using the time stamp contained in the request payload (**true**) or the time the message arrived at the Diameter Gateway (**false**). The default is **true**.
rarNotificationAgeLimitInSec	"300"	The time limit, in seconds, after which any Gy notification consumed by the Diameter Gateway is considered expired and delivery attempts for the Gy notification to the Gy client are halted.
snrNotificationAgeLimitInSec	"300"	The time limit, in seconds, after which any Sy notification consumed by the Diameter Gateway is considered expired and delivery attempts for the Sy notification to the Sy client are halted.

Checking for Client-Side Timeouts in Diameter Gateway

Client applications can contain their own request timeouts that activate when ECE takes too long to respond. When this occurs, the client application abandons the request, but ECE may continue to process the request and send a response, which the client discards. This extra processing is unnecessary and can be detrimental to ECE production systems approaching overload conditions. To improve performance, you can configure the Diameter Gateway to terminate requests that have timed out on the client side.

Copyright © 2019, 2026, Oracle and/or its affiliates.

To configure the Diameter Gateway to check for client-side timeouts:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.diameterGatewayConfigurations.***name*, where name is the name of the Diameter Gateway instance.
4. Expand **Attributes**.
5. Specify timeout values for the Diameter Gateway instance:
   - **ingressRequestTimeoutMs**: Specifies how long the client waits for a response, in milliseconds, before timing out. After this amount of time, the Diameter Gateway terminates the request. The default is **3000**.

     > **Note**
     >
     > Set this to a slightly lower value than the client's timeout to account for the time spent traversing the network.

   - **enableNetworkExpiration**: Specifies whether to calculate the ingress request timeout using the time stamp contained in the request payload (**true**) or the time the message arrived at the Diameter Gateway (**false**). The default is **true**.
6. Expand **charging.brsConfigurations.***name*, where *name* is the name of the BRS instance.
7. Expand **Attributes**.
8. Specify timeout values for the BRS instance:
   - **requestMaxRetry**: Specifies the maximum number of time the BRS retries sending the ingress request to ECS. Use a value between 1 and 5. The default is 3.
   - **requestRetryWaitTimeMs**: Specifies the amount of time, in milliseconds, between each retry attempt. The default is 5.

When a request is terminated, the Diameter Gateway logs an error and sets the **ResponseConsumerFailureCode** field to **Expired** in the response to the client.

Configuring ECE to Publish Notifications to Diameter Gateway

You can enable ECE to send SNR and RAR notifications to the ECE Notification topic, where they will be retrieved by the Diameter Gateway.

To do so:

1. Configure a unique dedicated Kafka partition for each Diameter Gateway instance. See "Configuring Diameter Gateway Nodes for Online Charging".

   > **Note**
   >
   > For Kafka partition-based routing, each Diameter Gateway instance must have a unique dedicated Kafka partition. Do not reuse a partition ID that is assigned to another gateway or to an HTTP Gateway replica.

Copyright © 2019, 2026, Oracle and/or its affiliates.

2. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
3. Expand the **ECE Configuration** node.
4. Expand **charging.server**.
5. Expand **Attributes**.
6. Set the **kafkaEnabledForNotifications** property to **true**.
7. Expand **charging.notification**.
8. Expand **Attributes**.
9. Set the **rarNotificationMode** and **spendingLimitNotificationMode** properties to one of the following:
   - **NONE**: ECE does not send this notification type.
   - **ASYNCHRONOUS**: ECE sends asynchronous notifications to the ECE notification topics.

     If configured to do so, the Diameter Gateway will consume notifications from the ECE notification topics and dispatch them through a REST API call.

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***/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 the active session duration based on the units 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 creating your usage request builder, specify the accounting behavior 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> 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 the 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*/**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     | ""                   | "2024-3-31T12:01:01 PST" |
   "INCREMENTAL"     |
   "gy.service@example.com" | "1"                | "11"         | "DATA"      |
   "D_USAGE" | 1.0     | ""                   | "2024-3-31T12:01:01 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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. Go to the *ECE_home*/**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 "Starting and Stopping ECE" in *BRM System Administrator's Guide*.

Configuring WebLogic Queues for Notifications

To configure Diameter Gateway to listen for notifications from ECE, you must specify the types of notifications that ECE generates.

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".

To enable Diameter Gateway to consume notifications from an ECE notification queue set up in WebLogic:

> **Note**
>
> The following steps assume that ECE is installed and that the required ECE postinstallation tasks have been completed.

1. On the Oracle WebLogic server, verify that the ECE 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.
3. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing 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" for information.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 in a JMX editor, such as JConsole. See "Accessing 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. Go to the *ECE_home***/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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **DiameterGateway** node.
3. Expand **PeerConnectionsTracker**.
4. Expand **Attributes**.
5. Click the **peerConnections** value.

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

Including Loan Sub-Balance in Balance Queries

By default, Diameter Gateway includes information about the main currency balance in balance query responses. You can configure Diameter Gateway to include both the main and

Copyright © 2019, 2026, Oracle and/or its affiliates.

loan balance information in balance query responses. To configure it to do so, in the ECE Balance Java API, set the **balancequerymode** to **TURBO**. For example:

```
ORA-Balance-Query-Mode (248,VM,v=3512,I=16) = TURBO_MODE (4)
```

For more information about **balancequerymode**, see the documentation for **oracle.communication.brm.charging.messages.query** in *Elastic Charging Engine Java API Reference*.

Enriching Balance Query Responses with Subscriber Preferences

You can configure ECE to include subscriber preferences, such as product lifecycle state, in balance query responses returned to Diameter Gy clients.

To enrich balance query responses with product preference and lifecycle state information:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.notification**.
4. Expand **Attributes**.
5. Set the **enrichBalanceQueryResponseMode** attribute to **PIGGYBACK**.
6. Under **charging.notification**, expand **notificationEnrichment**.
7. Expand **Attributes**.
8. Configure the notification enrichment list to include either only the product lifecycle state or all subscriber preferences:
   - To include only the lifecycle state:
     - Set **enrichName** to **subscriberPreferences**
     - Set **enrichValue** to **LifecycleState**
   - To include all subscriber preferences:
     - Set **enrichName** to **subscriberPreferences**
     - Set **enrichValue** to **ALL**
9. Map the enriched data to the Diameter CCA. Extend ECE at the **diameterGyExtension** extension point. For details, see the **SampleDiameterGyExtension** ECE SDK sample program.

About 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

threshold), Diameter Gateway sends the DIAMETER_TOO_BUSY result code response to network requests.

Cleaning Sy Sessions from the Cache

You can free system resources and improve performance by cleaning stale Sy sessions from the Sy session cache. Ongoing Sy sessions can become stale if ECE never receives the session termination request (STR) or if the session fails but the session entry remains in the cache.

ECE performs Sy stale-session cleanup on the ECE server (ECS). When this functionality is enabled, ECE deletes stale Sy sessions when it receives a new SLR Initial request for the same subscriber. The server-side implementation removes the need for full-cache scans during SLR Initial processing and reduces request latency.

> **Note**
>
> You can open one Sy session per service at any time. If a new session is initiated during an ongoing session, the system cleans the older session and uses the latest Sy session.

To clean stale Sy Sessions from the cache for on-premises systems:

- To change the value with a restart:
  1. In your **charging-settings.xml** file, set the **enableSySessionCleanup** element to **true**.
  2. Restart the ECE server for the change to take effect.
- To change the value without restarting:
  1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
  2. Expand the **ECE Configuration** node.
  3. Expand **charging.server**.
  4. Expand **Attributes**.
  5. Set the **enableSySessionCleanup** attribute to **true**.

To clean stale Sy Sessions from the cache for cloud native systems:

1. In your **override-values.yaml** file for **oc-cn-ece-helm-chart**, set the **charging.server.enableSySessionCleanup** key to **true**.
2. Run the **helm upgrade** command or run a **chargingConfigurationReloader** job. See "Reloading ECE Application Configuration Changes" in *BRM Cloud Native System Administrator's Guide* for more information.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> When the **subscriptionIdOnStrEnabled** configuration MBean is set to **true**, the Diameter Gateway relies on the PCF to send subscription IDs and the Sy session cache is not used. ECE does not perform a cache lookup or prime the cache. In this case, set **enableSyCleanup** to **false**. When **subscriptionIdOnStrEnabled** is set to **false** (default behavior), the Sy session cache is used.

Copyright © 2019, 2026, Oracle and/or its affiliates.

25

Connecting ECE to a RADIUS Client

You can use Oracle Communications Elastic Charging Engine (ECE) RADIUS Gateway for authenticating access requests, processing accounting requests from RADIUS clients, such as terminal servers or network access servers (NAS), and processing disconnect requests from the RADIUS server to RADIUS clients.

Topics in this document:

- Overview of Authentication, Accounting, and Disconnection Using RADIUS Gateway
- About RADIUS Gateway Authentication
- About RADIUS Gateway Accounting
- About RADIUS Gateway Disconnection
- Setting Up Authentication, Accounting, and Disconnection Using RADIUS Gateway

For information about conformance with industry standards, see *ECE RADIUS Gateway Protocol Implementation Conformance Statement*.

Overview of Authentication, Accounting, and Disconnection Using RADIUS Gateway

You use RADIUS Gateway for authenticating access requests, processing accounting requests, and processing disconnect 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:

- Translates the request into an ECE request.
- Submits the ECE request to the ECE server.
- Receives the ECE response from the ECE server and translates it into a RADIUS message response.
- Sends the RADIUS message response to the RADIUS client.
- Consumes notifications from the ECE notification topic, constructs request messages, and sends the request messages to the RADIUS client.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

> **Note**
>
> You can customize the RADIUS Gateway to fetch subscriber profile information, including customer and balance data, enabling its use for decision-making during the authentication process. See "Authentication Extension Points" for more information.

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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: 192.0.2.1
 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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="0xFBFCE5676F94433682718EF97F8AB24900"
 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-

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

Sample Access-Request Request for EAP-MD5 Authentication

```
Code: Access-Request(1)
 Identifier: 0
 Length: 120
 Authenticator: 0x7D564C041FD183A4DBA037E03E3244F3
 User-Name: BOB
 NAS-IP-Address: 127.0.0.1
 Calling-Station-Id: 02-00-00-00-00-01
 Framed-MTU: 1400
 NAS-Port-Type: 19
 Connect-Info: CONNECT 11Mbps 802.11b
 Service-Type: 2
 EAP-Message: 0x0200000801424F42
 Message-Authenticator: 0x4FB1186DDA9643CED0CD13D59ECD9D4E
```

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

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]
Radius Response Packet
[Code: Accounting-Response(5)
 Identifier: 1
 Length: 20
 Authenticator: 0x00000000000000000000000000000000]
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

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 indicates that it is 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
 Service-Type: 1
Radius Response Packet
[Code: Accounting-Response(5)
 Identifier: 5
 Length: 20
 Authenticator: 0x00000000000000000000000000000000]
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

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

  > **Note**
  >
  > The Accounting-Interim-Update 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.

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]
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

About RADIUS Gateway Disconnection

The RADIUS server sends a disconnect request to terminate a user session that is in progress. The Disconnect-Request message specifies the user name to disconnect, the IP address, and the port.

RADIUS Gateway processes disconnect requests to immediately end a customer's usage session as follows:

1. A user or administrator requests to end an active session.
2. BRM submits a disconnect request to ECE containing the user name to disconnect.
3. RADIUS Gateway creates a Disconnect-Request message containing details like the user name, the NAS IP address, and the port.
4. The RADIUS server sends the Disconnect-Request message to the NAS.
5. The NAS processes the Disconnect-Request message and does one of the following:
   a. If processing is successful, the NAS ends the session and returns a Disconnect-ACK message to RADIUS Gateway.
   b. If processing is not successful, the NAS does not end the session and returns a Disconnect-NAK message to RADIUS Gateway.
6. If RADIUS Gateway does not receive a response from the NAS, the RADIUS Gateway retransmits the Disconnect-Request message until it does.

Sample Disconnect-Request Message

```
Code: Disconnect-Request(40)
  Identifier: 0
  Length: 120
  Authenticator: AuthCode
  User-Name: JSmith
  NAS-IP-Address: 192.0.2.1
  NAS-Port: 7777
```

Sample Disconnect-ACK Message

```
Code: Disconnect-ACK(41)
  Identifier: 0
  Length: 104
  Authenticator: AuthCode
  Acct-Terminate-Cause: 6
```

Sample Disconnect-NAK Message

```
Code: Disconnect-NAK(42)
  Identifier: 0
  Length: 97
  Authenticator: AuthCode
  Error-Cause: 402
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

Setting Up Authentication, Accounting, and Disconnection Using RADIUS Gateway

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

1. Add additional RADIUS Gateway nodes required for your topology and configure each instance. See "Adding RADIUS Gateway Nodes" in *BRM System Administrator's Guide*.
2. Load your event definitions from PDC into ECE. See "Enabling Charging for Custom Events" in *PDC Creating Product Offerings*.
3. Load customer data and data keys from BRM into ECE. See "Loading Data Keys Extracted from BRM into ECE".
4. (Optional) Customize the RADIUS data dictionary to include custom vendor-specific attributes. See "Customizing the RADIUS Data Dictionary".
5. Load the RADIUS mediation specification data. See "Loading the RADIUS Mediation Specification Data".
6. Map RADIUS network attributes to event attributes. See "About Mapping RADIUS Network Attributes to Event Attributes".
7. (Optional) Configure ECE to send disconnect messages through the RADIUS Gateway to your RADIUS client. See "Configuring ECE and RADIUS Gateway to Manage Disconnect Requests".
8. 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.

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 ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **UpdateEventhandler** node.
3. Expand **Update**.
4. Expand **Operations**.
5. Select **updateDataKeys**.
6. Click the **updateDataKeys** button.

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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 *ECE_home***/config/radius/radiusDictionary.xml**.

> **Note**
>
> Do not remove, rename, or move the RADIUS data dictionary file to a different location.

Creating a Custom Data Dictionary

You can create a custom data dictionary file by using the *ECE_home***/config/radius/radiusDictionary.xml** file as a template. The default location for your custom data dictionary file is *ECE_home***/config/radius/custom/***dictionary_file*, where *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".

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.

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:

```
<?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>
```

Table 25-1 lists the vendor-specific attribute values and descriptions.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 25-1 Vendor-Specific Attribute Values

Parameters	Description
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: • 9 (Cisco) • 10415 (3GPP) • 2636 (Juniper)
vendor_name	Name of the vendor.
attribute_name	Name of the attribute. This must be unique. **Important:** Do not use the same attribute name as used in the default RADIUS data dictionary file. Using the same attribute name in the custom data dictionary file overrides the attribute values in the default RADIUS data dictionary file.
attribute_ID	Identification number assigned to the attribute in the dictionary.
data_type	Any one of the following data types: • **UnsignedInt** 32-bit unsigned value in big endian order (high byte first). • **Integer** 32-bit value in big endian order (high octet first). • **String** 0-253 octets • **Ipaddr** 4 octets in network octet order • **Binary** 0-254 octets • **Password** (n * 16) (>= 16) octets. This field is encrypted according to the User-Password AVP in RFC 2865. • **Short** 16-bit value • **Octet** 8-bit value • **ifid** IPv6 interface ID • **ipv6addr** IPv6 address • **date** Linux timestamp in seconds (since January 1, 1970 GMT)

Loading the RADIUS Mediation Specification Data

RADIUS Gateway uses the RADIUS mediation specification data to determine which product and event type combination and network mapping applies to an incoming request from the RADIUS client.

To load the RADIUS mediation specification data:

Copyright © 2019, 2026, Oracle and/or its affiliates.

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

   A sample mediation specification file (*ECE_home***/sample_data/config_data/specifications/ece_simple**) is available.

   > **Note**
   >
   > Create only one RADIUS mediation specification file to represent the mediation specification for RADIUS Gateway.

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***-***dd***T***hh***:***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.

4. Open the *ECE_home***/config/management/migration-configuration.xml** file.

5. Search the **configObjectsDataDirectory** parameter and copy the value. For example:

   ```
   configObjectsDataDirectory = ECE_home/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:

   ```
   start configLoader
   ```


Copyright © 2019, 2026, Oracle and/or its affiliates.

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 25-1 Sample RADIUS Mediation Specification Entry

```
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***/config/management/migration-configuration.xml** file

   A sample network mapping file is available in the (*ECE_home***/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:

Copyright © 2019, 2026, Oracle and/or its affiliates.

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:

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

5. Save and close the file.

> **Note**
>
> Verify that the name of this network mapping file is specified in the RADIUS mediation specification file.
>
> For BRM cloud native deployments, the pricingupdater pod and the configloader pod do not share the generated network mapping files anymore. After you save your changes, make the **network_mappings** directory available to the configloader pod and restart the pod to load the files. See "Loading RADIUS Network Mapping Files in ECE Cloud Native" in *BRM Cloud Native System Administrator's Guide*.

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.

Configuring ECE and RADIUS Gateway to Manage Disconnect Requests

You configure ECE to send disconnect messages through the RADIUS Gateway to your RADIUS client by doing the following:

1. Customize ECE to generate a **UserDisconnectServiceEvent** event whenever triggering events occur.
2. Connect ECE and the RADIUS Gateway to the ECE notification topics. See "Setting Up ECE and Gateways for Apache Kafka Using Partition-Based Routing".

Copyright © 2019, 2026, Oracle and/or its affiliates.

3. Connect RADIUS Gateway to the ECE notification topics. See "Configuring RADIUS Gateway to Send Disconnect Messages to Your RADIUS Client".
4. Configure ECE to send notifications to RADIUS Gateway through the ECE notification topics. See "Configuring ECE to Publish Notifications to RADIUS Gateway".

Configuring RADIUS Gateway to Send Disconnect Messages to Your RADIUS Client

To configure RADIUS Gateway to send disconnect messages to your RADIUS client:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.radiusGatewayConfigurations**.
4. Expand **Attributes**.
5. Specify values for the fields in Table 25-2.

Table 25-2 Fields for Disconnect Messages

Name	Default	Description
disconnectMessageRetryThreadPoolSize	""	The number of threads used by the RADIUS Gateway to send disconnect messages to the RADIUS client.
disconnectMessageMaxRetryCount	""	The maximum number of times the RADIUS Gateway can retry a connection to the RADIUS client.
disconnectMessageInitialRetryDelayMillis	""	The initial delay, in milliseconds, RADIUS Gateway waits before retrying a connection to the RADIUS client.
disconnectMessageConsecutiveRetryDelayMillis	""	The amount of time, in milliseconds, RADIUS Gateway waits between two consecutive retries.
kafkaBatchSize	""	The size of the Kafka batch.
notificationListenerConnectionPoolSize	""	The size of the connection pool required to process notifications from the ECE notification topics.

Configuring ECE to Publish Notifications to RADIUS Gateway

You can configure ECE to send disconnect request notifications to the ECE RADIUS notification topic, where they can be retrieved by the RADIUS Gateway. To do so:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **kafkaEnabledForNotifications** property to **true**.
6. Expand **charging.notification**.
7. Expand **Attributes**.
8. Set the **UserDisconnectServiceEventMode** property to one of the following:
   - **NONE**: ECE does not send this notification type.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **ASYNCHRONOUS**: ECE sends asynchronous disconnect request notifications to the ECE RADIUS notification topic.

Copyright © 2019, 2026, Oracle and/or its affiliates.

26

Configuring Policy-Driven Charging

You can implement policy-driven charging in Oracle Communications 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
- Configuring ECE to Reject Spending Limit Requests Without Counters
- About the Policy Management API

> ⚠ **Caution**
>
> Deploying policy-driven charging for 5G events requires a cloud native deployment of ECE and BRM components. 5G PCF can be used only on an ECE cloud native system.

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 specific QoS to download video content. The subscriber chooses from one of many packages that you have configured with gradations in the QoS based on usage amounts in MBs, such as 100-150, 150-200, and 200-250 MBs. When the subscriber starts downloading video content from the network, you can track the number of MBs the subscriber downloads during the session. When the downloaded quantity crosses the upper threshold set for the selected QoS (for example, 150 MBs), you can use 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 shift in the QoS from the current to the next level.

ECE supports policy-driven charging. Policy-driven charging implements network, customer, and service policies that service providers can use to improve customer experience and efficiently use network resources. Service providers can use policies for various reasons, such as controlling data usage, setting QoS, allocating bandwidth to each service, enforcing parental controls, implementing charging rules, and so on.

When you integrate Policy and Charging Rules Function (PCRF) policy clients with ECE, ECE acts as the Subscriber Profile Repository (SPR) because it stores the customer profile

Copyright © 2019, 2026, Oracle and/or its affiliates.

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. You can create network-aware policies for specific access technologies where the network condition can dynamically alter prices. You can develop service-aware policies to control how a customer consumes network resources.

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) to support policy-driven 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 QoS label might be defined as **normal-QoS** or **low-QoS**, as shown below:

```
<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>
  </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 a subscriber downloads. 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. For example, they could retrieve:

  – A customer's charging-related information (for example, if the customer purchased a Gold, Platinum, or Bronze package)

  – A customer's preferred channel for receiving notifications (for example, email or SMS)

  – A 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 post notifications when thresholds are breached (SpendingLimit notifications). When a subscriber's preferences change, ECE publishes notifications with the new or altered preference

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 to receive 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.

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 MBs of data per month at a normal 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 MBs of data before the end of the month.
- The PCRF rule uses the label names **normal-QoS** and **low-QoS** as follows:

```
If (status_label=normal-QoS) (Bandwidth=10 Mbps)
If (status_label=low-QoS) (Bandwidth=128 kbps)
```

When the customer reaches the 300 MB data quota, 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 or

Copyright © 2019, 2026, Oracle and/or its affiliates.

HTTP Gateway, consume the data in these notifications for sending in-session notifications to the PCRF.

About ECE and Policy Clients

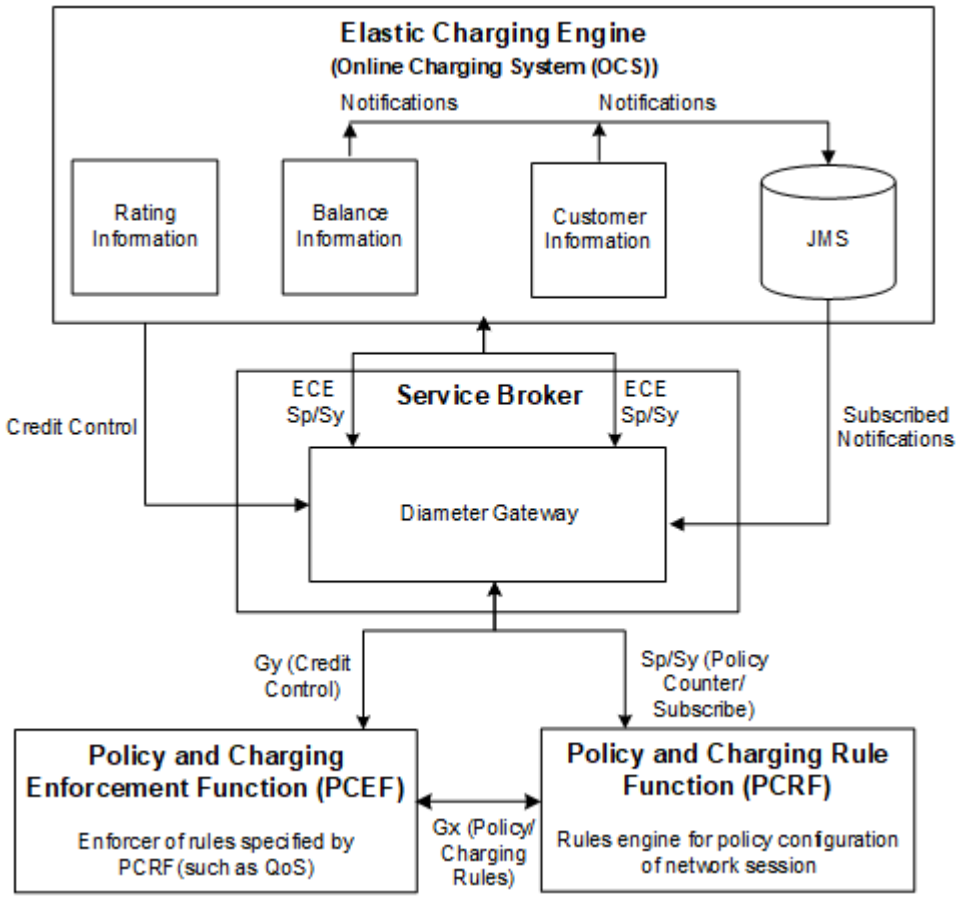
To support policy-driven charging, ECE offers a policy management API. Policy clients can use the API to retrieve data relevant to policy enforcement from its data grid.

Policy-driven charging in ECE is based on the PCRF, defined in the 3GPP TS 23.203 v9.9.0 specification. The PCRF integrates with ECE through your online network mediation software.

ECE exposes its in-memory cache 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 26-1 illustrates how ECE fits into a charging system that implements policy-driven charging.

Figure 26-1 ECE and Policy Client Integration



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).

Copyright © 2019, 2026, Oracle and/or its affiliates.

1. A customer starts to use a service, which initiates a network session.

   For example, the customer turns on a mobile phone that connects to a wireless network.

2. At the start of the network session, the PCEF obtains a policy configuration from the PCRF.

   The PCEF uses the Gx interface to get the policy configuration for the network session.

3. The PCRF requests policy counters and subscriber preferences from your online network mediation software (or 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 to receive notifications when the values of the policy counter information change.

   - Requests policy-related subscriber preferences by doing *one* of the following:

     – Retrieves the value for a specified set of subscriber preferences and subscribes to receive 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 to receive 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 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.

     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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

7. A charging session is established, and the PCEF sends a Ro message to your online network mediation software (or 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.
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 policy counter's new status. 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

    > **Note**
    >
    > If ECE detects multiple breaches during a session, it sends notifications sequentially. That is, it sends a notification for the first breach. Then, ECE waits for an acknowledgment from PCRF that it has received the notification before sending the subsequent breach notification.

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 policy-driven charging, policy-tier thresholds must be crossed to trigger the implementation of business rules, such as reduced QoS for subscribers who download excessive data.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 available. 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. The session ends when usage reduces the 80 MB balance to the point that the remaining balance cannot support the service.
- **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***/config/management/charging-settings.xml** and uncomment the following lines:

```
<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***/bin** directory.
4. Start Elastic Charging Controller (ECC):

   ```
   ./ecc
   ```

5. Start your charging servers:

   ```
   start server
   ```

6. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
7. Expand the **ECE Configuration** node.
8. Expand **charging.policyConfig**.
9. Expand **Operations**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

10. Select **setPolicyTolerance**.
11. 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 like Diameter Gateway integrate with ECE 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 to receive 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 to receive notifications containing 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 the PCRF uses to retrieve policy counter information. To support the Sy interface, ECE offers the following ECE Sy procedure 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- SpendingLimit Notification

  ECE uses this notification to report the status of requested policy counters for one or more services and 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 the PCRF uses 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 subscribe and unsubscribe (for the PCRF) to update 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 to receive 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 in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.policyConfig**.
4. Expand **Operations**.
5. Select **setDsl**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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:

      get*type***([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 to retrieve Sp and Sy information and subscribe or unsubscribe (for the PCRF) to receive updates to Sp and Sy data. This request combines 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 request processing results.

  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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.policyConfig**.
4. Expand **Attributes**.
5. 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 *Elastic Charging Engine Java API Reference*.

Configuring ECE to Treat Uninitialized Policy Counters as Zero

ECE uses policy counters for policy-driven charging. Some policy counters are dynamic and are created only after usage occurs. In these cases, a subscriber might not yet have an initialized balance bucket for a configured policy counter. This can affect policy-related notifications during change-plan operations (such as Spending Limit Notifications(SNRs)).

To control how ECE handles these uninitialized policy counters, set the **considerUninitialisedBalanceAsZeroForPolicy** ECE application configuration attribute to one of the following:

- **true**: ECE treats an uninitialized policy counter as having a value of **0**. This is the default.
- **false**: ECE skips the uninitialized policy counters that do not yet have a balance bucket.

To set the value of the **considerUninitialisedBalanceAsZeroForPolicy** attribute:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.policyConfig**.
4. Expand **Attributes**.
5. Set the value for the **considerUninitialisedBalanceAsZeroForPolicy** attribute to either **true** or **false**.
6. Save the changes in the JMX editor. Restart or reload the configuration.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part VI
Customizing ECE

This part provides information about customizing charging in Oracle Communications Elastic Charging Engine (ECE). It contains the following chapters:

- Customizing Rating
- ECE Sample Programs
- Testing ECE

Copyright © 2019, 2026, Oracle and/or its affiliates.

27

Customizing Rating

You can use the Oracle Communications Elastic Charging Engine (ECE) extensions to customize BRM Gateway, Diameter Gateway, HTTP 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.

> ⚠ **Caution**
>
> Deploying charging for 5G with HTTP Gateway (5G CHF) requires a cloud native deployment of ECE and BRM components. The HTTP Gateway can be used only on an ECE cloud native system.

Topics in this document:

- Operational Considerations
- Extension Points
- Implementing the Extensions Logic
- Sample Extensions

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.

Configuring Extensions

You configure implementation classes for the diameter-request processing, HTTP-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, HTTP-request processing, and usage-request processing extension points:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.extensions**.
4. Expand **Attributes**.
5. Specify values for the following attributes as needed:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **brmGwExtension**
- **diameterGyExtension**
- **diameterSyExtension**
- **httpExtension**
- **ocsBypassExtension**
- **postChargingExtension**
- **postRatingExtension**
- **postRatingMidSessionExtension**
- **postUpdateExtension**
- **preRatingExtension**
- **preRatingMidSessionExtension**
- **radiusAccountingExtension**
- **radiusAuthExtension**
- **ratingExtension**

About Performance with Extensions

If extensions are activated, they are called for during every usage request. Always consider performance for the code you run 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.

Using Value Map Results in Rating Extensions

Use the **getValueMapResult()** method in rating extensions to retrieve value map results. The **getValueMapResult()** method works only when the **SampleCustomDslExtension** JVM property is for the charging server nodes and Pricing Updater is configured to load selectors into the extensions cache.

Enabling custom DSL evaluation also improves performance for deployments that use large rate plan selectors or custom analyzer rules because ECE uses it to process selector rules during rating.

To enable the custom DSL JVM property results for rating extensions and configure Pricing Updater to load selectors into the extensions cache:

1. Configure each ECE charging server node that runs rating extensions to use the **customDslClassName** JVM system property:
    a. In *ECE_home*/**config/eceTopology.conf**, locate the rows where the role value is **server**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

b. For each server node that runs rating extensions, record the value that is in the **JVM Tuning File** column.

c. Open each corresponding tuning profile properties file in *ECE_home*/**config**. For example, if the **JVM Tuning File** value is **defaultTuningProfile**, edit **defaultTuningProfile.properties**.

d. Add the following JVM system property to the uncommented options line in the file. Do not add the property to commented sample lines.

```
-
DcustomDslClassName=oracle.communication.brm.charging.sdk.extensions.Sam
pleCustomDslExtension
```

e. Save and close the properties file.

> **Note**
>
> - Keep the JVM options on the same line. The **defaultTuningProfile.properties** file defines the options as a single JVM options string.
> - Do not remove existing flags. Append the new **-DcustomDslClassName** JVM system property to the existing options string.

2. Configure the **pricingUpdater** node to use the same JVM system property, if needed:

   a. In *ECE_home*/**config/eceTopology.conf**, locate the row where the role value is **pricingUpdater** and record the value that is in the **JVM Tuning File** column.

   b. If the **pricingUpdater** node uses a different tuning profile properties file from the charging server nodes, open that file in *ECE_home*/**config** and add the same JVM system property to the uncommented options line. If it uses a tuning profile properties file that you already updated, no additional change is required.

3. Start or restart the affected charging server nodes and Pricing Updater so that they use the updated JVM tuning profiles.

4. Republish the following PDC components to ECE:

   - Product Attribute Specifications
   - Custom Analyzer Rules
   - Rate Plan Selectors for charge selectors and discount selectors

5. Verify that selectors are loaded correctly into the extensions cache and test the extension code that calls **getValueMapResult()**.

For information about publishing PDC components, see "ImportExportPricing" in *PDC Creating Product Offerings*.

About Logging in Extensions

Logging is available in your extensions. You use the Log4j logger to write messages to the server node log file. For example:

```
extensionContext.getLogger().debug("Hello World!" + extensionContext);
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

About Extension Exceptions

If ECE needs to reject a 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 about **ExtensionsException,** see *Elastic Charging Engine Java API Reference*.

About Extension 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 processes
- Run latency-sensitive performance tests on the extensions hooks
- Use JAR signing

Extension Points

You customize BRM Gateway, Diameter Gateway, HTTP Gateway, RADIUS Gateway, pre-rating, rating, post-rating, post-charging, and post-update processes with these extension points:

- BRM Gateway Request Processing Extension Points
- Diameter-Request Processing Extension Points
- HTTP Gateway Request Processing Extension Points
- RADIUS-Request Processing Extension Points
- Update-Request Processing Extension Points
- Usage-Request Processing Extension Points

BRM Gateway Request Processing Extension Points

You use BRM Gateway request processing extension points to update an external notification that is bound for the BRM Gateway. It adds data for calling a specific BRM opcode. This allows you to update attributes in the ECE cache through the BRM-ECE synchronization process. See "About Synchronizing Data Between BRM and ECE".

ECE publishes external notifications to the ECE notification topics, where they are retrieved by BRM Gateway. BRM Gateway uses the information in each notification to call the specified opcode, which in turn updates customer data in the BRM database. BRM then resynchronizes the customer data with the ECE cache.

BRM Gateway provides these extension points for external notification flows that are targeted for the BRM Gateway:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **PostCharging extension.** The role of the PostCharging extension is to retrieve data for a specific opcode.
- **BRMCustomOpCodeCall extension.** The role of the BRMCustomOpCodeCall extension is to enrich external notifications with an input flist for a specific BRM opcode. This extension is called before the external notification is published to the ECE notification topics.

Diameter-Request Processing Extension Points

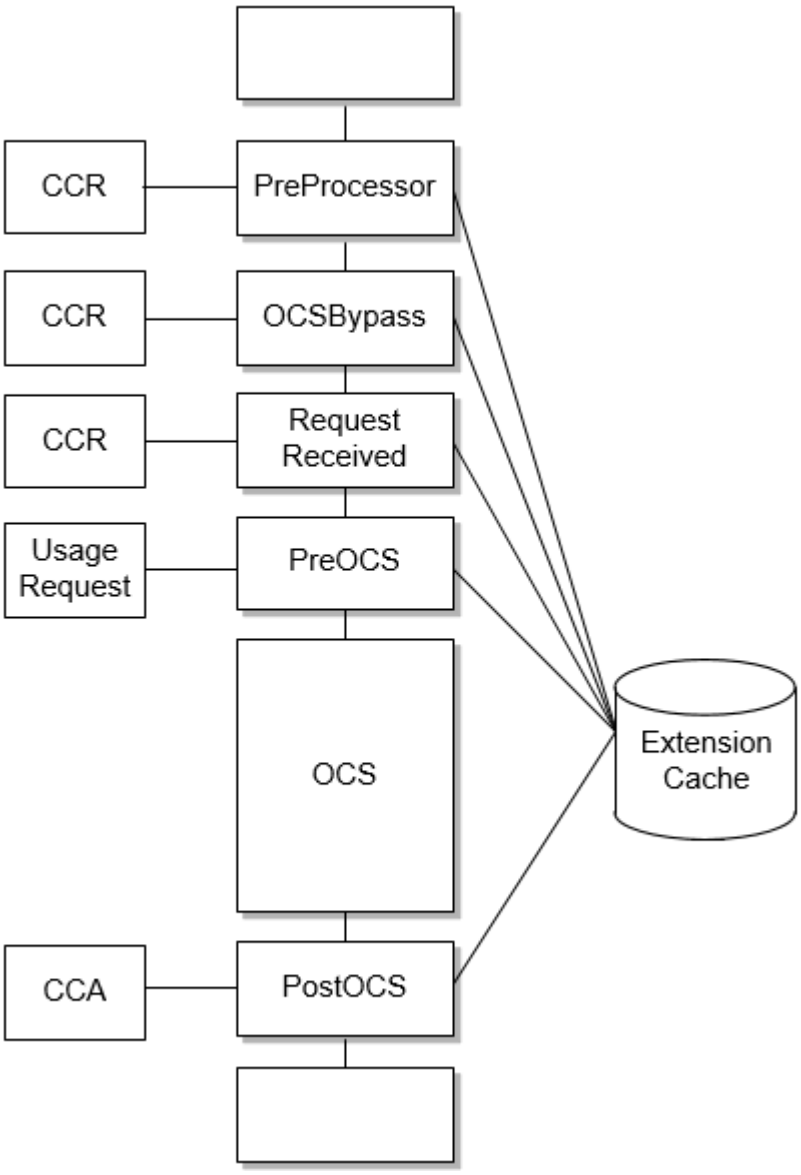
Diameter Gateway provides extension points for Credit Control Request (CCR) and Credit Control Answer (CCA) flows:

- **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.
- **PreProcessor extension.** The role of the PreProcessor extension is to add or remove any AVP in the custom response before the usage request is processed by Diameter Gateway and to provide an immediate response that bypasses the OCS completely.
- **PreOCS extension.** The role of the PreOCS extension is to manipulate the mapped ECE usage request payload and perform enrichment that is 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.
- **OCSBypass extension**. The role of the OCSBypass extension is to bypass the rating of Diameter CCRs received during a planned maintenance or an unplanned downtime of ECE and persist them to Oracle NoSQL.

Figure 27-1 shows the diameter-request processing extension points.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-1 Diameter-Request Processing Extension Points



HTTP Gateway Request Processing Extension Points

HTTP Gateway provides extension points for 5G flows:

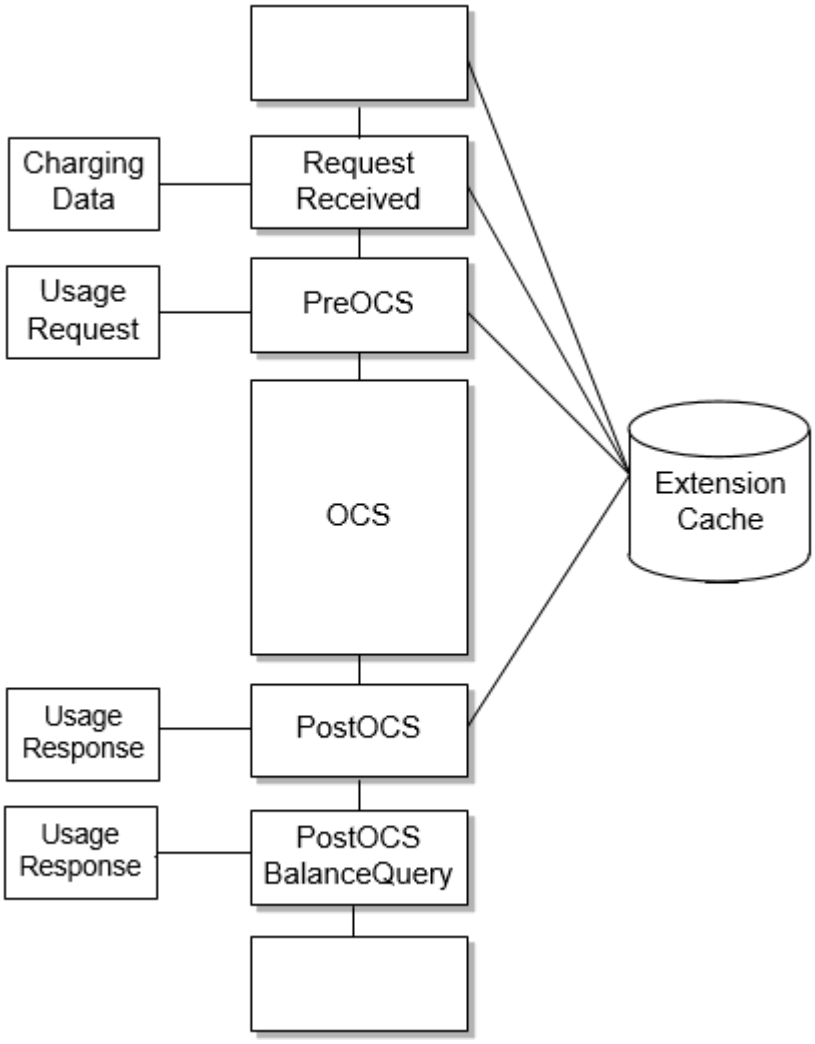
- **RequestReceived extension.** The role of the RequestReceived extension is to manipulate the charging data before the usage request is processed by HTTP Gateway and to provide an immediate response that bypasses the online charging system (OCS) completely.
- **PreOCS extension.** The role of the PreOCS 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 ECE usage request before the HTTP Gateway response is returned to the 5G client.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **PostOCSBalanceQuery** extension. The role of the PostOCSBalanceQuery extension is to manipulate the ECE usage response before the HTTP Gateway response is returned to the 5G client.

Figure 27-2 shows the HTTP Gateway request processing extension points.

Figure 27-2 HTTP Gateway 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:

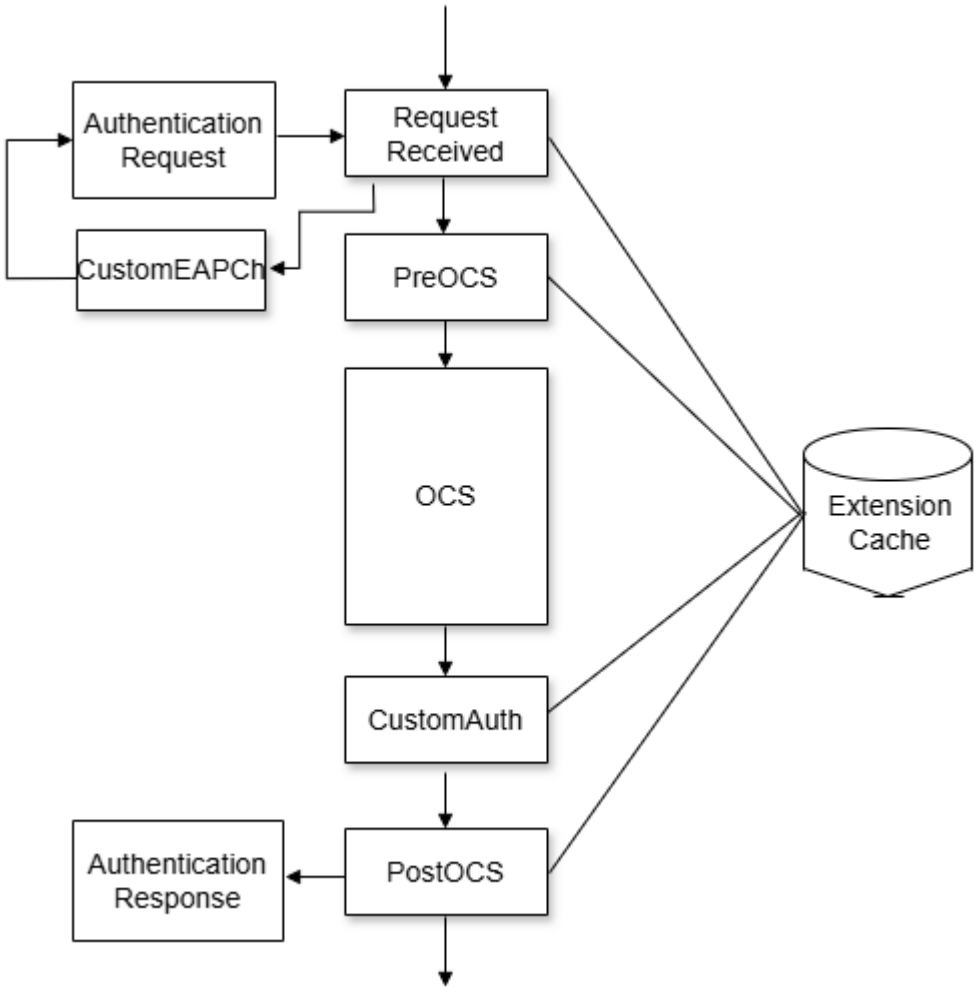
- **RequestReceived extension.** The role of the RequestReceived extension is to add or update a custom AVP before the authentication request is processed by RADIUS Gateway and to provide an immediate response that bypasses the OCS completely.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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.
- **RadiusAuthPreProcessorExtension extension.** The role of the RadiusAuthPreProcessorExtension extension is to provide access to subscriber profile information including customer and balance data in the extension hooks in the authentication flow of the RADIUS Gateway.
- **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 27-3 shows the RADIUS-request processing extension points for EAP authentication.

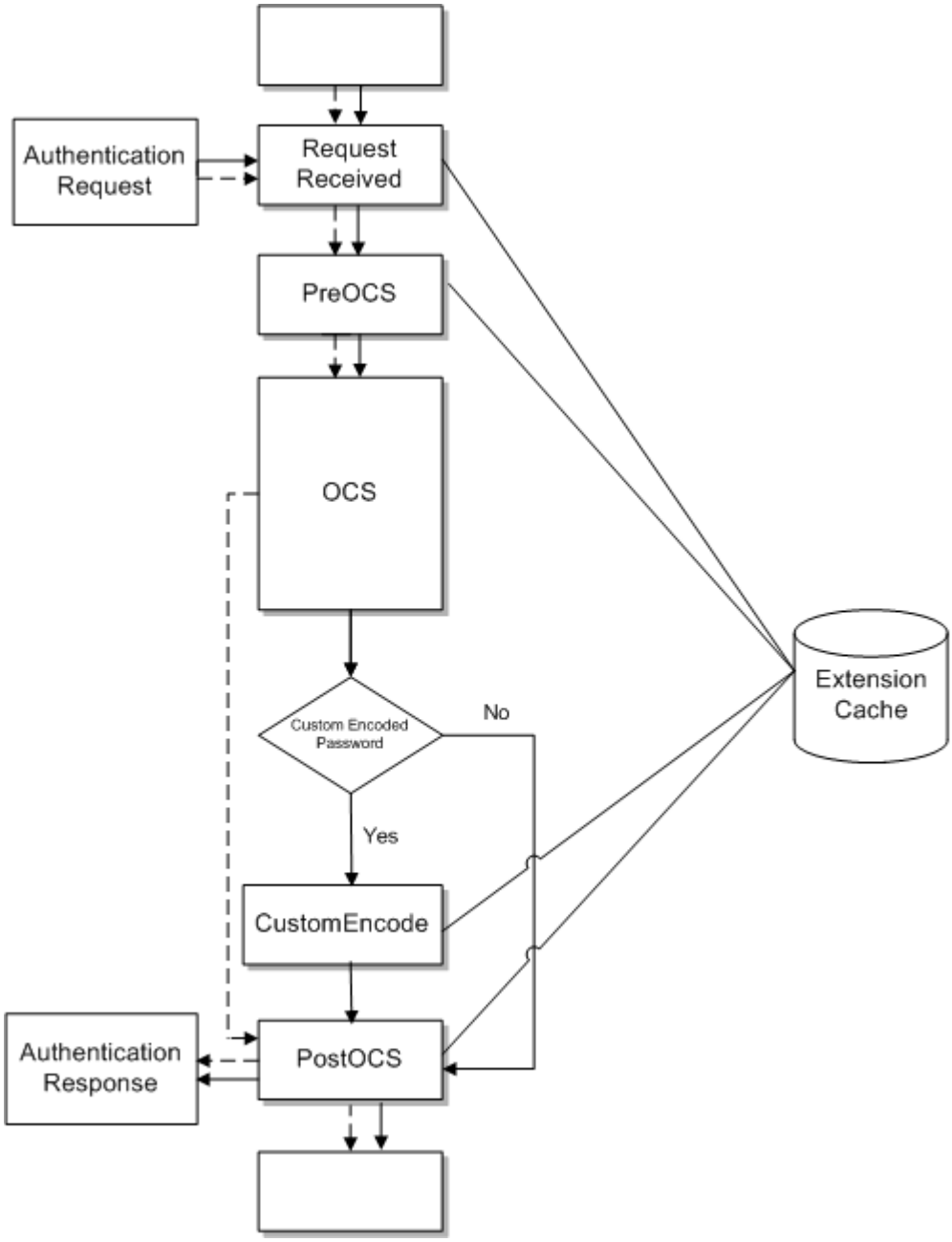
Figure 27-3 Extension Points for EAP Authentication



Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-4 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.

Figure 27-4 Extension Points for PAP and CHAP Authentication



Accounting Extension Points

RADIUS Gateway provides extension points for accounting flow:

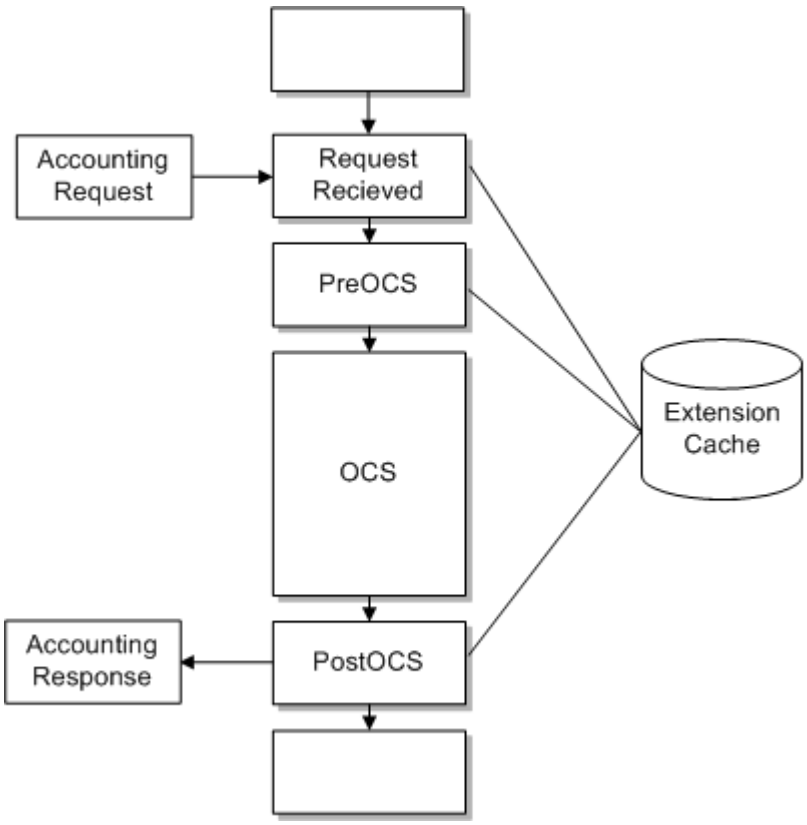
- **RequestReceived extension.** The role of the RequestReceived extension is to add or update a custom AVP before the accounting request is processed by RADIUS Gateway and to provide an immediate response that bypasses the OCS completely.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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 27-5 shows the RADIUS-request processing extension points for accounting.

Figure 27-5 Extension Points for Accounting



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.

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).

Table 27-1 describes the role of each extension point in the rating flow.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 27-1 Rating Flow Extension Roles

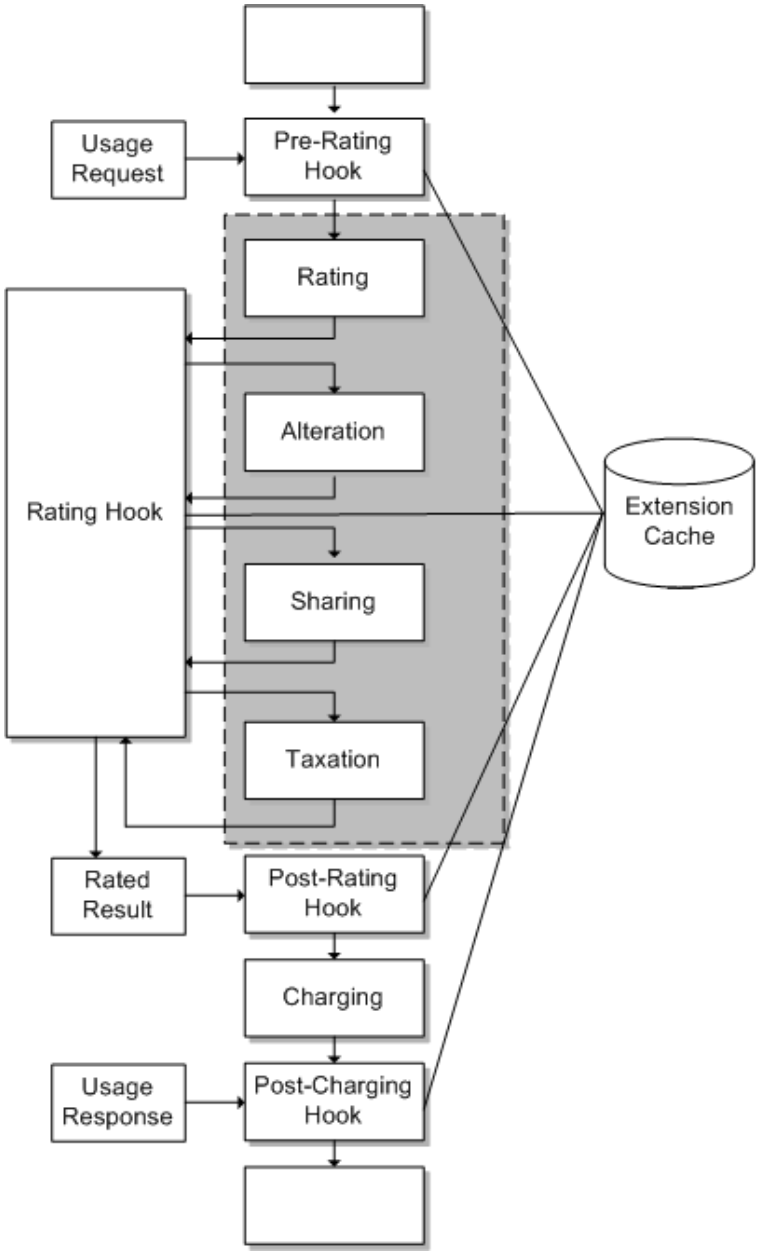
Extension Point	Role
Pre-Rating Extension	Alter the usage request
Post Rating Extension	Alter the rated result
Rating Extension	Alter rated results after each of the following processes: rating, alteration, sharing, and taxation
Post-Charging Extension	Enrich the usage response

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 27-6 shows the usage-request processing extension points.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-6 Usage-Request Processing Extension Points



Implementing the Extensions Logic

The **BRMCustomOpCodeCallExtension**, **DiameterGyExtension**, **DiameterSyExtension**, **HTTPExtension**, **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.

- **handleRequestReceived().** Called for every CCR that is processed by the charging flow.

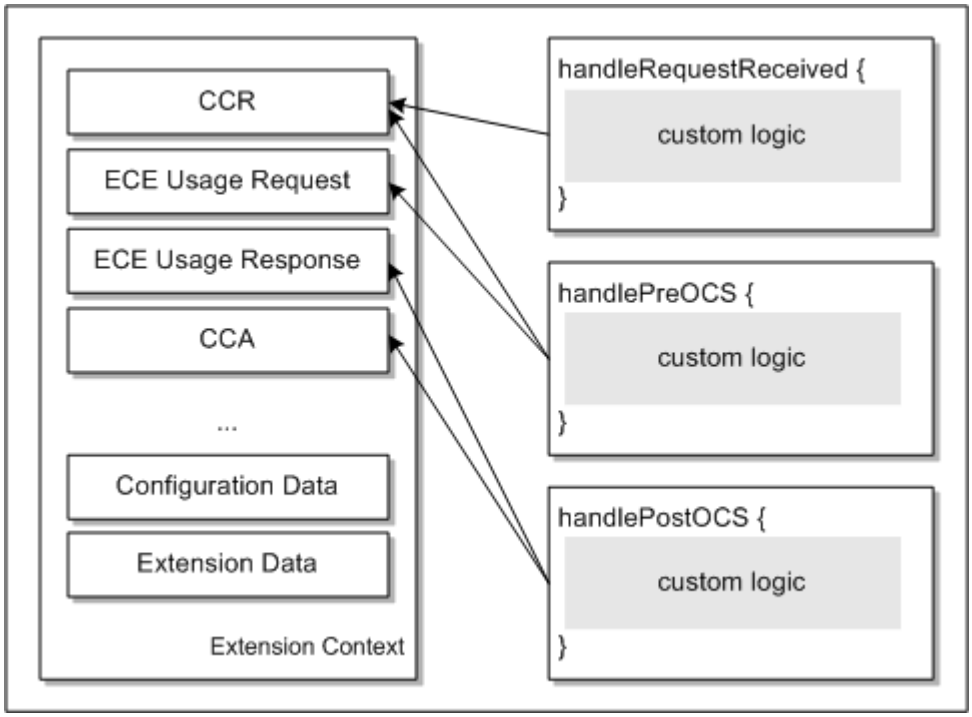
Copyright © 2019, 2026, Oracle and/or its affiliates.

- **handlePreOCS().** Called for every CCR and usage request that is processed by the charging flow.
- **handlePostOCS().** 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 27-7 shows the data used in the diameter-request processing extension points.

Figure 27-7 Data Used in Diameter-Request Processing Extension Points



For HTTP-request processing extension points, a different method is called for each extension point.

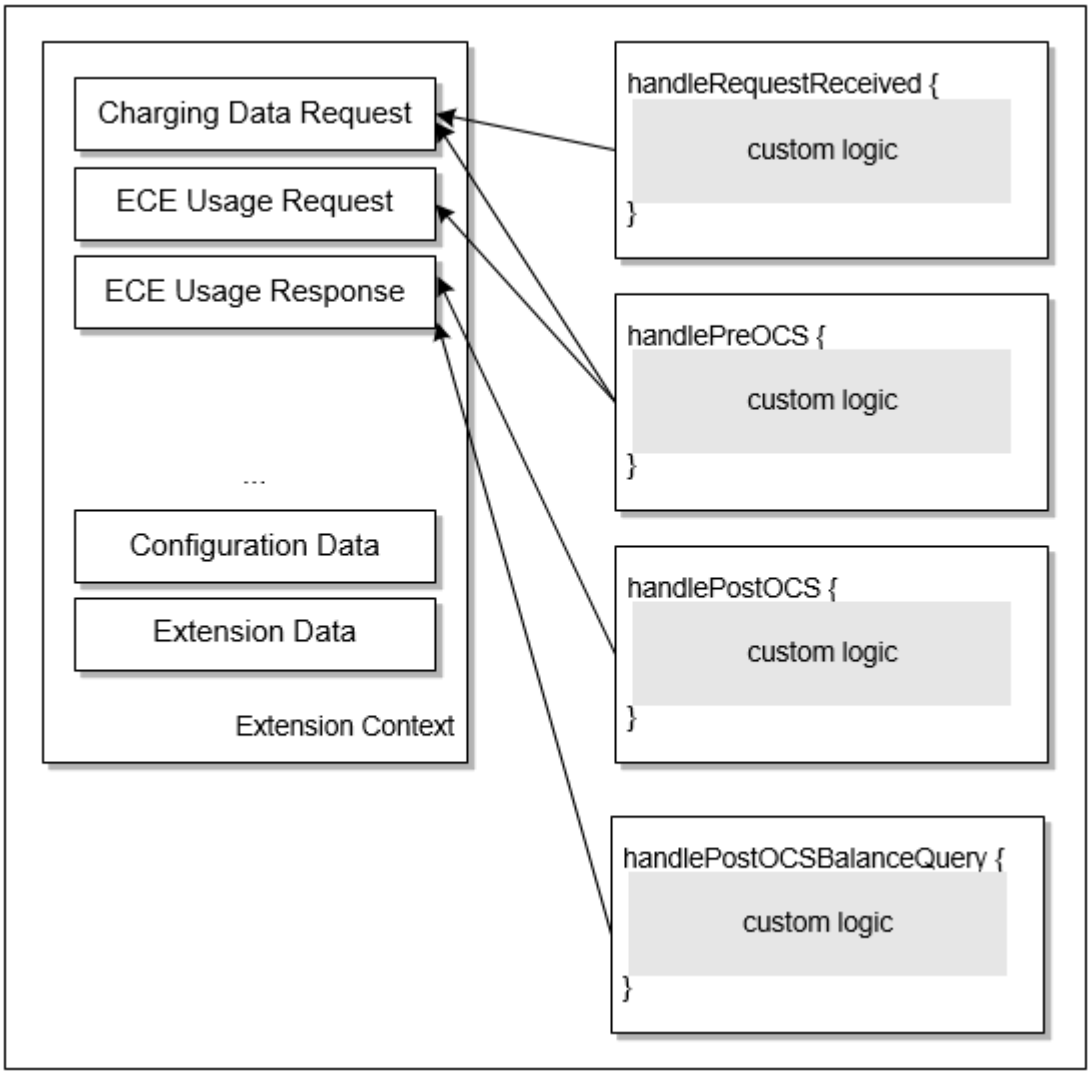
- **handleRequestReceived().** Called for every charging data request that is processed by the charging flow.
- **handlePreOCS().** Called for every usage request that is processed by the charging flow.
- **handlePostOCS().** Called for every usage response that is processed by the charging flow.
- **handlePostOCSBalanceQuery().** Called for every 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 27-8 shows the data used in the HTTP-request processing extension points.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-8 Data Used in HTTP-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:

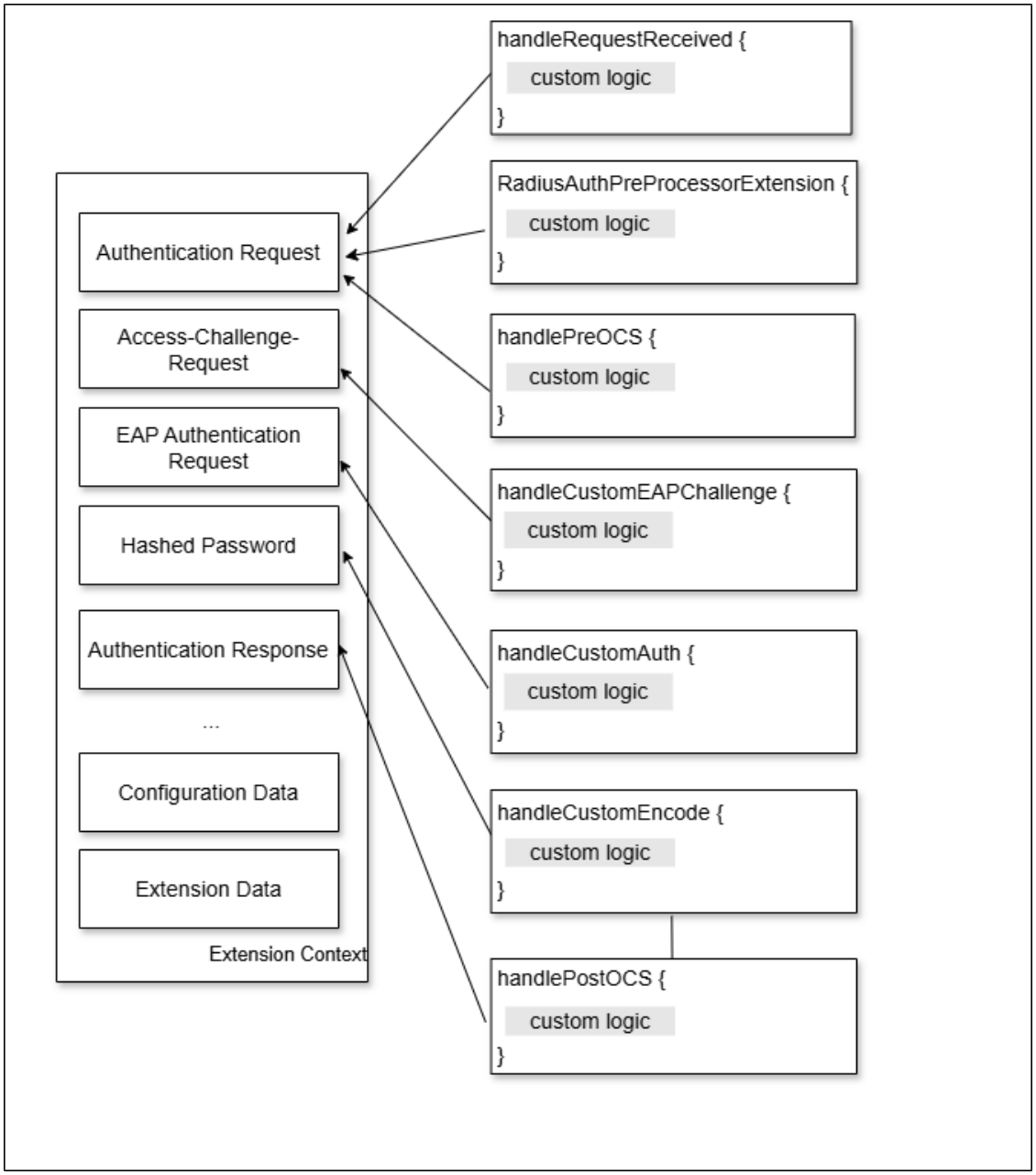
- **handleRequestReceived().** Called for every authentication request that is processed by the authentication flow.
- **handlePreOCS().** Called to perform any actions related to authentication that are required in the authentication flow.
- **handlePostOCS().** Called for each authentication response that is processed by the authentication flow.
- **handleCustomEAPChallenge().** Called to send custom access-challenge requests to the RADIUS client in the EAP authentication flow.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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 27-9 shows the data used in the RADIUS-request processing extension points for authentication.

Figure 27-9 Data Used in RADIUS-Request Processing Extension Points for Authentication



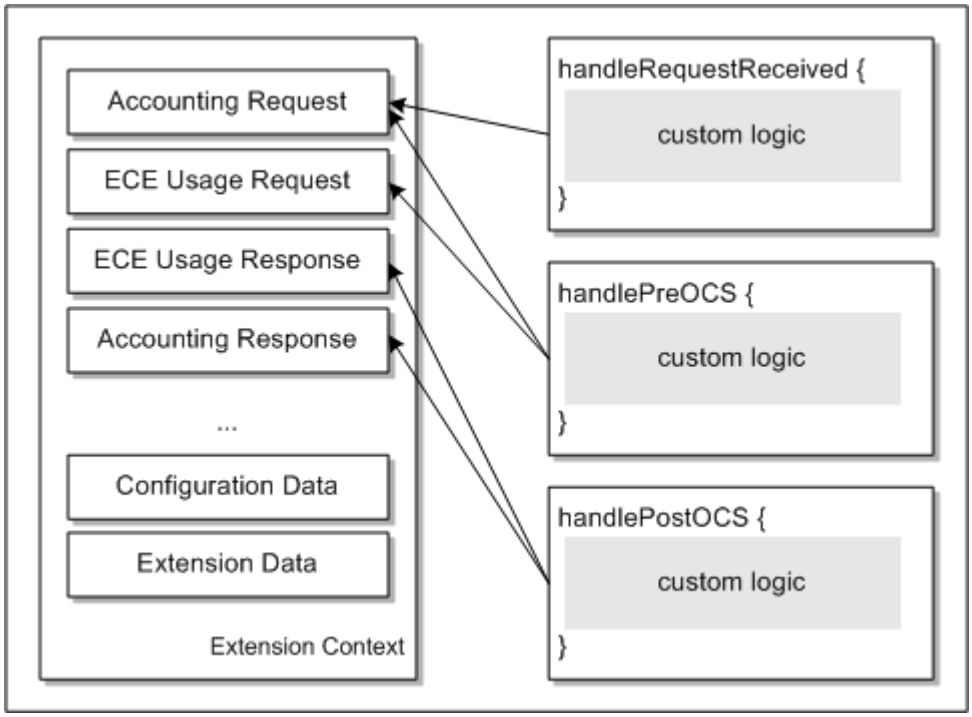
For accounting-related extension points, the following methods are called by the accounting flow:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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 27-10 shows the data used in the RADIUS-request processing extension points for accounting.

Figure 27-10 Data Used in 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 27-11 shows the data used in the pre-rating extension point.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-11 Data Used in Pre-Rating Extension Point

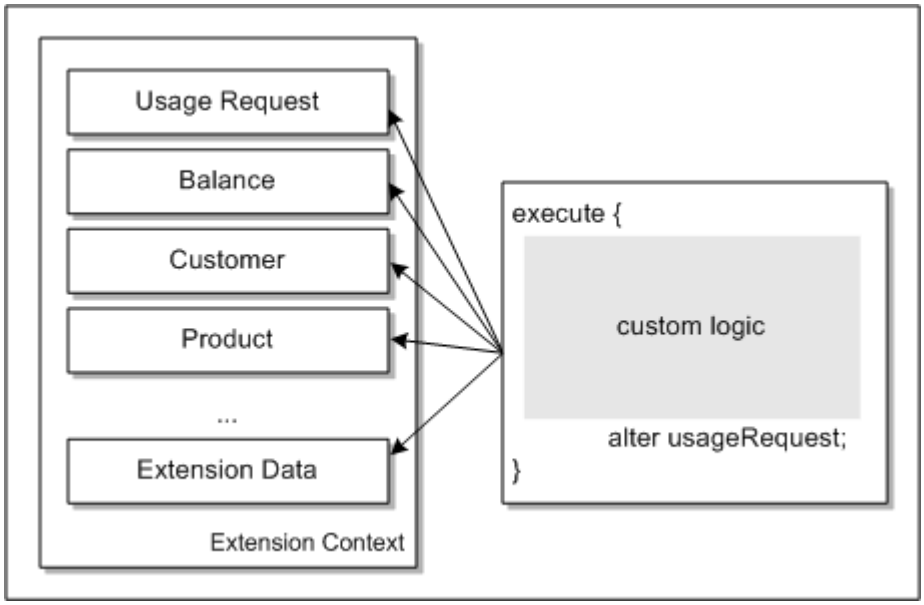


Figure 27-12 shows the data used in the rating extension point.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-12 Data Used in Rating Extension Point

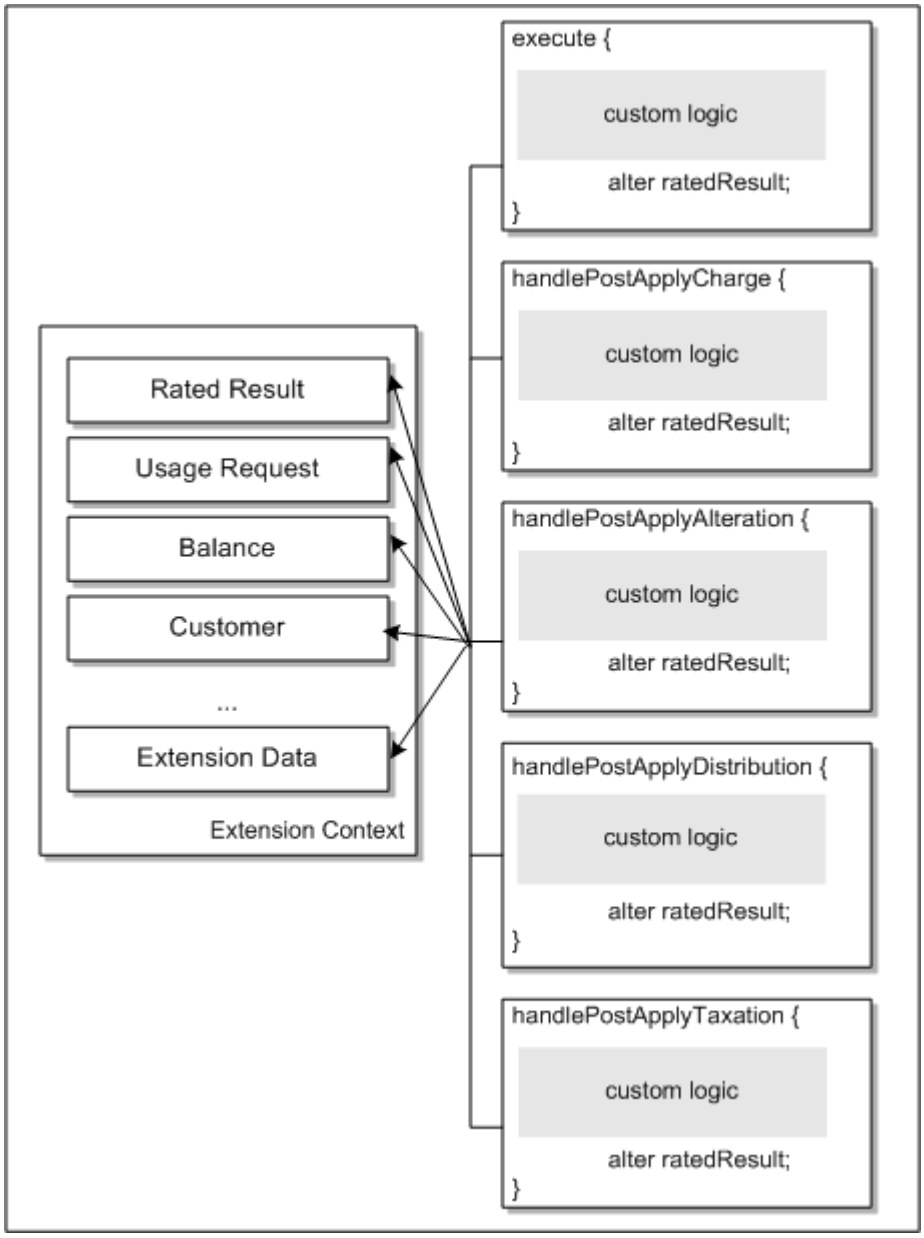


Figure 27-13 shows the data used in the post-rating extension point.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Figure 27-13 Data Used in Post-Rating Extension Point

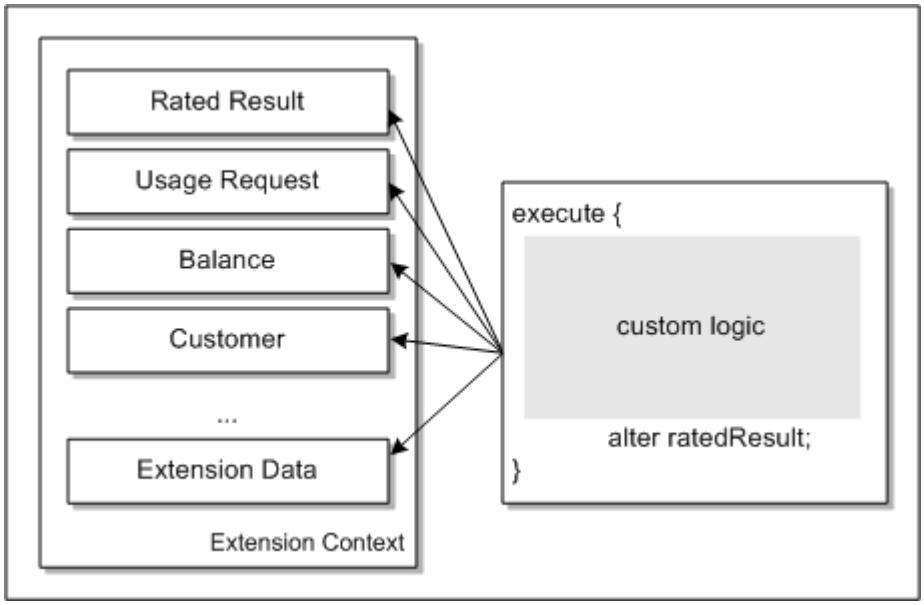
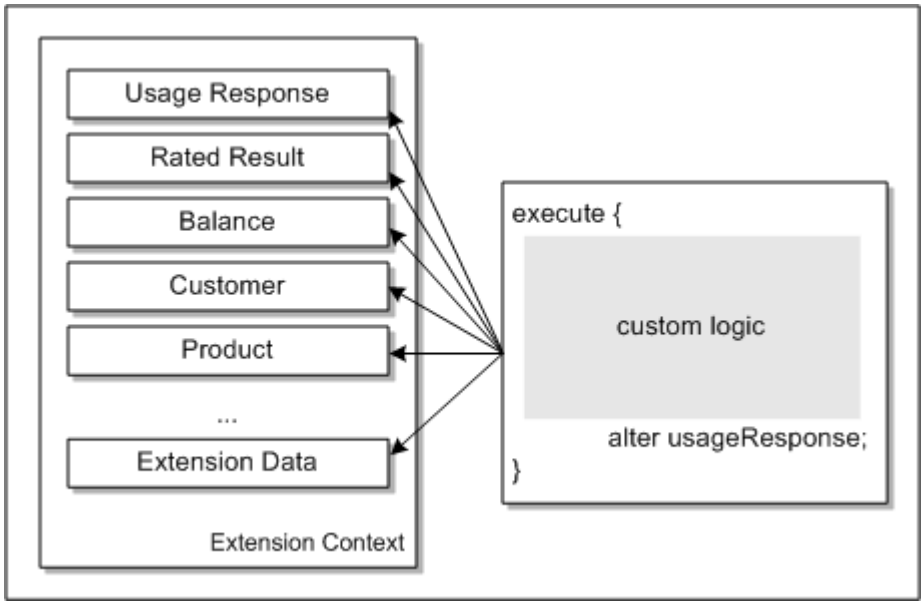


Figure 27-14 shows the data used in the post-charging extension point.

shows the data used in the post-charging extension point.

Figure 27-14 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).

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

Customizing a DSL Selector Extension for Impact-Based Selection

By default, a DSL selector extension returns only the first matching Charge Selector result. For impact-based candidate selection, ECE must receive every matching result so that it can evaluate the corresponding candidates, compare their impacts, and select the winner. Customize the extension to return all matching Charge Selector result IDs when ECE requests them.

To customize a DSL selector extension:

1. Implement the **CustomDslExtension.execute()** method that accepts four arguments:

   ```
   execute(Input input, String id, String type, boolean getAll)
   ```

2. When **getAll** is **false**, return zero or one matching result for **FIRST_MATCH**.
3. When **getAll** is **true**, return all matching result IDs for **MIN_IMPACT** and **MAX_IMPACT**.
4. Return each matching result ID only once and in configured evaluation order.
5. Keep the existing three-argument method for backward compatibility. Update it so that it calls the four-argument method with **getAll** set to **false**.

The four-argument method returns:

- **null** when selector evaluation fails and ECE should use the standard selector path.
- An empty list when no rule matches.
- A list of result IDs when one or more rules match.

For a Charge Selector, return the ordered result IDs and evaluated value-map entries as:

```
Pair ArrayList, String, Map String, String
```

The extension supplies the matching result IDs. ECE groups and evaluates the candidates, compares their impacts, resolves ties, and selects the winner.

BRMCustomOpCodeCall Extension

The BRMCustomOpCodeCall extension enriches external notifications with an input flist for a BRM opcode. The extension is called before publishing external notifications to the ECE notification topics, where they will be retrieved by the BRM Gateway.

You can modify the external notifications to include an input flist for a different BRM opcode.

CustomAuth Extension

The CustomAuth extension implements custom EAP authentication methods; for example, EAP-POTP, EAP-PSK etc.

The extension can access this data:

Copyright © 2019, 2026, Oracle and/or its affiliates.

- EAP-Authentication-Request
- System configuration

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.

The extension can access this data:

- Access-Challenge-Request
- System configuration
- Extensions data

You can use the extension point to send the custom access-challenge request to the RADIUS client when EAP is used for authentication.

CustomEncode Extension

The CustomEncode extension implements the custom hashing algorithm that is used on passwords for authentication.

The extension can access this data:

- Encoded Password
- System configuration
- Extensions 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.

OCSBypass Extension

The OCSBypass extension bypasses rating of CCRs received during ECE downtime or a planned maintenance activity. The CCRs are persisted to Oracle NoSql. When ECE is restored, the persisted requests are replayed to the ECE charging server for rating and updating balance impacts along with the real-time requests. The entire diameter request is accessible and modifiable and the entire request information is passed to the custom extension. When rating bypass is enabled and the custom extension is run, Diameter Gateway returns responses that you configured in the custom extension. If the extension is not enabled but bypassing of rating is enabled, ECE returns Diameter Result Code 5012 for the described conditions, which must be handled in your custom extension implementation.

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. In addition, the PreOCS extension performs any actions related to authentication that are required before the request is sent to ECE. This extension is called before any rating, discounting, or alteration logic has been invoked.

Copyright © 2019, 2026, Oracle and/or its affiliates.

The extension can access this data:

- Credit Control Request
- Authentication Request
- Accounting Request
- ECE Usage Request
- System configuration
- Extensions 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.

PreProcessor Extension

The PreProcessor extension manipulates CCR requests before the usage request is created, so that the request can match the business requirement. The PreProcessor extension can bypass charging or mutate the contents of the Sy message. This extension is called before any rating, discounting, or alteration logic has been invoked.

The extension can access this data:

- Credit Control Request
- Request AVPs
- ECE payload
- System configuration
- Extensions data

PostOCS Extension

The PostOCS extension manipulates CCA, accounting, usage, or authentication responses to match the business requirement before returning the response to the client. This extension is called after charging, authentication, and accounting has been completed and recorded.

The extension can access this data:

- Credit Control Request
- Accounting Response
- Authentication Response
- ECE Usage Response
- Diameter Credit Control Answer
- System configuration
- Extensions 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

PostOCSBalanceQuery Extension

The PostOCSBalanceQuery extension manipulates ECE usage responses to match the business requirement before returning the response to the HTTP client. This extension is called after a balance query has been completed and recorded.

Pre-Rating Extension

The pre-rating extension enhances the usage request based on customer, service, balance, product, and system 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.

The extension can access this data:

- ECE usage request information
- Customer data (including profile data)
- Service data (including profile data)
- Balance information
- Product information
- System configuration
- Extensions 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.
- Generate a midsession-rated event when an update operation occurs. This is implemented in the sample extensions provided.

You can also modify the values of the pricing attributes with custom logic. This enables you to override a product price.

Post-Rating Extension

The post-rating extension modifies the rated event based on customer, service, balance, product, and system data. This extension is called after any rating, discounting, or alteration logic has been invoked.

The extension can access this data:

- Balance information
- Balance element metadata from the ECE balance element cache, including the balance element name and description
- Customer data, including profile data
- ECE usage request information
- Product information
- Shared customers, if part of a sharing relationship
- Service data, including profile data

Copyright © 2019, 2026, Oracle and/or its affiliates.

- System configuration
- Extensions data
- Rated result

You can retrieve balance element metadata from the balance element repository. The **BalanceElement** object includes the balance element name and description. For example, when your post-rating extension has a balance element ID, you can retrieve the balance element and description as follows:

```
String balanceElementDescription =

extensionContext.getBalanceElementRepository().findBalanceElement(balanceElementId).getBa
lanceElementDescription();

String balanceElementName =

extensionContext.getBalanceElementRepository().findBalanceElement(balanceElementId).getBa
lanceElementName();
```

You can modify rated events in the following ways:

- You can modify the balance impact amount, GL code, tax code, balance element or invoice data for rating impacts generated from ECE.
- You can generate a midsession-rated event when an update operation occurs. This is implemented in the sample extensions provided.

You can also create new tax rating impacts, such as implementing a tax on tax.

Rating Extension

The rating extension modifies the rated results after each of the following processes: rating, alteration, sharing, and taxation.

The extension can access this data:

- Customer (including profile)
- Shared customer (if part of a sharing relationship)
- Product (including profile)
- Balance information
- System configuration
- Extensions
- Rated result

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

RequestReceived Extension

The RequestReceived extension manipulates the CCR, charging data request, authentication request, or accounting request so that the 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.

The extension can access this data:

- Credit Control Request
- Authentication Request
- Accounting Request
- System configuration
- Extensions 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.

Post-Charging Extension

The post-charging extension enriches usage responses and external notifications, excluding Advice of Charge (AOC) notifications. This extension is called after charging is completed but before the usage response is generated.

The extension can access this data:

- Customer data, including profile data and subscriber preferences
- Shared customers, if part of a sharing relationship and subscriber preferences
- Service data, including profile data, subscriber preferences, life cycle state
- Balance information, including current request impacts
- Balance element metadata from the ECE balance element cache, including the balance element name and description
- Business profile
- System configuration
- Extension data
- Rated result

You can retrieve balance element metadata from the balance element repository. The **BalanceElement** object includes the balance element name and description. You can use this information when enriching usage responses or external notifications. For example, when your post-charging extension has a balance element ID, you can retrieve the balance element name and description as follows:

```
String balanceElementDescription =

extensionContext.getBalanceElementRepository().findBalanceElement(balanceElementId).getBa
lanceElementDescription();

String balanceElementName =
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
extensionContext.getBalanceElementRepository().findBalanceElement(balanceElementId).getBa
lanceElementName();
```

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 the balance element name or description to custom data so that downstream systems can display user-readable balance information.
- Filter out external notifications that you do not want to be published to external systems.

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.

The extension can access this 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

You can modify the external notifications with custom logic. You can use the post-update extension 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 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 27-1 is an example of a specifications file for the post-rating extension.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Example 27-1 Sample Tax Table

```
/*
 * 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 27-2 shows the associated data to load into the cache using the specification file above.

Example 27-2 Example Data File

```
# 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
 usage   |U   |0.06    |01/01/2013 |12/31/2014 |Fed   |US     |USF         |Std
 usage   |U   |0.085   |01/01/2013 |12/31/2014 |Sta   |CA,AZ  |USTA        |Std
 purchase|V   |0.08525 |01/01/2013 |12/31/2014 |Sales |CA     |PSLS        |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.

The extension includes these 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- 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

The BRM SDK includes sample extensions. For information about how to use the samples, see "How To Use the Sample Extensions" and "Validating Sample Extensions".

For information about each sample extension, see the following:

- BRM Gateway Extension – Creating Opcode Flist
- Diameter Gateway Extension – Gy Service
- Diameter Gateway Extension – Sy Service
- HTTP Gateway Extension – Service
- OCSBypass Extension – Bypassing Rating
- Pre-Rating Extension – Dynamic Quota Management
- Pre-Rating Extension – Retrieving Function Values for Discount Expressions
- Pre-Rating Extension – Generating Midsession-Rated Event
- Pre-Rating Extension – Overriding Price in Product Offerings
- Post-Rating Extension – Complex Taxation
- Post-Rating Extension – Generating Midsession-Rated Events
- Post-Rating Extension – Adding or Deleting Rating Periods
- Post-Charging Extension – Adding Custom Data to Usage Responses and Notifications
- Post-Charging Extension – Overriding Dynamic Quota
- Post-Charging Extension – Adding or Modifying Redirection Rules
- Post-Charging Extension – Creating Custom Notifications for Top Ups
- Post-Charging Extension – Modifying the Renewal Message
- Post-Update Extension – Enriching External Notifications
- Rating/Charging Extension – Triggering RAR Notifications
- Rating Extension – Custom Item Assignment
- Extensions Data Load Sample

How To Use the Sample Extensions

To use the sample extensions:

1. ECE SDK is installed under **$SDK_HOME**. The directory listing is shown below:

```
$ ls -l
total 124
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
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 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.)

```
$ 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 XXXXX Jun 21 10:47 SampleOCSByPassExtension.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.brm.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**:

```
### 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

```
$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
$sh $SDK_HOME/bin/extensions/build_deploy_extension.sh build
SampleOCSByPassExtension
```

   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:

```
$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 using a rolling upgrade.

```
$sh $SDK_HOME/bin/extensions/build_deploy_extension.sh deploy
```

7. Define the fully-qualified class names of the pre-rating, bypass rating, rating, post-rating, post-charging, and post-update extensions by configuring the **charging.extensions** MBean in the **ECE Configuration** node using a JMX editor. For instructions, see "Configuring Extensions".

Copyright © 2019, 2026, Oracle and/or its affiliates.

```
preRatingExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePreRatingExtension"
RatingExtension="oracle.communication.brm.charging.sdk.extensions.
SampleRatingExtension"
postRatingExtension="oracle.communication.brm.charging.sdk.extensions.
SamplePostRatingComplexTaxationExtension"
postChargingExtension="oracle.communication.brm.charging.sdk.extensions.SamplePostCha
rgingExtension"
diameterGyExtension="oracle.communication.brm.charging.sdk.extensions.
SampleDiameterGyExtension"
postUpdateExtension="oracle.communication.brm.charging.sdk.extensions.SamplePostUpdat
eExtension"
ocsBypassExtension="oracle.communication.brm.charging.sdk.extensions.SampleOCSByPassE
xtension"
```

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 run 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 brought 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 these messages:

```
SamplePreRatingExtension invoked
PostRatingComplexTaxationSampleExtension executed
```

BRM Gateway Extension – Creating Opcode Flist

The sample program **SampleBRMGwCustomOpCodeExtension** shows how to use the ECE Extensions API to enrich an external notification with an input flist for a specified opcode.

The sample program builds an input flist for the PCM_OP_BILL_DEBIT opcode.

Diameter Gateway Extension – Gy Service

The sample program **SampleDiameterGyExtension** shows how to use the immediate-response feature based on an incoming AVP value.

The logic: If Service-Context-Id is OFFLINE, respond with Diameter Code DIAMETER_REDIRECT_INDICATION and set the Redirect-Host AVP value

Diameter Gateway Extension – Sy Service

The sample program **SampleDiameterSyExtension** shows how to use the ECE extensions API to suspend the Sy interface when a subscriber is suspended. The sample program does the following:

- Bypasses rating by calling the **setBypassOCS()** method and setting the DiameterResultCode to DIAMETER_REDIRECT_INDICATION.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> To skip the bypass, comment or remove the **setBypassOCS()** method invocation under the **handleRequestReceived()** method.

- Adds or removes AVPs in the Custom Sy Response.
- Modifies the request data time and product type in the policy request.
- Modifies the status and reason codes in the policy response.

HTTP Gateway Extension – Service

The sample program **SampleHTTPExtension** shows how to use the ECE Extensions API to override the requested number of units and perform call screening.

- The **handleRequestReceived()** method overrides the requested units for a call.
- The **handlePreOCS()** method sets the timestamp of when the call originally occurred.
- The **handlePostOCS()** method logs a message when call screening is done and returns a response to the 5G client.
- The **handlePostOCSBalanceQuery()** method logs a message when a balance query occurs and returns a response to the 5G client.

OCSBypass Extension – Bypassing Rating

The sample program **SampleOCSByPassExtension** shows how to use the ECE extensions API to bypass rating during ECE downtime. When bypass rating is enabled, the CCRs are persisted to Oracle NoSQL. When ECE is restored, the persisted requests are replayed to the ECE charging server for rating and updating balance impacts along with the real-time requests. Using this extension, you can write your own logic for modifying the AVP of incoming diameter messages and for creating the diameter responses.

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 the 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

Copyright © 2019, 2026, Oracle and/or its affiliates.

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 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

You can use the ECE extensions API to modify the quota configuration. The **SampleDynamicQuotaExtension** program is included to provide examples of how to use 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.

When a pre-rating extension updates a dynamic quota attribute, such as **QUOTA_ATTRIBUTES.VALIDITY_TIME**, ECE uses the value provided by the extension for the usage request instead of using the default reservation validity time. This behavior also applies when tariff time change (TTC) processing splits the usage request across tariff periods.

For example, to set the validity time from a pre-rating extension:

```
extensionContext.setQuotaAttribute(RatingConstants.QUOTA_ATTRIBUTES.VALIDITY_TIME,
valTime);
```

> **Note**
>
> The **getRequestStart()** method returns the session start time. To get the request start time for each request in a session, call **getOriginal().getRequestStart()** on the charging request:
>
> ```
> ((ChargingRequestImpl)
> extensionContext.getUsageRequest()).getOriginal().getRequestStart();
> ```

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.

If the PDC pricing specifies a 10% discount for all accounts active less than 12 months, the logic is the following:

Copyright © 2019, 2026, Oracle and/or its affiliates.

If **customerActiveMonths** value is:

- < 12 then apply a discount of 10%
- > 12 then apply a discount of 0%

Pre-Rating Extension – Generating Midsession-Rated Event

The **SamplePreRatingMidSessionExtension** program shows how to use the ECE extensions API to generate a midsession-rated event when an update request contains the following:

- A specified balance, such as greater than or equal to $500
- A changed field in a usage request
- A specified account field
- A specified product type, such as TelcoGsmTelephony

When an update request matches the criteria, the extension generates a midsession-rated event and adds the reason why it was created. It adds these reason codes to the event's **midSessionCDRSplitReason** field: CONFIGURED_DURATION_REACHED and RATING_CONDITION_CHANGE.

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.

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.

The extension uses the following logic:

1. Determines the federal tax rate from the tax configuration table using the tax code, request start time.
2. Calculates the federal tax based on this tax rate.
3. Modifies the original impact in the tax rating period based on the taxable impact from the linked charge, alteration, or distribution rating period.
4. Determines the state tax rate from the tax configuration table using the tax code, request start time.
5. Calculates state tax based on this tax rate.

Copyright © 2019, 2026, Oracle and/or its affiliates.

6. Creates 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.

Post-Rating Extension – Generating Midsession-Rated Events

The **SamplePostRatingMidSessionExtension.java** sample program shows how to use the ECE extensions API to generate a midsession-rated event when a rated event contains the following:

- A balance granted by an offer is exhausted
- A balance bucket is expired
- The charge offer used during rating is different from that of the ongoing session

When an event matches the criteria, the extension generates a midsession-rated event and adds the reason why it was created. It adds these reason codes to the event's **midSessionCDRSplitReason** field: CONFIGURED_VOLUME_REACHED and CONFIGURED_TIME_OF_THE_DAY_CROSSED.

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 containing the consolidated charge for all rating periods of type CHARGE.
- Delete the existing rating periods of type CHARGE.

You can use this sample program to override the rating periods in the final rating result by adding, modifying, or deleting rating periods.

When your extension creates aggregated rating periods, you must enable aggregated rating-period handling. Otherwise, the newly created rating periods might not be included in the rating result.

To include aggregated rating periods in the rating result:

1. Open the post-rating extension code.
2. In the extension's **execute()** method, call **setARPEnabled(true)** before calling the rating period aggregation logic. For example:

```
extensionContext.setARPEnabled(true);
aggregateRatingPeriods();
```

3. Compile and deploy the extension.

> **Note**
>
> If you do not call **extensionContext.setARPEnabled(true)** before aggregating the rating periods, the aggregated rating periods might be omitted from the rating result. As a result, downstream processing, such as REF extraction, might not produce the expected rated-event output.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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-Charging Extension - Enriching Notifications

The **SamplePostChargingExtension** program shows how to enrich the notification. You can use this sample program and update the payload with the following data:

- Session ID
- Notification event type
- Even timestamp
- Original balance
- Current balance

The sample program uses the **getCustomDataMap()** method to retrieve custom data and put it in the Notification's Payload. **CustomDataMap** is a map of String-String, where our custom name-value pair is included in the response.

Post-Charging Extension – Creating Custom Notifications for Top Ups

The sample program **SamplePostChargingExtension** shows how to use the ECE extensions API to create a custom notification for topping up a customer's balance element through the BRM PCM_OP_BILL_DEBIT opcode.

Copyright © 2019, 2026, Oracle and/or its affiliates.

The sample program retrieves data for a specified service type, request type, and balance element combination. It also specifies the BRM opcode to call. The sample program includes the following:

- The **getUsageRequest()** method, which fetches the input service usage request.
- The **getBalance()** method, which specifies that the logic applies to the USD balance element.
- The **getCustomDataMap()** method, which fetches the **customDataMap** for the custom data set by the user.
- The **addCustomServiceEventWithOpCode()** method, which specifies to call the PCM_OP_BILL_DEBIT opcode.
- The **getOriginalBalance()** method, which calculates and returns the original balance valid at the start time of the current usage session.
- The **getCurrentBalance()** method, which returns the current balance for the given resource considering the balance items valid during the specified validity range.
- The **getServiceEventType()** method, which retrieves the **ServiceEventType** associated with a service event.

Post-Charging Extension – Modifying the Renewal Message

The **SamplePostChargingExtension** program demonstrates how to use the ECE extensions API to modify the message included in renewable grant notifications (**renewalGrantActivationNotification**). These notifications are sent when an automatically renewable allowance is renewed or eaches its maximum number of renewals. To customize the message, add your content to the **CustomDataMap()** method under the "Message" key.

The sample program uses the **CustomDataMap()** method to insert your message into the notification's payload.

Post-Update Extension – Enriching External Notifications

The sample program **SamplePostChargingExtension** 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. For example:

```
s.getCustomDataMap().put("NotificationType",
      String.valueOf(s.getServiceEventType()));
```

In the above code snippet, we fetch the notification type from the Service Event and store its value in the **customDataMap**. This will subsequently be populated in the Notification Payload.

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

In the custom logic, if the **sendGenericRARNotification** is set to **true**, ECE generates generic RAR notifications for all Diameter sessions for the client. 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.

If **sendRarForSharedBalances** is set to **true**, ECE generates RAR notifications for all active sessions using a sharing group balance. For example, assume accounts A, B, C, and D and in a sharing group, and accounts A and C have active sessions using the sharing group balance. If an RAR is triggered for account C, ECE sends RAR notifications for accounts A's and C's active sessions. If **sendRarForSharedBalances** is **false**, ECE generates RAR notifications for all active sessions initiated by the subscriber. Using the previous example, ECE would generate RAR notifications for account C's active sessions only.

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.

The extension uses the following logic:

1. After taxation, determines the custom item type to be used based on the data accessible through the rating extension.
2. Assigns the rating impacts to the custom bill items based on the new custom item type.

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***/ocecesdk/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)

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

28

ECE Sample Programs

You use the sample programs included in the Oracle Communications Elastic Charging Engine (ECE) SDK to learn how to call the ECE APIs.

See *Elastic Charging Engine Java API Reference* for more information about the ECE SDK.

Topics in this document:

- About the ECE Sample Programs
- Finding the Sample Programs
- Descriptions of the Sample Programs
- Compiling and Running the Sample Programs
- Example of SampleDebitRefundSession
- Compiling and Deploying SampleRatedEventFormatterCustomPlugin

About the ECE Sample Programs

The ECE SDK includes sample programs that demonstrate how to use the ECE API for sending requests to ECE.

You can use these sample programs in the following ways:

- Use the sample programs as code samples for calling the ECE APIs.
- Use the sample programs as code samples for writing custom applications.
- Run sample programs to send requests to ECE and receive responses.

  The sample programs print information about the messages exchanged.
- Use the sample program scripts to get an idea of the configuration and dependencies that are required for integrating the ECE client into your build system (Maven, Ant, and so on).

You can also look at the sample program source code to see how it works. For example, if you want to write a program that sends a unit-based debit request to ECE, examine SampleDebitRefund to:

- View the methods to use in your code.
- How to use the libraries and calls.

Finding the Sample Programs

Table 28-1 shows the ECE SDK software directory structure, where *ECE_home* is the directory in which the ECE Server software is installed.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 28-1 Elastic Charging Engine Sample Program Directories

Directory	Description
ECE_home/ococesdk/bin	Directories that contain shell scripts for compiling and running various types of sample programs.
ECE_home/ococesdk/bin/extensions	Shell scripts for extension-implementation sample programs.
ECE_home/ococesdk/bin/notification	Shell scripts for notification sample programs.
ECE_home/ococesdk/bin/plugin	Shell scripts for the custom plug-in sample programs.
ECE_home/ococesdk/bin/policy	Shell scripts for policy sample programs.
ECE_home/ococesdk/bin/query	Shell scripts for query sample programs.
ECE_home/ococesdk/bin/update	Shell scripts for update sample programs.
ECE_home/ococesdk/bin/usage	Shell scripts for usage sample programs.
ECE_home/ococesdk/config	Configuration files common to all sample programs.
ECE_home/ococesdk/config/extensions	Configuration files for extension-implementation sample programs.
ECE_home/ococesdk/source	All Java sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/extensions	Source files for extension-implementation sample programs (for pre-request-processing and post-request-processing). Includes the data loader used for extensions.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/notification	Source files for notification sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/plugin	Source files for custom plug-in sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/policy	Source files for policy sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/query	Source files for query sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/update	Source files for update sample programs.
ECE_home/ococesdk/source/oracle/communication/brm/charging/sdk/usage	Source files for usage sample programs.

Descriptions of the Sample Programs

All of the sample programs can work with the ready-to-use sample data included with the ECE Server software installation. The sample programs are supported on the Linux platform.

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**.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **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:

- Usage Request Sample Programs
- Update Request Sample Programs
- Policy Request Sample Programs
- Query Request Sample Programs
- Extension Implementation Sample Programs
- Notification Sample Programs
- Custom Plug-In Sample Programs

To determine which parameter values you must use for running a sample program, you can use the sample script's help option. For descriptions of the methods the sample programs use, see **oracle.communication.brm.charging.sdk** in *Elastic Charging Engine Java API Reference*.

Usage Request Sample Programs

Table 28-2 lists the usage sample programs, their descriptions, the shell scripts used to compile and run them, and the applicable **.ecc** script.

Table 28-2 ECE Sample Programs for Usage Requests

Sample Program	ECC Script	Shell Script	Description
SampleAccountingOnOff	**sdk_production_loader.ecc**	**sample_accounting_on_off.sh**	Simulates an accounting on/off request being sent from the mediation client.
SampleDataSession	**sdk_production_loader.ecc**	**sample_data_session.sh**	Simulates a simple data session, including an INITIATE, an UPDATE and a TERMINATE request.
SampleDebitRefundSession	-	**sample_debit_refund_session.sh**	Shows how to send debit and refund requests with multiple values in unit-based and amount-based mode. See "Example of SampleDebitRefundSession".
SampleGenericSession	-	**sample_generic_session.sh**	Simulates any kind of voice or data session.
SampleGprsSession	-	**sample_gprs_session.sh**	Simulates a GPRS session.
SampleIncrementalUsageRequestLauncher	**sdk_production_loader.ecc**	**sample_incremental_usage_request.sh**	Simulates a voice session with incremental mode.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 28-2 (Cont.) ECE Sample Programs for Usage Requests

Sample Program	ECC Script	Shell Script	Description
SampleMultipleServicestLauncher	**sdk_production_loader.ecc**	**sample_multiple_service.sh**	Shows how to send usage requests for the Multiple Services Credit Control (MSCC) case (multiple subrequests are sent in a single usage request).
SamplePriceEnquiry	**sdk_production_loader.ecc**	**sample_price_enquiry.sh**	Sends a price enquiry request.
SampleReAuthRequest	**sdk_production_loader.ecc**	**sample_RAR.sh**	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.
SampleStartUpdateAccountingRequestLauncher	-	**sample_start_update_accounting_request.sh**	Simulates a sample usage session including a START_ACCOUNTING, UPDATE_ACCOUNTING, and TERMINATE request. For example, a usage session for a DSL data download in a postpaid scenario.
SampleUsageRequestLauncher	**sdk_production_loader.ecc**	**sample_usage_request.sh**	Enables you to send custom voice usage requests. Customer ID, number of requests to send, request type (INITIATE/UPDATE...) and duration must be given as arguments (e.g. sample_usage_request.sh run 6500000000 2 TERMINATE 120).
SampleVoiceSession	**sdk_production_loader.ecc**	**sample_voice_session.sh**	Simulates a simple voice session, including an INITIATE, an UPDATE and a TERMINATE request.

Update Request Sample Programs

Table 28-3 lists the update sample program.

Table 28-3 ECE Sample Programs for Update Requests

Sample Program	ECC Script	Shell Script	Description
SampleExternalTopUpRequestLauncher	-	**sample_external_topup_notification_request.sh**	Shows how third party systems can perform direct top ups in ECE.

Policy Request Sample Programs

Table 28-4 lists the policy sample programs.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 28-4 ECE Sample Programs for Policy Requests

Sample Program	ECC Script	Shell Script	Description
SamplePolicySessionRequestLauncher	-	**sample_policy_session_request.sh**	Simulates a policy session. Shows how to send a policy request to ECE that requests both Sp and Sy information.
SampleSpendingLimitReportRequestLauncher	-	**sample_spending_limit_report_request.sh**	Simulates a policy Sy query request. Shows how to send a request to retrieve policy counter status information.
SampleSubscribeNotificationRequestLauncher	-	**sample_subscribe_notification_request.sh**	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).
SampleSubscriberPreferenceUpdateRequestLauncher	-	**sample_subscriber_preference_update_request.sh**	Simulates a policy-related update request. Shows how to update the subscriber preferences in ECE.
SampleUserDataRequestLauncher	-	**sample_user_data_request.sh**	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.

Query Request Sample Programs

Table 28-5 lists the query sample programs.

Table 28-5 ECE Sample Programs for Query Requests

Sample Program	ECC Script	Shell Script	Description
SampleAuthenticationQuery	**sdk_production_loader.ecc**	**sample_auth_query_request.sh**	Sends an authentication query request.
SampleBalanceQueryRequestLauncher	**sdk_production_loader.ecc**	**sample_balance_query_request.sh**	Sends a balance query request.

Extension Implementation Sample Programs

Table 28-6 lists the extension sample programs.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 28-6 ECE Sample Programs for Extension Implementations

Sample Program	ECC Script	Shell Script	Description
-	-	**sample_extensions_loader**	Data loader for extension implementations.
-	-	**build_deploy_extension**	Sample extension implementation.
-	-	**tax_configuration.spec**	Sample extension implementation.
-	-	**tax_configuration_data.csv**	Sample extension implementation.

Notification Sample Programs

Table 28-7 lists the notifications sample programs.

Table 28-7 ECE Sample Programs for Notifications

Sample Program	ECC Script	Shell Script	Description
SampleDurableJmsClient	-	**sample_durable_jms_client.sh**	Simulates a durable JMS client.
SampleJmsClient	-	**sample_jms_client.sh**	Simulates a JMS client.
SampleJmsServer	-	**sample_jms_server.sh**	Simulates a JMS server.

Custom Plug-In Sample Programs

Table 28-8 lists the custom plug-in sample programs.

Table 28-8 ECE Sample Programs for Custom Plug-In

Sample Program	ECC Script	Shell Script	Description
SampleRatedEventFormatterCustomPlugin	-	**build_deploy_plugin.sh**	Writes rated events into CDR records.
SampleRatedEventFormatterKafkaCustomPlugin	-	**build_deploy_plugin.sh**	Writes fully rated events into a JSON file that is published to Kafka topics.

Compiling and Running the Sample Programs

You compile and run a sample program with the shell script provided for that sample program.

To compile and run a sample program:

1. Open the *ECE_home*/**config/eceTopology.conf** file.
2. Uncomment the line where the **sdkCustomerLoader** node is defined.

   You are required to uncomment this line to be able to run the SDK sample programs.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **⚠ Caution**
>
> Do *not* run the **customerLoader** utility without the **-incremental** parameter in a production environment.

3. Go to the ECE server **bin** directory:

   ```
   cd ECE_home/bin
   ```

4. Load the ECE runtime environment:

   ```
   ./ecc 'load sdk_production_loader.ecc'
   ```

5. Go 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 **extensions**, **notification**, **plugin**, **policy**, **query**, **update**, or **usage**. See Table 28-1 for more information.

   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***/ocecesdk/bin/usage** directory.

6. Compile the sample program:

   ```
   sh ./scriptname.sh build
   ```

   You must compile the sample program once.

   For example, to compile the **sample_voice_session.sh** sample program, enter:

   ```
   sh ./sample_voice_session.sh build
   ```

7. Run the sample program.

   ```
   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 command with no parameters to use default parameter values from the SDK scripts.

8. When you are done with the sample program, shut down the ECE runtime environment:

   ```
   ecc stop server
   ```

Example of SampleDebitRefundSession

The following shows an example of how to run the SampleDebitRefundSession sample program.

```
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 ...]
```

where:

- **build** compiles the related SDK source files.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **run** runs the SDK program (debit refund) according to the parameters you provide. You can supply the parameters in the command line or provide no parameters to run the 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.
- The 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.
- 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.
- DEBIT_UNIT or REFUND_UNIT, TOTAL, IN, and OUT must be specified with numbers in MB (megabytes).

For example, the following command debits 10 USD and 25 Included Minutes:

```
sample_debit_refund_session.sh run 650999777 DEBIT_AMOUNT CORR_ID 840,10 95,25
```

This example command refunds 50 USD and 5 Included Minutes:

```
sample_debit_refund_session.sh run 650999777 REFUND_AMOUNT CORR_ID 840,50 95,5
```

Compiling and Deploying SampleRatedEventFormatterCustomPlugin

To compile and deploy the **SampleRatedEventFormatterCustomPlugin** sample program:

1. Configure Rated Event Formatter by doing the following:
   a. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
   b. Expand the **ECE Configuration** node.
   c. Expand **charging.ratedEventFormatters.***Instance_Name*, where *Instance_Name* is the name of the instance you want to configure such as ratedEventFormatter1.
   d. Expand **Attributes**.
   e. Set the **pluginType** attribute to **oracle.communication.brm.charging.sdk.plugin.SampleRatedEventFormatterKafkaCustomPlugin**.
2. Go to the *ECE_home***/ocecesdk/bin/plugin** directory and then build the sample program:

   ```
   sh ./build_deploy_plugin.sh build SampleRatedEventFormatterKafkaCustomPlugin
   ```
3. Deploy the sample program:

   ```
   sh ./build_deploy_plugin.sh deploy SampleRatedEventFormatterKafkaCustomPlugin
   ```
4. Ensure that your Kafka Server is up and running.
5. Create a topic in your Kafka Server for the rated events.

   For example, this command creates a Kafka topic named **RatedEvent**:

   ```
   bin/kafka-topics.sh --create --bootstrap-server kafkaHost:port --replication-factor
   1 --partitions 1 --topic RatedEvents
   ```


Copyright © 2019, 2026, Oracle and/or its affiliates.

where *kafkaHost* and *port* are the host and port that the Kafka client will connect to in a bootstrap Kafka cluster the first time it starts.

6. Wait until the entire configuration is ready and all components are up and running.
7. Perform usage, such as initiate, update, or terminate, for one of your customers.

   Rated events will be published to the Kafka topic in JSON format. Also, JSON information will be added to the **ratedEventFormatter1.log** file under the *ECE_home***/log** directory.
8. Confirm that the event was published to the Kafka topic by running the Kafka console consumer:

   ```
   bin/kafka-console-consumer.sh --bootstrap-server kafkaHost:port --topic RatedEvents
   ```
9. When you are done with the sample program, shut down the ECE runtime environment:

   ```
   ecc stop server
   ```


Copyright © 2019, 2026, Oracle and/or its affiliates.

29

Testing ECE

You learn how to test your implementation of Oracle Communications Elastic Charging Engine (ECE).

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

About ECE Testing Utilities

ECE offers the testing utilities in Table 29-1 that you can use when implementing ECE in a charging system.

Table 29-1 ECE Testing Utilities

Utility	Description
Simulator	Emulates the role of a 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. See "Running the Simulator to Send Usage Requests".
query	Enables you to run queries on ECE data for development or debugging purposes. See "Using the query Utility to Test ECE". **Note:** 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Table 29-1 (Cont.) ECE Testing Utilities

Utility	Description
customerGenerator	Creates XML files that represent sample customer data, which can then be loaded into ECE. **Note:** The ECE customer XML data file must conform to the format of the ECE customer XML schema file (*ECE_home***/odi_transformation/ECE_Schema.xsd**).
customerLoader	Loads customer data from BRM incrementally, in batches or in bulk. 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.
PerformanceMonitor MBean	Monitors 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***/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.

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

- **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.

> **Note**
>
> 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 configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Specify the values for the following attributes:
   - **virtualTimeMode.** Enter one of the following values:
     - **0.** Use this to reset the time to ECE's current time. Time is changed to the operating system time.
     - **1.** Use this to set the time as a constant time. Time is frozen at the specified time until this value or the time you set is changed.

Copyright © 2019, 2026, Oracle and/or its affiliates.

  - **2.** Use this to set the time to change every second. Time is changed to the time specified, and then advances one second every second.

- **virtualTime.** Enter the time and date in the following format: *YYYY*-*MM*-*DD*T*HH*:*mm*:*ss*.*SSS*. For example, to set the time and date to 11:30:02:600 on September 03, 2016, enter 2016-09-03T11:30:02.600.

After you change the time and date, perform testing as needed. You can also change the time and date between testing stages. After completing testing, reset the time to ECE's current time and perform database cleanup if needed.

Using the query Utility to Test ECE

The **query** utility provides access to ECE cache content, enabling you to run 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.

> **Note**
>
> Oracle recommends that you not query the cache for the entire list of customer information. Filtered/ selected query is recommended because the complete customer list would cause ECE to crash.

The **query** utility is included with the ECE Server software in *ECE_home***/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 *ECE_home***/logs/query_out.log**.

The query statement history is contained in *ECE_home***/bin/.cohql-history**. You can use the up and down arrows to move through the command history.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Note**
>
> 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='292278994-08-17T07:12:55.807Z}]}
, 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 = {}
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
,         subscribedSpendingLimitCounters = {}
,         Life cycle state = 102
,         Life cycle Expiration time = 0
,         activeSessions = {}
,         debitRefundSessions = {}
,         audited purchased charge offerings = {}
,         audited purchased alteration offerings = {}
,         audited profiles={}
,         audited UsedAlterationSharingAgreements={}
,         audited UsedDistributionSharingAgreements={}
        balance=null
}}
, balances={}
,
 billingUnits={BillingUnit#6500000001=[BillingUnitImpl
{billingUnitId=BillingUnit#6500000001},
{accountingCycle=[Triple{first=2013-08-21T11:10:16.224-07:00},
{second=2013-09-21T11:10:16.224-07:00},{third=2013-10-21T11:10:16.224-07:00},]},
{billingCycle=[Triple{first=2013-08-21T11:10:16.224-07:00},
{second=2013-09-21T11:10:16.224-07:00},{third=2013-10-21T11:10:16.224-07:00},]},
{billingFrequency=1},]}
, auditedProducts={}
, auditedProfiles={}
, audited AlterationSharingAgreements={}
, audited DistributionSharingAgreements={}
, customerRerateProcessingInfo=CustomerRerateProcessingInfoImpl
{ RerateProcessingStatus='NOT_IN_RERATING, ReratingJobId='null}
 ####################Customer End ##########################################
```

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}]),]
```

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **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. Go to the ECE_home**/bin** directory:
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 in a JMX editor, such as JConsole. See "Accessing ECE Configuration 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:

Copyright © 2019, 2026, Oracle and/or its affiliates.

```
simulate simulator
```

The simulator takes a few seconds to finish processing the workload.

4. Open the **invocation.log** file located in *ECE_home***/logs**. 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***/config/charging-cache-config.xml** file.
2. For the ServiceContext module, change the cache-store configuration entry by replacing the following:

```
<init-param>
  <param-name>cache-store</param-name>
    <param-
value>oracle.communication.brm.charging.notification.internal.coherence.AsynchronousNotificationPublisher</param-value>
```

with this:

Copyright © 2019, 2026, Oracle and/or its affiliates.

```
<init-param>
  <param-name>cache-store</param-name>
    <param-
value>oracle.communication.brm.charging.util.coherence.internal.NoPersistenceCacheStore</param-value>
```

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.

   The sample XML file is available at *PDC_home***/apps/Samples/Examples/PDCSampleCAR.xml**.

   PDCSampleCAR.xml contains two custom rules: **Friends&Family** 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 friends and family calls as follows:
   a. Create a generic selector with the **Friends&Family** custom analyzer rule.
   b. Create the charge offer for the friends and family calling service.
   c. For the charge offer, select the provisioning tag that specifies **Friends&Family**.
   d. Create the charge for the charge offer.
   e. For the charge, include the generic selector with the **Friends&Family** rule.

Copyright © 2019, 2026, Oracle and/or its affiliates.

> **Tip**
>
> You associate the friends and family rule in the generic selector with a *result*: a string value that maps to the rule, such as **Friends&Family**. At run time, ECE uses this result in the charge to apply different rates for calls to friends and family.

5. Verify that the ECE Pricing Updater is started.
6. Publish the PDC pricing data to ECE.

   The Pricing Updater synchronizes the pricing data to ECE.
7. Verify that EM Gateway is started.
8. In BRM, create the customer account, purchase the charge offer, and configure the friends and family phone numbers.

   The BRM customer data updates are incorporated into the ECE cache in real time through EM Gateway.
9. Generate usage for a friends and family call for the customer.

   Use the ECE SDK sample programs to generate usage.
10. Verify that balances are impacted as expected.

After you verify that friends and family calls are processed as expected using the ready-to-use friends and family sample data in PDC and BRM, create your own friends and family configurations.

Verifying That Closed User Group Calls Are Processed

To verify 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.

      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**

Copyright © 2019, 2026, Oracle and/or its affiliates.

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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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:

> **Note**
>
> 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

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:

```
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.

Copyright © 2019, 2026, Oracle and/or its affiliates.

Part VII
ECE Utilities

This part provides information about the utilities provided with Oracle Communications Elastic Charging Engine (ECE). It contains the following chapters:

- Charging Utilities

Copyright © 2019, 2026, Oracle and/or its affiliates.

30
Charging Utilities

You learn about syntax and parameters used by the Oracle Communications Elastic Charging Engine (ECE) charging utilities.

Topics in this document:

- query

query

Use the **query** utility to launch the Oracle Coherence query client and run queries against the ECE Coherence cache. You can use the utility to write scripts that interact with the ECE domain objects, creating CohQL queries. The utility supports all ECE caches and objects. For more information, see "Using the query Utility to Test ECE".

You can run this utility in interactive mode or non-interactive mode.

Location

ECE_home/**bin**

Syntax: Interactive Mode

```
./query.sh
Coherence Command Line Tool
CohQl>
```

Syntax: Non-Interactive Mode

```
./query.sh [-t] [-c] [-s] [-e] [-l "statement"] [-f filename] [-h]
           [-P "CohQLquery"]
```

Parameters: Non-Interactive Mode

-t
Enables trace mode for printing debugging information.

-c
Specifies to exit the utility after processing all command-line arguments.

> **Note**
>
> Do not use this argument when redirecting from standard input, because the utility exits as soon as the command-line arguments have been processed and the redirected input will never be read.

-s
Runs the utility in silent mode. This allows the utility to be used in pipes or filters by redirecting to standard input and standard output.

Copyright © 2019, 2026, Oracle and/or its affiliates.

-e
Runs the utility in extended language mode. This mode allows object literals in update and insert commands.

-l "*statement*"
Runs the specified CohQL statement. Enclose your CohQL statements in single or double quotes.
For more information about CohQL statements, see "Using Coherence Query Language" in *Oracle Fusion Middleware Developing Applications with Oracle Coherence*.

-f *filename*
Processes the CohQL statements in the specified file. The statements in the file must be separated by a semicolon (**;**). The file is an operating system-dependent path and must be enclosed in single or double quotes.

-P "*CohQLquery*"
Runs the specified CohQL SELECT query against the database. You can run the **select Count()** and **select Value()** queries. If you include a WHERE clause, it supports the condition **Customer_id = '***value***'**. For example:

```
./query.sh -P "select value() from Customer where Customer_id = 'Cust#6500000242'"
```

For more information about CohQL queries, see "Using Coherence Query Language" in *Oracle Fusion Middleware Developing Applications with Oracle Coherence*.

-h
Displays the syntax for this utility.

Results

Look in the *ECE_home***/logs/query_out.log** file for errors.

> **Note**
>
> Oracle recommends that you not query the cache for the entire list of customer information. Filtered/ selected query is recommended because the complete customer list would cause ECE to crash.

Copyright © 2019, 2026, Oracle and/or its affiliates.

A

Sample Notification Payloads

You can use the sample notification payloads to view the format of notifications that Oracle Communications Elastic Charging Engine (ECE) publishes into the JSM notification queue.

Topics in this document:

- Aggregated Threshold Breach Event (Aggregated Based on Balance Element ID)
- Billing Event
- Credit Ceiling Breach Event
- Credit Floor Breach Event
- Custom Notification for BRM Gateway
- 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

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 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>
</Notification>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

Billing Event

The payload published for a billing notification uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <BillingNotification>
       <NotificationType>BILLING_NOTIFICATION_EVENT</NotificationType>
       <CustomerId>12345</CustomerId>
       <BillingUnitId>2345</BillingUnitId>
       <ExternalReference>1</ExternalReference>
    </BillingNotification>
</Notification>
```

Credit Ceiling Breach Event

The payload published for a credit limit ceiling breach uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <CreditCeilingBreachNotification>
    <NotificationType>CREDIT_CEILING_BREACH_EVENT</NotificationType>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>1.00</CurrentBalance>
    <CreditCeiling>0</CreditCeiling>
    <AlertType>2</AlertType>
    <Reason>0x01</Reason>
    <OperationType>USAGE</OperationType>
    <SubOperationType>INITIATE</SubOperationType>
   </CreditCeilingBreachNotification>
</Notification>
```

Credit Floor Breach Event

The payload published for a credit limit floor breach uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <CreditFloorBreachNotification>
    <NotificationType>CREDIT_FLOOR_BREACH_EVENT</NotificationType>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>-1.00</CurrentBalance>
    <CreditFloor>0</CreditFloor>
    <AlertType>2</AlertType>
    <Reason>0x01</Reason>
    <OperationType>USAGE</OperationType>
    <SubOperationType>INITIATE</SubOperationType>
   </CreditFloorBreachNotification>
</Notification>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

Custom Notification for BRM Gateway

The payload published for a custom notification that is targeted for BRM Gateway uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification version="3.0.0.0.0">
<CustomNotification>
   <NotificationType>CUSTOM_EVENT_NOTIFICATION_EVENT</NotificationType>
   <OpCode>CUST_MODIFY_CUSTOMER<OpCode>
   <CustomDataMap>
      <CustomDataKey>CustomerId</CustomDataKey>
      <CustomDataValue>123</CustomDataValue>
   </CustomDataMap>
   <CustomDataMap>
      <CustomDataKey>BalanceId</CustomDataKey>
      <CustomDataValue>456</CustomDataValue>
   </CustomDataMap>
   <CustomDataMap>
      <CustomDataKey>CustomerId</CustomDataKey>
      <CustomDataValue>123</CustomDataValue>
   </CustomDataMap>
   <CustomDataMap>
      <CustomDataKey>BalanceId</CustomDataKey>
      <CustomDataValue>456</CustomDataValue>
   </CustomDataMap>
</Notification>
```

External Top-up Event

The payload published for a external top-up notification uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <ExternalTopUpNotification>
     <NotificationType>EXTERNAL_TOP_UP_NOTIFICATION_EVENT</NotificationType>
     <PublicUserIdentities>
            <PublicUserIdentity>1000000</PublicUserIdentity>
     </PublicUserIdentities>
     <CustomerId>137826171</CustomerId>
     <ExternalReference>1</ExternalReference>
     <RequestTime>1325269800000</RequestTime>
     <Id>RECHARGE_ID1</Id>
     <BalanceImpact>
       <ProductId>137826171</ProductId>
       <ProductType>VOICE</ProductType>
       <BalanceItemImpact>
       <BalanceItemId>1</BalanceItemId>
       <BalanceElementCode>FSEC</BalanceElementCode>
       <Quantity>-10</Quantity>
       <ExtendValidityFlag>false</ExtendValidityFlag>
             <ValidFrom>1325269800000</ValidFrom>
             <ValidTo>1388514600000</ValidTo>
                    </BalanceItemImpact
           </BalanceImpact>
    <SubscriberPreferences>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
                <SubscriberPreference PublicUserIdentity="1000001:VOICE, 1000000:VOICE">
      <SubscriberPreferencesInfo>
        <PreferenceName>Language</PreferenceName>
        <PreferenceValue>French</PreferenceValue>
      </SubscriberPreferencesInfo>
             </SubscriberPreference>
   </SubscriberPreferences>
 </ExternalTopUpNotification>
</Notification>
```

First Usage Validity

The payload published for a first usage validity notification uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <FirstUsageValidityNotification>
    <NotificationType>FIRST_USAGE_VALIDITY_INIT_NOTIFICATION_EVENT</NotificationType>
    <CustomerId>12345</CustomerId>
    <ExternalReference>1</ExternalReference>
    <BalanceId>12345</BalanceId>
    <validity>
        <BalanceElementId>100025</BalanceElementId>
        <BalanceItemId>1</BalanceItemId>
        <ValidFrom>1325269800000</ValidFrom>
        <ValidTo>1388514600000</ValidTo>
    </validity>
     <SubscriberPreferences>
              <SubscriberPreference PublicUserIdentity="1000001:VOICE, 1000000:VOICE">
       <SubscriberPreferencesInfo>
          <PreferenceName>Language</PreferenceName>
          <PreferenceValue>French</PreferenceValue>
       </SubscriberPreferencesInfo>
              </SubscriberPreference>
     </SubscriberPreferences>
      </FirstUsageValidityNotification>
</Notification>
```

Life-Cycle Transition

The payload published for a life cycle transition notification uses the following format:

```
<?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>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
            <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 version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <ReplenishPoidIdNotification>
       <NotificationType>REPLENISH_POID_ID_NOTIFICATION_EVENT</NotificationType>
       <SchemaName>1</SchemaName>
       <Quantity>10000</Quantity>
    </ReplenishPoidIdNotification>
</Notification>
```

Spending Limit

The payload published for a spending limit notification uses the following format:

```
<?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 version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <CreateSubscriberPreference>
     <NotificationType>SUBSCRIBER_PREFERENCE_NOTIFICATION_EVENT</NotificationType>
     <CustomerId>340876</CustomerId>
     <ProductInfo>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
      <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 version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <ModifySubscriberPreference>
     <NotificationType>SUBSCRIBER_PREFERENCE_NOTIFICATION_EVENT</NotificationType>
     <CustomerId>customer1</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>
    </ModifySubscriberPreference>
</Notification>
```

The payload published for deleting a subscriber preference notification uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <DeleteSubscriberPreference>
     <NotificationType>SUBSCRIBER_PREFERENCE_NOTIFICATION_EVENT</NotificationType>
     <CustomerId>customer1</CustomerId>
     <ProductInfo>
      <ProductId>12345</ProductId>
      <PublicUserIdentities>
       <PublicUserIdentity>9886753556</PublicUserIdentity>
       <PublicUserIdentity>login</PublicUserIdentity>
      </PublicUserIdentities>
     </ProductInfo>
     <SubscriberPreferencesInfo>
      <PreferenceName>Language</PreferenceName>
      <PreferenceValue>English</PreferenceValue>
     </SubscriberPreferencesInfo>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
     <SubscriberPreferencesInfo>
      <PreferenceName>Channel</PreferenceName>
      <PreferenceValue>Email</PreferenceValue>
     </SubscriberPreferencesInfo>
    </DeleteSubscriberPreference>
</Notification>
```

Threshold Breach Event (Breach Direction Down)

The payload published for a threshold breach (breach direction down) uses the following format:

```
<?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 Up)

The payload published for a threshold breach (breach direction up) uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <CreditThresholdBreachNotification>
    <NotificationType>THRESHOLD_BREACH_EVENT</NotificationType>
    <BalanceElementId>840</BalanceElementId>
    <BalanceElementCode>USD</BalanceElementCode>
    <CurrentBalance>-4.00</CurrentBalance>
    <ThresholdAmount>-4.5</ThresholdAmount>
    <ThresholdPercent>55.0</ThresholdPercent>
    <BreachDirection>THRESHOLD_BREACH_UP</BreachDirection>
    </CreditThresholdBreachNotification>
</Notification>
```

Top-up Event

The payload published for a top-up uses the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification>
    <RARNotification>
       <NotificationType>TOP_UP_NOTIFICATION_EVENT</NotificationType>
       <PublicUserIdentities>
            <PublicUserIdentity>123</PublicUserIdentity>
            <PublicUserIdentity>456</PublicUserIdentity>
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

```
        </PublicUserIdentities>
        <ActiveSessions>
             <ActiveSessionId>SESSION1</ActiveSessionId>
             <ActiveSessionId>SESSION2</ActiveSessionId>
        </ActiveSessions>
        <ProductType>VOICE</ProductType>
        <ProductId>test</ProductId>
        <CustomerId>12345</CustomerId>
        <DuplicateEvent>True</DuplicateEvent>
    </RARNotification>
</Notification>
```

Enriched Notification

The payload published for a custom enriched notification with the customDataMap populated with attributes like original balance, current balance, event timestamp, for example:

```
[ThresholdBreachServiceEventImpl{ balanceId='BALANCE', beCode='USD', beId=840,
      currentBalance=-179.00, thresholdAmount=-180, publicUserIdentities='[6500000001]',
      thresholdPercentage=10.0, breachDirection='THRESHOLD_BREACH_UP', alertType='2',
reason='0x01',
      validityUpdateOnly='false', customerId='CUSTOMER',
eventTimeStamp='2024-01-17T08:50:17.096Z',
      operationType='USAGE',
      subOperationType='UPDATE'[ExternalNotifiableServiceEventImpl{sessionId='null',
      clientRoutingData=null[ThresholdBreachServiceEventImpl{serviceEventId=null,
      serviceEventType=THRESHOLD_BREACH_EVENT, notificationVersion=7.0.0.0.0,
      customDataMap={OriginalBalance=UnitValue{quantity=0, unit=Occ},
CalledNumber=650000002,
      NotificationType=THRESHOLD_BREACH_EVENT, CurrentBalance=UnitValue{quantity=-50.0,
unit=Occ},
      EventTimeStamp=2024-01-17T08:50:17.096Z,
ExtensionThresholdKey1=ExtensionThresholdValue1,
      CustomerId=CUSTOMER, SessionId=SESSION ID, BalanceId=BALANCE,
CallingNumber=6500000001},
      puidPrefMap={}}]'}]'}]
```


Copyright © 2019, 2026, Oracle and/or its affiliates.

B

Specifications and Standards Compliance in ECE

Learn about the specifications and standards used in Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- About Specifications and Standards Compliance

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"

Copyright © 2019, 2026, Oracle and/or its affiliates.

- "3GPP TS 32.290 Telecommunication management; Charging management; 5G system; Services, operations and procedures of charging using Service Based Interface (SBI)
- "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 2865, "Remote Authentication Dial In User Service (RADIUS)," June 2000, RADIUS. Updated by: RFC 2868, RFC 3575, RFC 5080.
- 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.

Copyright © 2019, 2026, Oracle and/or its affiliates.