

# Oracle® Communications Offline Mediation Controller User's Guide



Release 12.0  
E91425-04  
June 2022

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Copyright © 2017, 2022, 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 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" or "commercial computer software documentation" 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, and MySQL 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.

# Contents

## Preface

---

Audience	vi
Documentation Accessibility	vi
Diversity and Inclusion	vi

## 1 Overview of Offline Mediation Controller

---

About Offline Mediation Controller	1-1
Offline Mediation Controller System Architecture	1-5

## 2 About Configuring Nodes and Node Chains

---

About Creating Node Chains	2-1
About Node Configuration	2-3
About Configuring EC and AP Nodes	2-4
About Editing Rule Files	2-4
About Node Routing	2-5
About Distributed Node Chains	2-6

## 3 Configuring Node Chains

---

Configuring Node Chain Input	3-1
Configuring Input from BRM	3-2
Configuring Node Chain Output	3-2
Managing Sequencing	3-3
About Record-Level Sequencing	3-3
About File-Level Sequencing	3-3
Customizing the Sequencing File Name Syntax	3-4
Managing Duplicate Records	3-5
About Removing Duplicate CDR Files	3-6
About Removing Duplicate Records	3-6
About Aggregation	3-7
About Time-Based Aggregation of CDRs	3-7

About Volume-Based Aggregation of CDRs	3-8
About Mapping and Enrichment	3-9
Record Enhancement for BRM Charging	3-9
Filtering Records Based on Data	3-10
Restoring Excluded Records	3-10
Adding Data to Records	3-10
Routing Records Based on Data	3-11
File and Record Processing Options	3-11
Processing CDRs at File Level	3-11
Configuring the File-Level Transaction Threshold	3-12
Viewing the File-Level Transaction Performance	3-12
Improving EP Performance by Using Multi-Threaded Processing	3-12
Improving CC Node Processing Performance with Multi-Threaded Processing	3-13
Improving Record Archive File Performance with Bulk Read and Write Processing	3-13
Discarding Records	3-13

## 4 Suspending and Recycling Records

---

About Suspending and Recycling Records and Batches	4-1
About Suspense and Recycle Node Chains	4-1
About Searchable Fields	4-4
Configuring the File-Level Transaction Threshold	4-4
About record Field Mapping	4-5
Working with SQL Files to Restore a Recycled CDR	4-5
Installing the Sample Detail and Header SQL Files	4-6
Updating and Applying a Sample SQL File in the Recycle EP Node	4-6
SQL Statement Examples	4-6
Configuring Suspense Errors	4-7
Configuring New Error Codes	4-8
Working with the Offline Mediation Controller Error Code API	4-9

## 5 Sending Records to Elastic Charging Engine

---

About the ECE Cartridge Pack	5-1
Rule File	5-1
Configuration Block	5-2
Input Record Block	5-4
Output Record Block	5-4
Expose Block for PAYLOAD	5-5
Expose Block for Payload Data Type	5-5
Expose Block for Usage Object	5-6

## A Suspense Error Configuration Examples

---

About These Examples	A-1
Product Type Error Event NPL Example	A-1
Batch CDR File Error NPL Example	A-4

## Glossary

---

# Preface

This document describes how to implement and use Oracle Communications Offline Mediation Controller.

## Audience

This document is intended for anyone who installs, configures, administers, or customizes Offline Mediation Controller.

## Documentation Accessibility

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

### **Access to Oracle Support**

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

## Diversity and Inclusion

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

# 1

## Overview of Offline Mediation Controller

Learn about Oracle Communications Offline Mediation Controller.

Topics in this document:

- [About Offline Mediation Controller](#)
- [Offline Mediation Controller System Architecture](#)

### About Offline Mediation Controller

Offline Mediation Controller is a mediation application for communications services, such as wireless voice and data, content downloads, and voice over IP (VoIP). Offline Mediation Controller receives data from network devices, normalizes and transforms it, and sends it to systems and applications such as Oracle Communications Elastic Charging Engine (ECE).

Offline Mediation Controller processes data to support services such as:

- Wireless: GSM, GPRS, CDMA
- VoIP
- IP

Offline Mediation Controller sends data to systems and applications such as:

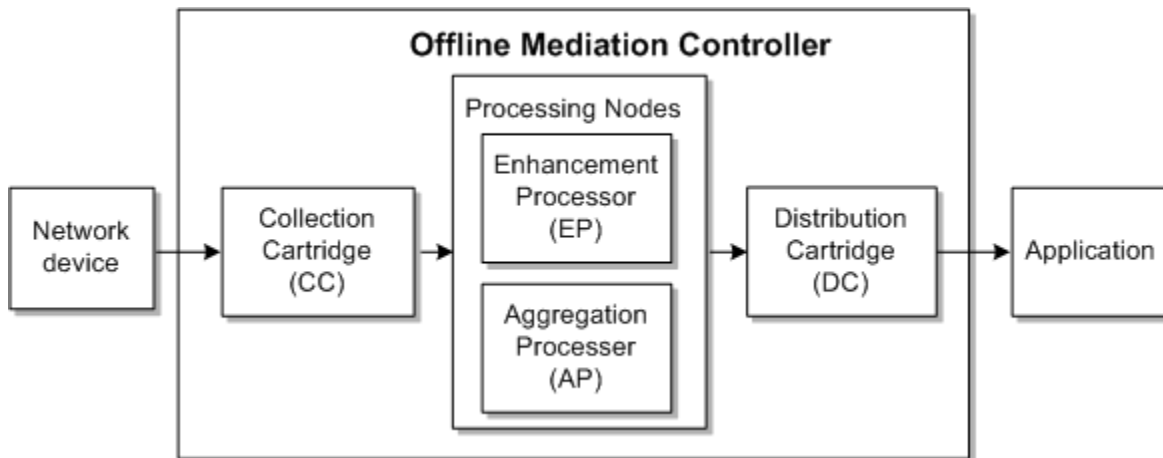
- Charging and billing systems
- Performance Management and Service Quality Management systems
- Inventory systems

Offline Mediation Controller receives input from a network device and translates the data into network accounting records (NARs). A NAR uses an internal format that is used by all Offline Mediation Controller components. A NAR can contain information used for charging, such as the calling number, called number, origin, event start time, event end time, amount of data downloaded, IP address, and so on.

Offline Mediation Controller collects data from the source application or the network, normalizes the data to NAR format, and then processes and prepares it for one or more downstream systems.

[Figure 1-1](#) shows the Offline Mediation Controller work flow.

Figure 1-1 Offline Mediation Controller Work Flow



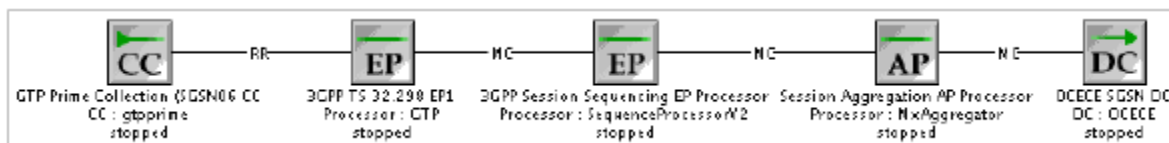
Processing is performed by four types of nodes:

- **Collection cartridge (CC).** CC nodes collect raw data from devices outside of the Offline Mediation Controller system and transform the data into NARs that can be processed by Offline Mediation Controller.
- **Enhancement processor (EP).** EP nodes add, modify, or delete data in NARs. For example, an EP node can add data based on IP ports, or source and destination IP addresses detected in incoming NARs. You can also use the EP node to enhance NARs with information from outside of the Offline Mediation Controller system.
- **Aggregation processor (AP).** AP nodes aggregate data and records. For example, long-duration sessions can be received in multiple call detail records (CDRs). You can use the AP node to aggregate the CDRs into one NAR file.
- **Distribution cartridge (DC).** DC nodes distribute the network data collected and processed by other functional nodes. The DC node converts NAR files to a specified output format and moves the resulting files to its output queue.

You add these nodes to a *node chain*. The first node is always a CC node, and the last node is always a DC node. The processing nodes (EP and AP) are optional, and can be in any order. You typically use multiple EP nodes for different types of processing, such as mapping, normalizing, and so on.

Figure 1-2 shows a sample node chain that processes GPRS events for Elastic Charging Engine.

Figure 1-2 Node Chain



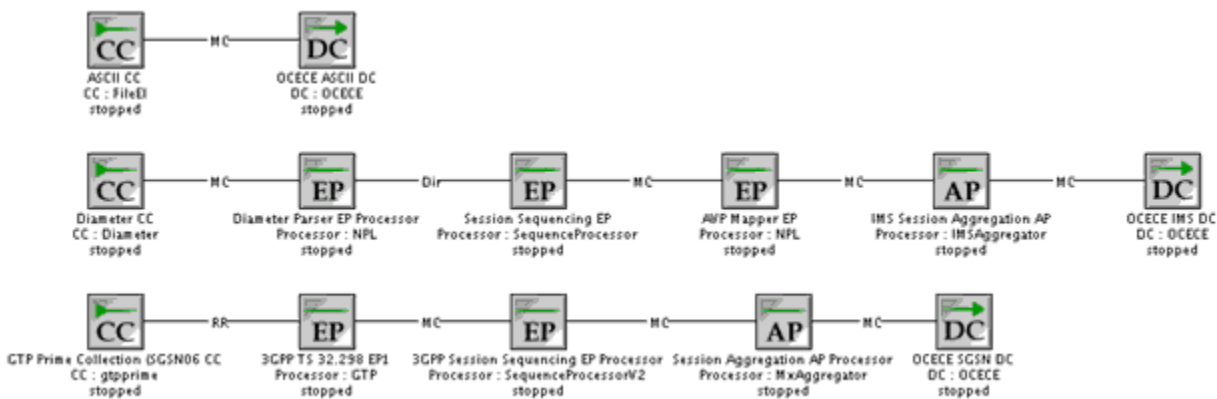
In this example:



- The CC node reads GPRS records and transforms them into NARs.
- The EP nodes perform normalization and record sequencing to ensure that events are processed correctly.
- The AP node aggregates partial records into a single record.
- The DC node sends the NARs to the Elastic Charging Engine.

Depending on your service offerings, you can create multiple node chains that handle different types of input, and provide output to the same system. [Figure 1-3](#) shows a typical configuration for receiving input from ASCII, IMS, and SGSN, and providing output to Elastic Charging Engine.

**Figure 1-3 Multiple Node Chains**

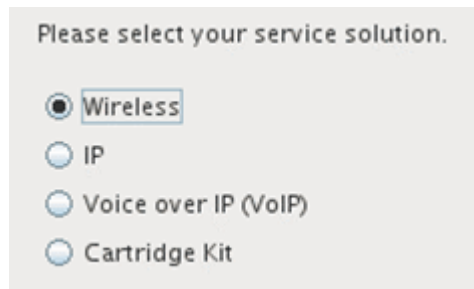


To create a node chain, you configure the nodes that are required to process events for a particular input and output; for example, a node chain that processes wireless events to send to ECE. When you configure each node, you work with two aspects of the node:

- **Node configuration:** This specifies how the node functions in the node chain. For example, when defining the node configuration for an EP node that checks for duplicate records, you specify the directory that holds duplicate records, the number of records in a duplicate-record file, and the next node in the node chain.
- **Rule file:** This defines how the node handles each record and carries out its work. For example, a rule file for an EP node that checks for duplicate records defines the field in the record that is used for detecting duplicate records, such as the calling number or session ID.

Offline Mediation Controller includes predefined nodes and rule files to support specific communication services: wireless, IP, and VoIP. When you create a node, you start by choosing the solution you are delivering (as shown in [Figure 1-4](#)), or you can choose a cartridge kit that is independent of a solution.

**Figure 1-4 Selecting a Service Solution**



The image shows a dialog box titled "Please select your service solution." with four radio button options: "Wireless" (selected), "IP", "Voice over IP (VoIP)", and "Cartridge Kit".

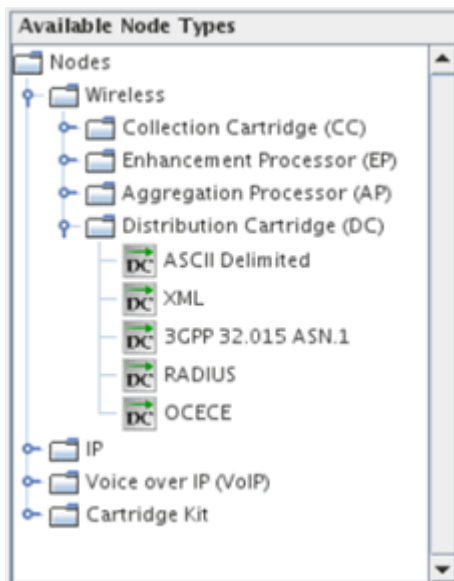
Each solution includes a set of solution-specific nodes and rule files to choose from. For example, to configure a node chain for wireless events, you can use predefined EP nodes for processing ASN data, CDR sequence management, and record filtering.

- For wireless services, Offline Mediation Controller delivers the Charging Gateway Functionality (CGF) as specified by the 3GPP for wireless GSM/GPRS/UMTS networks. It goes beyond the basic CGF by supporting numerous applications and services.
- For IP services, Offline Mediation Controller includes pre-integrated support for numerous industry-leading routers, switches, and IP service platforms. Raw service and network usage data is collected, aggregated, and enhanced with QoS and customer-specific information.
- For VoIP services, Offline Mediation Controller supports next generation VoIP telephony platforms as well as legacy protocols and record formats, enabling migration from legacy voice networks to VoIP.

In addition to solution-based nodes and rule files, you can use application-specific cartridges. Cartridges include specialized nodes and rule files. For example:

- Use the Oracle CDR Format cartridge to configure a CC node when you integrate Offline Mediation Controller with Oracle Communications Billing and Revenue Management (BRM).
- Use the suspense and recycle nodes to integrate Offline Mediation Controller with BRM Suspense Manager.

After you install a cartridge, the nodes in the cartridge are available in the Administration Client. [Figure 1-5](#) shows how node types are organized by domain (wireless, and so on) and by type (CC, EP, and so on). In this figure, the OCECE DC node is present, indicating that the ECE Cartridge Pack has been installed.

**Figure 1-5 Available Node Types in Administration Client**

Cartridges are installed as JAR files. Each cartridge supports a specific domain of functional area. You can use the Cartridge Development Kit (CDK) to create custom cartridges that support input from new network elements, or can output records to custom systems and applications.

## Offline Mediation Controller System Architecture

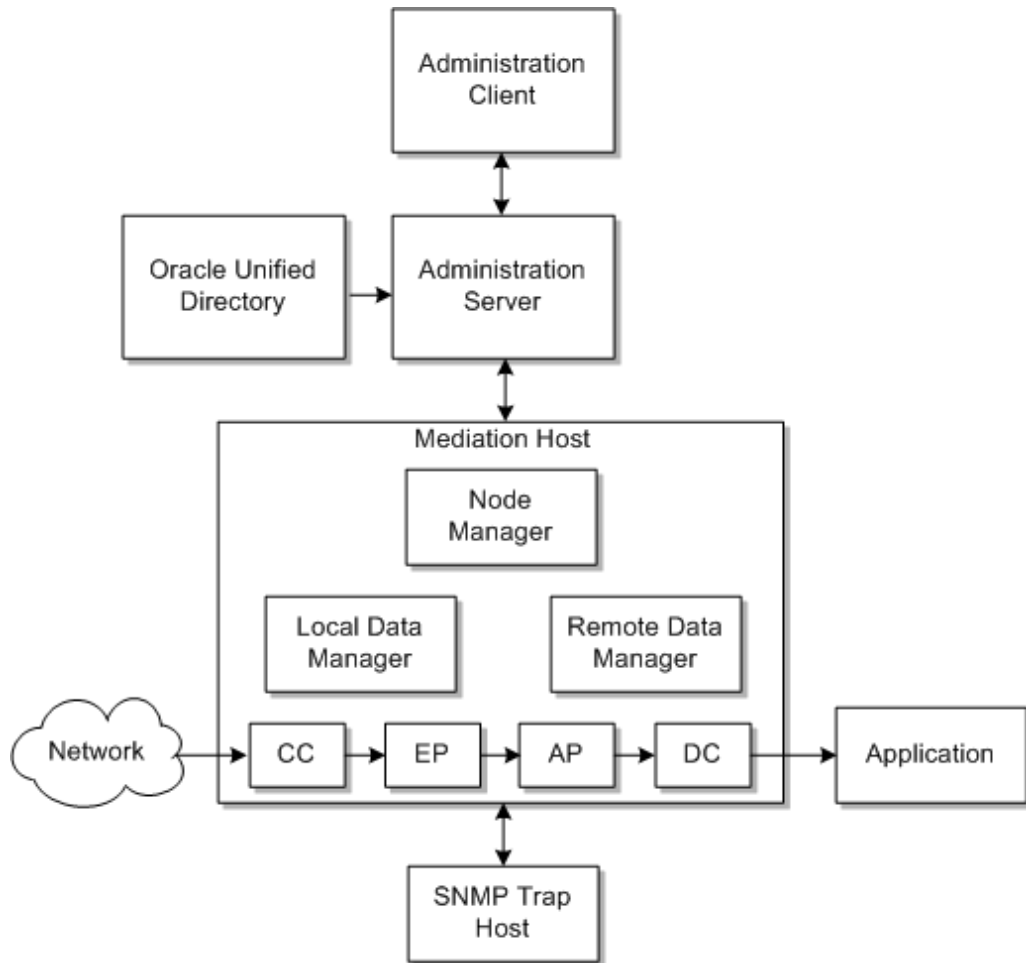
Offline Mediation Controller include three main components:

- *Node managers* run the nodes in node chains.
- The *administration server* runs the node managers, and manages all of the Offline Mediation Controller processes.
- You configure node chains and administer the system by using the *Administration Client*.

To run Offline Mediation Controller, you start all three components. You can also start and run individual nodes.

[Figure 1-6](#) shows the Offline Mediation Controller system architecture.

Figure 1-6 Offline Mediation Controller System Architecture



In Figure 1-6:

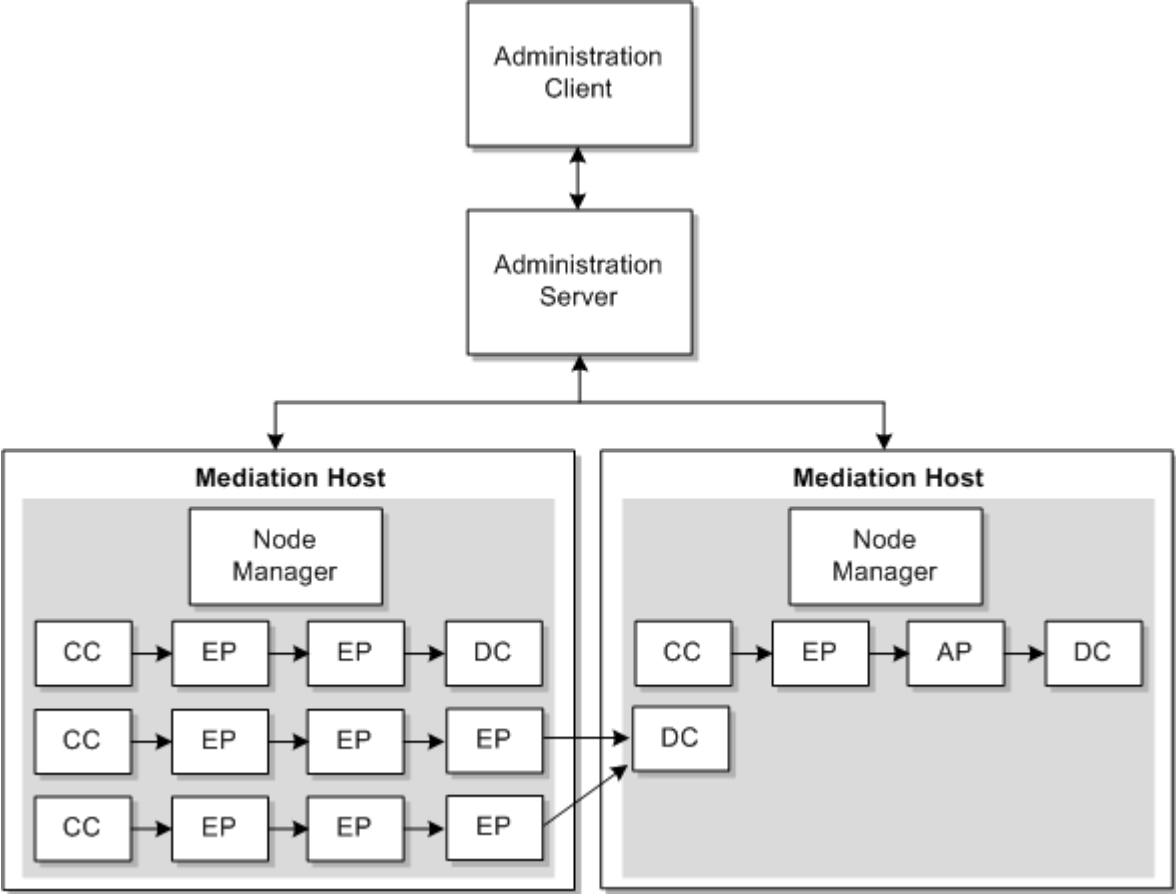
- Administration Client is a GUI application that you use for creating node chains and editing rule files. You also use Administration Client for administering Offline Mediation Controller. For example, you can use it to manage users and define instances of system components.
- The administration server is a process that passes commands between Administration Client and the node managers, which in turn control the nodes. The administration server manages log files and manages the data flow in the entire Offline Mediation Controller system. You can run one primary administration server and one backup administration server.
- A mediation host is a server on which nodes and node managers run:
  - Node managers run the nodes in a node chain. You send commands to a node manager to stop and start nodes.
  - Local data managers pass data from one node to another on the same mediation host.
  - Remote data managers pass data from a node on one mediation host to a node on a different mediation host.

A mediation host is not a component that runs. You do not stop and start a mediation host. Instead, it provides a location on the system where nodes and node managers can run. You typically configure multiple mediation hosts to distribute processing.

- Oracle Unified Directory manages Offline Mediation Controller users.
- The optional SNMP trap host is an IP host that receives SNMP trap messages from the Offline Mediation Controller system. Offline Mediation Controller issues SNMP trap messages to send alarms to network management systems.

Offline Mediation Controller is typically distributed on multiple systems. You can run one or more administration servers, and each administration server connects to node managers hosted on multiple mediation host systems. [Figure 1-7](#) shows a configuration with multiple mediation hosts and node managers. Using multiple mediation hosts enables more efficient use of system resources and improves performance.

**Figure 1-7 Distributed Processing on Two Mediation Hosts**



# 2

## About Configuring Nodes and Node Chains

You configure node chains in Oracle Communications Offline Mediation Controller by using the Administration Client application.

Topics in this document:

- [About Creating Node Chains](#)
- [About Node Configuration](#)
- [About Configuring EC and AP Nodes](#)
- [About Editing Rule Files](#)
- [About Node Routing](#)
- [About Distributed Node Chains](#)

### About Creating Node Chains

Before you can configure node chains, you need to install and configure the Offline Mediation Controller components.

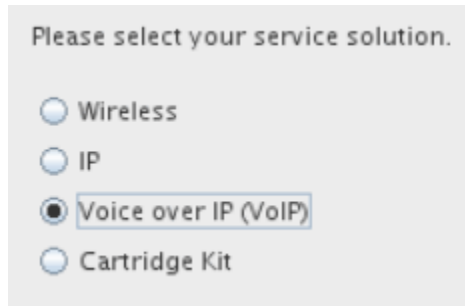
1. Start the Offline Mediation Controller components:

- Oracle Universal Directory
- One or more node managers
- The administration server
- Administration Client

For information, see "Starting and Stopping Offline Mediation Controller" in *Offline Mediation Controller System Administrator's Guide*.

2. In Administration Client, create mediation hosts for each instance of a node manager. To create a mediation host, you assign the host name and port number of a node manager. After a mediation host is created, you can create nodes and node chains.
3. When you create a new node on the mediation host, Administration Client presents a series of choices that enable you to select a node for your solution. First, you select the solution type:

**Figure 2-1 Selecting a Service Solution**

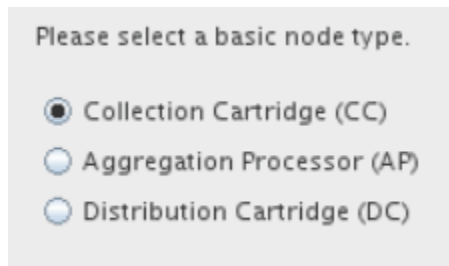


Please select your service solution.

- Wireless
- IP
- Voice over IP (VoIP)
- Cartridge Kit

Then, you select the node type:

**Figure 2-2 Selecting a Node Type**

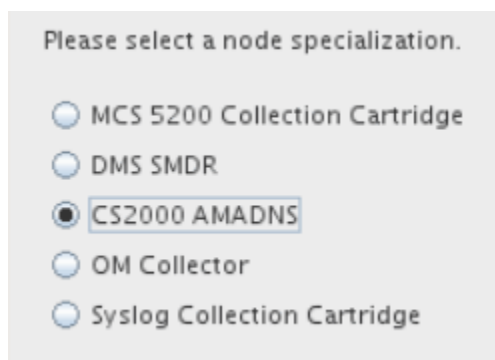


Please select a basic node type.

- Collection Cartridge (CC)
- Aggregation Processor (AP)
- Distribution Cartridge (DC)

Depending on the solution type and node type, you then select a node specialization:

**Figure 2-3 Selecting a Node Specialization**



Please select a node specialization.

- MCS 5200 Collection Cartridge
- DMS SMDR
- CS2000 AMADNS
- OM Collector
- Syslog Collection Cartridge

4. Each node requires configuration and might require you to edit the rule file. All configurations and editing of rule files are performed in Administration Client.
5. When all nodes are created and configured, create a node chain by doing the following:
  - Create the CC, EP, AP, and DC nodes as necessary.
  - Specify the order of the nodes in the node chain by dragging and dropping connection lines from the output of one node to the input of another node.

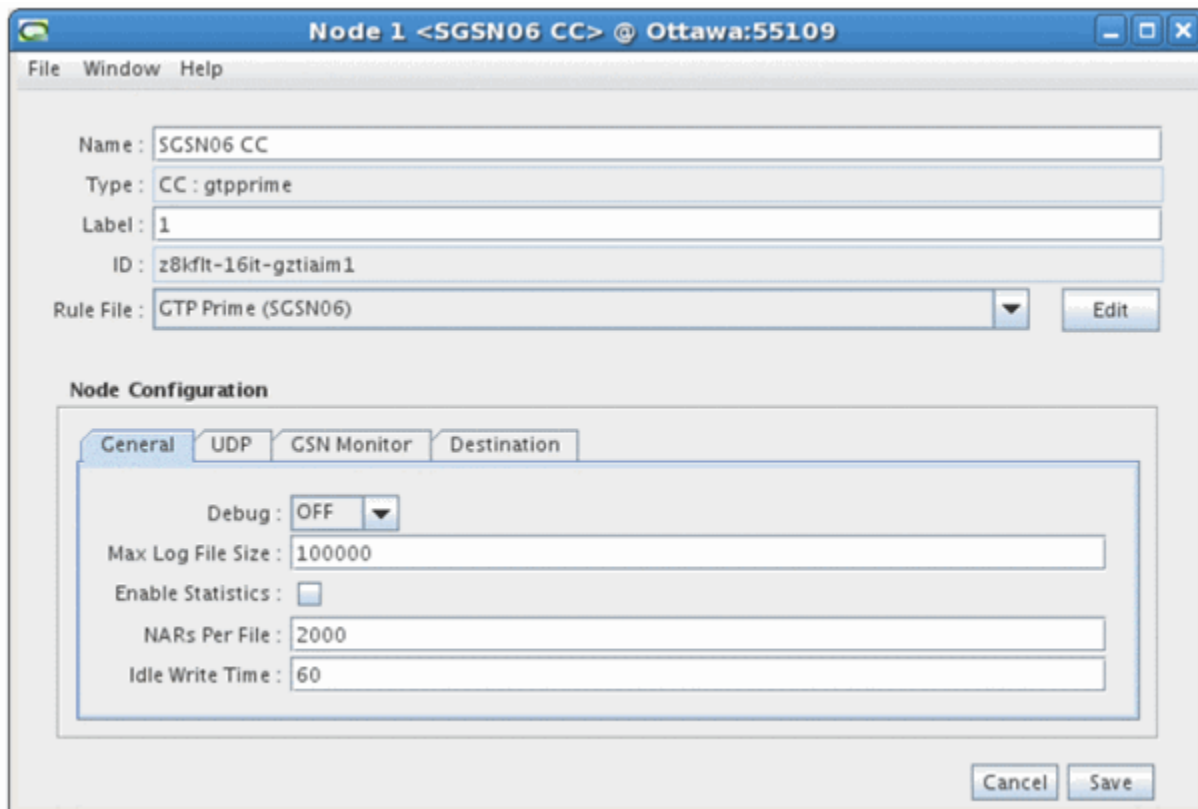
- In the CC node, set up a method for Offline Mediation Controller to receive input from an external system.
  - In the DC node, set up a destination for Offline Mediation Controller to send data to.
6. After the nodes are configured and connected, start the node chain.

## About Node Configuration

To configure nodes in Administration Client, choose the type of node that you want and then edit the configuration in a dialog box. Each node can have different configuration options depending on the node functionality. For example, CC nodes can include file input configuration, and DC nodes have no configuration for destination nodes.

Figure 2-4 shows a node configuration dialog box.

Figure 2-4 Node Configuration Dialog Box



In this figure:

- All nodes include an option to select a rule file and edit it.
- All nodes include a **General** tab, which specifies administration parameters, such as log file configuration. Each type of node can have different requirements, so the options vary. For example, some nodes raise an alarm if no input is received in a specified amount of time.



- CC, EP, and AP nodes include a **Destination** tab. This tab lists the nodes that the node connects to. You use the **Destination** tab to specify the type of routing to use, such as multicast or round robin. See "[About Distributed Node Chains](#)".
- All other tabs in a node configuration depend on the node functionality. For example, in [Figure 2-4](#), the node needs to use User Datagram Protocol (UDP) to get data, so the **UDP** tab configures the UDP connection.

For information about the options in each dialog box tab, see *Offline Mediation Controller Online Help*.

## About Configuring EC and AP Nodes

In addition to general and destination configurations, EC nodes and AP nodes typically include node-specific configuration parameters. For example:

- The duplicate check EP node uses a **De-Duplication** tab to configure options such as where to store the files that hold duplicate records, the number of records per file, and so on.
- The record recycling EP node uses a **Database Info** tab to configure the connection information required to send records to a database.
- The session aggregation AP node uses an **Aggregator** tab to specify how often to send aggregated records to the next node in the node chain.

## About Editing Rule Files

You edit rule files when you configure nodes in Administration Client. You open the rule file, edit it, and perform a test compilation. If compilation fails, you can edit it again. When compilation succeeds, save the file.

Rule files define the functionality that a node carries out. For example:

- The rule file for a CC node contains that mapping and normalizing rules that translate incoming data into a NAR.
- The rule for a DC node contains that rules that transform a NAR into the output needed by the target system.
- The rule file for an EP node contains functions that manipulate data.

In many cases, a node has a pre-configured rule file that does not need editing. In some cases, a node can include multiple rule files that you can choose from, such as to specify a different output. In some cases, you need to select a rule file and edit it.

Editing a rule file can be as simple as providing a value in a name-value pair. For example, the following entry in a rule file specifies the field to use when checking for duplicate records:

```
DUP_CHECK_KEYS "calling_number session_id seq_no";
```

You can also use Java hooks to implement more complex mappings and configurations. For example, this Java hook implements a JDBC connection:

```
Config {  
  JDBCdriver "oracle.jdbc.driver.OracleDriver";  
  JDBCurl "jdbc:oracle:thin:@%DBHOST%:%DBPORT%:%DBSID%";  
}
```

Java hooks are provided by each cartridge. For example, the Oracle CDR Format Cartridge includes many Java hooks to validate input.

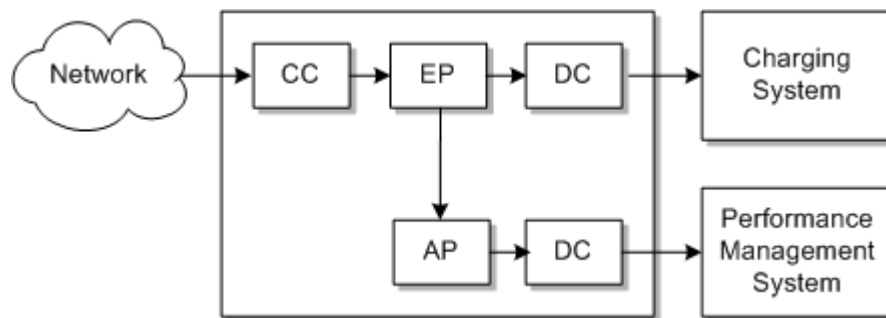
For more information about using Java hooks with NPL, see "Java hooks" in *Offline Mediation Controller Cartridge Development Kit NPL Reference Guide*.

## About Node Routing

You can use various types of routing in a node chain to process NARs in different ways and implement load balancing. You can use:

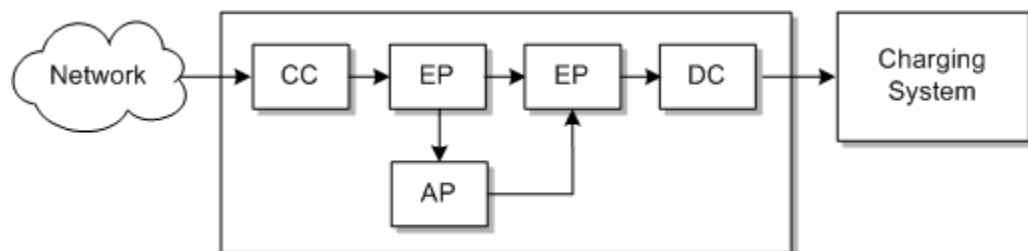
- *Multicast routing* to branch a node chain and send all of the NAR files to both branches. [Figure 2-5](#) shows a node chain with one input and two outputs. In this configuration, the EP node sends all of the NAR files to two different nodes: one for output to a charging system, and one to an aggregation processor that sends the NARs to a performance management system.

**Figure 2-5 Node Chain With Multicast Routing**



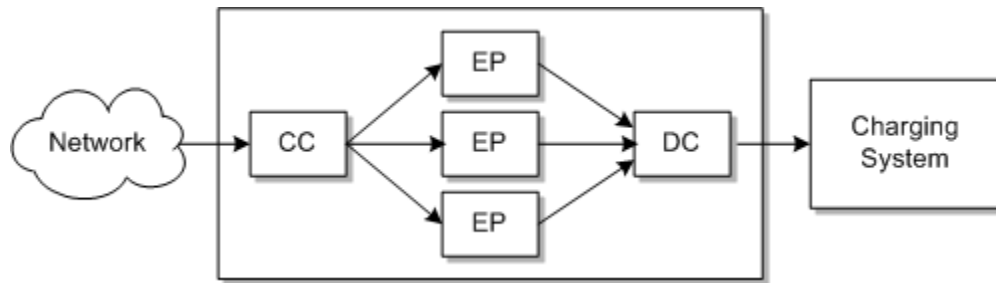
- *Directed routing* to send NAR files to different branches. For example, you can branch a node chain based on the value of a field in the NAR. [Figure 2-6](#) shows a node chain that implements an additional aggregation step for only some of the NAR files.

**Figure 2-6 Directed Routing**



You can improve performance by creating multiple instances of the same processing node. You can then route NAR files to different instances to process the files in parallel. [Figure 2-7](#) shows a CC node configured to send NAR files to one of three different EP nodes.

**Figure 2-7 Distributing NAR Files to Multiple Instances of the Same EP Node**



To send NAR files to different nodes as shown in [Figure 2-7](#), you can use:

- *Round robin routing*, which rotates to which node the NAR file is sent.
- *Modulus routing*, which sends the NAR file to a node based on a field value in the NAR.

See *Offline Mediation Controller Online Help* for information about configuring NAR file routing.

You configure NAR routing in two places:

- In the **Destination** tab when you configure the node. For example, you can choose multicast or round robin routing. [Figure 2-8](#) shows a node configured to use multicast routing for one destination node, but no routing for the other nodes.

**Figure 2-8 Routing Configured in the Destination Tab**

Label	Name	Host	Type	Enable	Routing
2	SGSN06 Parsing Enh...	Ottawa:55109	Processor : GTP	<input checked="" type="checkbox"/>	Multicast
3	3GPP Session	Ottawa:55109	Processor : Se...	<input type="checkbox"/>	None
4	Session Aareator ...	Ottawa:55109	Processor : M...	<input type="checkbox"/>	None

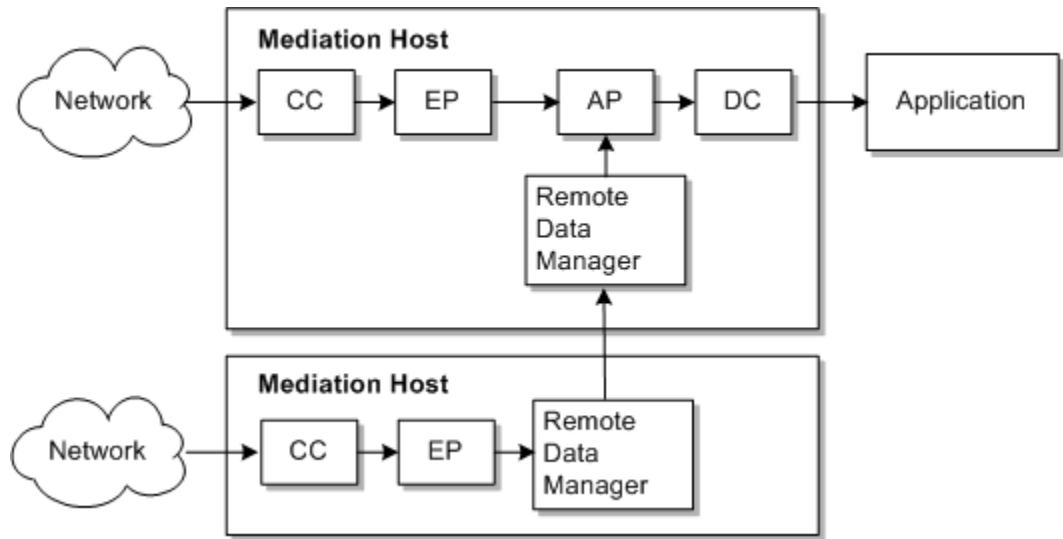
- When you drag and drop a connection from one node to another. If routing is enabled for the destination node, you enter the routing method. For example, to configure distributed routing, you specify the conditions that enable the NAR file to be sent to the destination node.

## About Distributed Node Chains

You can manage performance by configuring node chains on multiple mediation hosts.

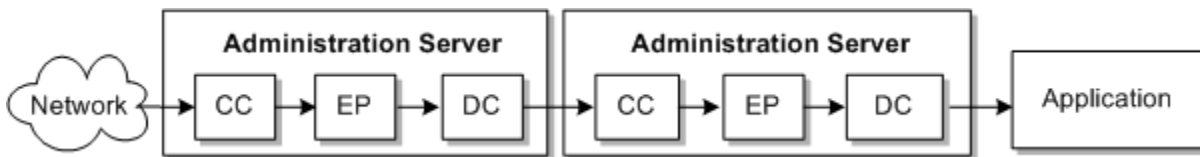
[Figure 2-9](#) shows a node chain that runs on two mediation hosts. This node chain receives input from two devices, but outputs to one application. The Remote Data Manager sends the output NAR of an EP node on one mediation host to the input of the AP node in the other mediation host.

Figure 2-9 Running a Node Chain on Two Mediation Hosts



You can run multiple instances of Offline Mediation Controller by running multiple pairs of administration server primary and backup servers. If you do, you can send NAR files from the DC node in one instance of Offline Mediation Controller to the CC node in another instance. [Figure 2-10](#) shows node chains connected across administration servers.

Figure 2-10 Connecting Node Chains on Different Administration Servers



# 3

## Configuring Node Chains

Learn about the functionality you can configure in Oracle Communications Offline Mediation Controller, such as checking for duplicate records.

Topics in this document:

- [Configuring Node Chain Input](#)
- [Configuring Input from BRM](#)
- [Configuring Node Chain Output](#)
- [Managing Sequencing](#)
- [Managing Duplicate Records](#)
- [About Aggregation](#)
- [About Mapping and Enrichment](#)
- [Routing Records Based on Data](#)
- [File and Record Processing Options](#)
- [Discarding Records](#)

### Configuring Node Chain Input

All input into Offline Mediation Controller comes through a collection cartridge (CC) node. To receive input, Offline Mediation Controller supports the following methods, depending on the capabilities of the network device providing input:

- You can configure the CC node to look in a specified directory. You need to configure the network device to send files to that directory.
- You can configure the CC node to use FTP to retrieve files from an external location.
- You can configure the CC node to communicate directly with the network element. For example, you can configure a CC node to receive records from a Diameter client.

If the CC node receives input from a network connection, such as from a Diameter client, you can configure options such as the port to connect to and time out parameters.

If the CC node receives input in files, you can configure file management settings, such as:

- How to recognize which files the node should process, usually by identifying a file suffix or prefix.
- Specify to change the file suffix after the CC node processes it.
- Specify the FTP connection settings to retrieve files by using FTP.
- Specify whether to delete input files after processing. If you choose not to delete the input files, they are moved from the input directory to the directory specified in the **inputBackupDir** property in the *OMC\_home/config/node\_id/general.cfg* file for the node.  
By default, **inputBackupDir** is set to *OMC\_home/backup/node\_id*.

In addition, you might edit a rule file to customize how data is handled, or specify the type of rule file that applies the input format.

The following nodes handle protocol-specific input:

- The GTP Data Collection CC node receives GSM data directly from the network.
- The GTP Prime Collection CC node receives data from the GPRS Tunneling Protocol (GTP) protocol.

## Configuring Input from BRM

The Oracle CDR format is used by Oracle Communications Billing and Revenue Management (BRM) to process CDRs for offline charging. The Oracle CDR Format CC node collects CDR input files conforming to the Oracle CDR format and parses it into a network accounting record (NAR).

The Oracle CDR Format CC node uses the schema information in a schema file to parse the CDR files conforming to the Oracle CDR format into NARs. If the CDR files you process include information that has no corresponding fields in the default schema file, you can configure a custom schema file to accommodate the custom data.

To use a custom schema, copy your new schema file into the *OMC\_home/config/sol42/schema* directory. You then choose your custom schema when configuring the Oracle CDR Format CC node in Administration Client.

## Configuring Node Chain Output

Node chain output is created by a distribution cartridge (DC) node. The DC node reads the NAR and converts it to the format required by the destination.

Offline Mediation Controller supports several methods to provide output to external systems. For example:

- You can generate files in various formats and send them to an external system by using FTP.
- You can use JDBC to send output to a database. You can configure the DC node rule file to enable sending large objects, such as images, audio, and multimedia, to the database.
- You can connect directly to an external system and send data directly to their receiving interface. For example, you could use ASN.1.

When you configure a DC node, most of the configuration specifies how to send the output. For example:

- If the output is a file sent by FTP, you configure the output directory, file name attributes, FTP connection information, and so on.
- If the output is to a database, you configure database connection information, such as the SID, host name, and so on.

You can also configure how to handle files after output. For example, you could specify to archive or delete the files.

## Managing Sequencing

Sequencing ensures that records are processed in the correct order. You can configure sequencing at:

- The record level. See "[About Record-Level Sequencing](#)".
- The file level. See "[About File-Level Sequencing](#)".

You can also customize the file name syntax for file-level sequencing. See "[Customizing the Sequencing File Name Syntax](#)".

### About Record-Level Sequencing

Sequencing EP nodes detect records that are not in the correct order and then correct the processing order. If the EP node discovers records that are not in sequence, it holds on to those records until it can sequence them, or until the flush timer expires. If the EP node is able to sequence the records, it flushes them right away. If the EP node is not able to sequence the records, it flushes them when the timer expires. You can set the timer when you configure the node.

The logic for determining the correct sequence is defined in the node rule file. You usually do not need to edit the rule file.

You can use the following EP nodes for sequencing records:

- Sequencing EP node
- 3GPP Session Sequencing EP node

### About File-Level Sequencing

CC nodes process input files based on the alphabetical order of the CDR input file name. You can also configure the CC nodes to validate the file names based on a sequence number.

When file-level sequencing is enabled, the CC node performs the following validations on each CDR input file:

- Checks whether the file name includes a sequence number.
- Checks whether the sequence number is in order.

If the file name passes both validations, the CC node processes the input file and sets *seqNum* in memory and in the *OMC\_home/ocomc/scratch/node\_id/file\_seq\_order\_info* file to the next sequence number. For example, if the CDR input file has a sequence number of 12, the CC node sets *seqNum* to 13.

If the file name fails one or both validations, the CC node adds a warning message to the node log file and displays an alarm before processing the input file.

 **Note:**

- You can use information in the node log file to identify the out-of-sequence file names.
- If the CC node stops while processing CDR input files, Offline Mediation Controller uses **file\_seq\_order\_info** to resume validations when the CC node is restarted.

For example, this shows how the CC node would process CDR input files received in the following order:

1. **ABC\_10.txt**: The CC node changes *seqNum* to 11 and processes the input file.
2. **ABC\_11.txt**: The CC node changes *seqNum* to 12 and processes the input file.
3. **ABC\_15.txt**: The CC node logs the out-of-sequence file name in the node log file, displays an alarm, processes the input file, and changes *seqNum* to 16.
4. **ABC\_5.txt**: The CC node logs the out-of-sequence file name in the node log file, displays an alarm, processes the input file, but does not change *seqNum*.

To use file-level sequencing, configure your file-based CC node as follows:

1. Enable sequence ordering in the CC node.

In the CC node's Node Configuration dialog box, select the **Advanced** tab and then select the **Sequence Ordering by File Name** option.

2. (Optional) Enable multi-threading in the CC node to improve performance. See "[Improving CC Node Processing Performance with Multi-Threaded Processing](#)".
3. Ensure that your CDR input files adhere to the file name syntax required for sequencing. Your files can use one of the following:

- The default file name syntax for sequencing:

```
sourceFilename[_seqNum].fileExtension
```

where:

- *sourceFilename* is the name of the original CDR input file.
  - *seqNum* is the sequence number of the CDR input file. This should contain a fixed number of digits.
  - *fileExtension* is the extension of the original CDR input file.
- A custom file name syntax for sequencing. See "[Customizing the Sequencing File Name Syntax](#)".

## Customizing the Sequencing File Name Syntax

To customize the file name syntax for sequencing:

1. Stop the CC node. See "Stopping Nodes" in *Offline Mediation Controller System Administrator's Guide*.
2. Open the `OMC_home/config/nodeID/general.cfg` file in a text editor.



3. Set the **fileNamePattern** entry to the syntax of your CDR input file names (without the file name extension).

```
fileNamePattern 'X_SEQ.X'
```

where:

- **X** represents a group of alpha-numeric characters. The group cannot contain symbols or special characters.
  - **SEQ** is the place holder for the sequence number.
  - **.** is a dot that appears between a group of alpha-numeric characters.
  - **\_** is an underscore that appears between a group of alpha-numeric characters.
4. Save and close the **general.cfg** file.
  5. Start the CC node. See "Starting Nodes" in *Offline Mediation Controller System Administrator's Guide*.

### Sample File Name Syntax

The following shows a group of sample input file names with the sequence number shown in bold:

```
APSCDR_M_USERBILLING.2030.02.20.1031.STD.000036.1.rcd  
APSCDR_M_USERBILLING.2030.02.20.1032.STD.000037.1.rcd  
APSCDR_M_USERBILLING.2030.02.20.1033.STD.000038.1.rcd
```

In this case, you would configure the **fileNamePattern** parameter as follows:

```
fileNamePattern 'X_X_X.X.X.X.X.X.SEQ.X'
```

### Sample File Name Syntax with Sequence Number Last

The following shows a group of sample input file names with the sequence number shown in bold:

```
BW-CDR-20300604093000-2-00505691D073-000503.csv  
BW-CDR-20300604093000-2-00505691D073-000504.csv  
BW-CDR-20300604093000-2-00505691D073-000505.csv
```

In this case, you would configure the **fileNamePattern** parameter as follows:

```
fileNamePattern 'X-X-X-X-X.X.SEQ'
```

## Managing Duplicate Records

Offline Mediation Controller includes the following methods of detecting and removing duplicate records:

- You can remove duplicate CDR files. See "[About Removing Duplicate CDR Files](#)".
- You can remove duplicate records. See "[About Removing Duplicate Records](#)".

## About Removing Duplicate CDR Files

By default, the file-based CC nodes do not validate the input files based on the file names. The CC nodes process the input files based on the alphabetical order of the file name. You can configure a CC node to identify duplicate files based on the file name.

The CC node compares the names of the incoming files to the names in memory. If the file name does not exist in memory, the file is processed. If the file name does exist in memory, the CC node renames the file to *sourceFilename.fileExtension.duplicate*, logs a warning in the node log, and displays an alarm.

The CC node reads the CDR input file names and stores the file names in memory and in the *OMC\_home/ocomc/scratch/node\_id/file\_duplicate\_info* file for a specified length of time. *node\_id* is the unique ID assigned to the node when the node configuration is saved.

The CC node compares the names of the incoming files to the names in memory. If the file name does not exist in memory, the file is processed. If the file name does exist in memory, the CC node renames the file to *sourceFilename.fileExtension.duplicate*, logs a warning in the node log, and displays an alarm.

If the CC node stops while processing CDR input files, Offline Mediation Controller uses the information in the **file\_duplicate\_info** files to resume validations when the CC node is restarted.

## About Removing Duplicate Records

Use the duplicate check EP node to find duplicate records while processing CDRs. This prevents you from charging a customer twice for the same usage.

To configure the duplicate check EP node, you define a duplicate check key. A duplicate check key is a set of fields in the CDR that uniquely identify the record and the record timestamp. When you run the EP node, the key is added to a memory partition, and assigned an amount of time for which it is used.

When CDRs are processed, the duplicate check node compares the keys in the records to the keys in the partition. If the key already exists in the partition, the record is rejected and is written to a file in the duplicate record storage directory, and the information related to the duplicate records is written to a log file.

If a duplicate key is not found, the key is added to the list of keys in the partition and the duplicate check file, and the CDR is distributed to the next node in the mediation node chain.

When you configure the duplicate check EP node, the configuration includes:

- Editing the EP node rule file to specify the data used for the duplicate check key. For example:

```
DUP_CHECK_KEYS "calling_number session_id seq_no";
DUP_CHECK_EVENT_TIME_FIELD "start_time";
```
- Configuring the EP node to specify parameters such as how often to create a partition to store duplicate check keys (daily or monthly), how often to flush records, where to store duplicate records files, and so on.

In addition to the duplicate check EP node, you can also detect duplicate records by configuring some sequencing EP nodes, for example:

- Sequencing EP node
- 3GPP Session Sequencing EP node
- Hot Billing Duplicate CDR Removal EP node

Oracle recommends using the duplicate check EP node.

## About Aggregation

Aggregation can be used for a variety of purposes. The primary function is to aggregate multiple records for a single session into one record.

When you configure aggregation, you can specify to aggregate data based on the following:

- **Time.** You can specify to aggregate records based on time. There are two options:
  - Aggregate records based on the first record received. For example, if the time is set for 10 minutes, records are aggregated for 10 minutes and then aggregated. Following that, records are aggregated for another 10 minutes and so on.
  - Aggregate records based on the time since the last record was received. For example, if the time is set for 10 minutes, and ten minutes pass before no record is processed, the records are aggregated.

- **Volume.** For example, you could create a record when the volume reaches 1 MB.

When using volume-based aggregation, the AP node may create NARs for a partial session. That is, if the session contains two CDRs of 6 MB and 2 MB respectively, and the maximum volume is set to 1 MB, Offline Mediation Controller creates two NARs for the session instead of one NAR per session.

- **Time and volume.** If you specify to aggregate based on time *and* volume, the records are aggregated based on which value is reached first, time or volume.

## About Time-Based Aggregation of CDRs

Offline Mediation Controller supports aggregating the CDRs based on the arrival time of the CDRs before the AP node writes the aggregated CDRs as one or more NARs to a NAR archive file.

If you set the option for session segmentation by the arrival time of the first CDR, the CDRs are aggregated in memory from the arrival time of the first CDR until the time configured in the **Flush Time** field (or until the arrival of a termination CDR). When the flush time is reached, the AP node writes the aggregated CDRs as a NAR to a NAR archive file. With this option, if the flush time occurs before the arrival of the termination CDR, the CDRs are written as multiple NARs to a NAR archive file.

For example, if the segmentation of the aggregated CDRs is time-based set by the arrival of the first CDR, the flush time is set to 600 (10 minutes), and the flow of CDRs is as follows:

1. The first CDR for session A arrives at 10:10 am
2. The second CDR for session A arrives at 10:15 am
3. The third CDR for session A arrives at 10:17 am
4. The fourth CDR for session A arrives at 10:21 am

The first three CDRs are aggregated and stored in the memory until 10:20 am (time of arrival of the first CDR + flush time). At 10:20 am, the AP node writes the aggregated CDRs as a NAR to a NAR archive file. The fourth CDR is aggregated and stored in the memory in a new aggregation session.

If you set the option for session segmentation by the arrival time of the last CDR, the CDRs are aggregated in memory from the arrival time of the last CDR until the time configured in the **Flush Time** field (or until the arrival of a termination CDR). When the flush time is reached, the AP node writes the aggregated CDRs as a NAR to a NAR archive file. If a new CDR arrives before the flush time is reached, the flush time is reset to begin from the arrival time of the new CDR. With this option, the CDRs for long-duration sessions are aggregated into a single NAR unless the flush time is reached before the arrival of the termination CDR.

For example, if the segmentation of the aggregated CDRs is time-based set by the arrival of the last CDR, the flush time set to 600 (10 minutes), and the flow of CDRs is as follows:

1. The first CDR for session A arrives at 10:10 am
2. The second CDR for session A arrives at 10:15 am
3. The third CDR for session A arrives at 10:17 am

If no more CDRs arrive, the three CDRs are aggregated and stored in memory until 10:27 am (time of arrival of the last CDR + flush time). At 10:27 am, the AP node writes the aggregated CDRs as a NAR to a NAR archive file.

## About Volume-Based Aggregation of CDRs

Offline Mediation Controller supports aggregating the CDRs based on volume. When the volume of the aggregated CDRs reaches the configured maximum volume, the AP node writes the aggregated CDRs as a NAR to a NAR archive file.

For example, if the segmentation of the aggregated CDRs is volume-based, the data volume is a maximum of 1 MB, and the flow of CDRs is as follows:

1. The first CDR for session A has the value of 300 KB in the CDR's **dataVolume** field
2. The second CDR for session A has the value of 700 KB in the CDR's **dataVolume** field
3. The third CDR for session A has the value of 100 KB in the CDR's **dataVolume** field

The maximum volume limit is exceeded when the third CDR arrives and the AP node writes the aggregated CDRs as a NAR to a NAR archive file.

### Note:

When using volume-based aggregation, the AP node may create NARs for a partial session. That is, if the session contains two CDRs of 6 MB and 2 MB respectively, and the maximum volume is set to 1 MB, Offline Mediation Controller creates two NARs for the session instead of one NAR per session.

## About Mapping and Enrichment

Charging often requires that you change or add data to records before they are rated. To do so, use EP nodes. See the following topics:

- [Record Enhancement for BRM Charging](#)
- [Restoring Excluded Records](#)
- [Adding Data to Records](#)

## Record Enhancement for BRM Charging

Use the record enhancement charging EP node to prepare records for charging by BRM. This node configures the following:

- Social number
- Prefix description
- Service code
- Usage class
- Access point name (APN)

To configure the record enhancement charging EP node, define the mapping and enrichment values in the rule file.

Before you create the record enhancement charging EP node, configure the database connection information to Oracle Communication Billing and Revenue Management (BRM) database. The record enhancement charging EP node uses the database connection information to retrieve the mapping information while enhancing the CDR data.

To configure the database connection:

1. Open the `OMC_home/config/nodemgr/prc.properties` file in a text editor, where `OMC_home` is the directory in which Offline Mediation Controller is installed.
2. Add the following text:

```
driver=oracle.jdbc.driver.OracleDriver
url= jdbc:oracle:thin:@server_name:port_number:database_alias
username=login
password=password
```

where:

- `server_name` is the system on which the BRM database is installed.
  - `port_number` is the port number of the system running the BRM database.
  - `database_alias` is the BRM database alias of the schema you are connecting.
  - `login` is the user name for the database schema you are connecting.
  - `password` is the encoded password for `login`.
3. Save and close the file. You now need to encode the database connection password.
  4. Go to the `OMC_home/bin` directory.
  5. Run the following command:

```
./encode password
```

where *password* is the password for the user name for the database schema you are connecting.

The command window displays *password* in encoded form. For example:

```
72,-46,62,71,41,-115,14,-68,-34,-4,-105,-113,-125,1,-18,-70
```

## Filtering Records Based on Data

The Record Filter EP node deletes records according to multiple selection criteria. Filtering is based on:

- CDR Type: G-CDR, S-CDR, M-CDR, S-SMO-CDR, S-SMT-CDR
- List of APNs
- List of SMS-Cs

The EP node also suppresses certain optional fields if they are not required.

## Restoring Excluded Records

Use the Partial CDR Completion AP node to restore specific fields to records. Even though this is an AP node, it does not aggregate any records.

The rule file specifies to restore eight optional fields to the record, which are normally excluded. The Nortel SGSN generates S-CDRs that exclude eight optional fields in non-initial partial records. These eight fields are excluded because their values do not change throughout the PDP session. If you use this rule file, the node adds the eight fields to the non-initial S-CDRs, using the values from the initial S-CDR. The eight optional fields are:

- MS-Network-Capability
- Charging-Characteristics
- Served-MSISDN
- APN-Selection-Mode
- Node-ID
- Network-Initiation
- QosNegotiated
- QosRequested

## Adding Data to Records

You can add data to records by using the following EP nodes:

- Use the Record Processing EP node to enhance records with the user-defined rules set in the rule file.
- Use the Record Enhancement (Local File) EP node to enhance records with data from a local file.
- Use the Record Enhancement (Remote File) EP node to enhance records with data from a remote file.

## Routing Records Based on Data

The Flexible Routing EP node provides the ability to route records based on field values. For example, the node can route records based on the value in the Charging Characteristics field and send pre-paid subscribers to one DC node and post-paid subscribers to another DC node. The node can also route records based on the value in the SGSN IP Address field and send home subscribers to one DC node and outbound roamers to another DC node.

## File and Record Processing Options

See the following topics:

- [Processing CDRs at File Level](#)
- [Improving EP Performance by Using Multi-Threaded Processing](#)
- [Improving CC Node Processing Performance with Multi-Threaded Processing](#)
- [Improving Record Archive File Performance with Bulk Read and Write Processing](#)

## Processing CDRs at File Level

By default, Offline Mediation Controller processes the call detail record (CDR) input file as individual records. Offline Mediation Controller can also process the CDR input file as a single unit. This enables Offline Mediation Controller to reject the original CDR input file at any stage of processing by the nodes in the node chain.

When processing individual records, the CC node processes the CDR input file, and each CDR in the CDR input file is converted to a NAR. In the CC node, any run-time errors during the CDR data processing are handled through the NPL rule file, and the CDR input file is renamed to *sourceFilename.error*. The records that are already processed by the CC node and sent to the next node cannot be rolled back.

You can configure file-level transaction for nodes, only if:

- Every node in the node chain is configure to use file-level transaction.
- The node chains contain a single flow without any splits.

When processing CDR input files, Offline Mediation Controller performs the following tasks:

- Generates a transaction ID and associates the ID with the input file.
- Does one of the following:
  - If the transaction is successful, completes the transaction and renames the original CDR input file to *sourceFilename.processedFileSuffix*.  
where:
    - \* *sourceFilename* is the name of the original CDR input file.
    - \* *processedFileSuffix* is the file pattern suffix configured in the CC node to rename the CDR input file post processing. The default is **.done**.
  - If the transaction is rejected, renames the CDR input file to *sourceFilename.reject* and moves the file to the *OMC\_homeIreject* directory. These rejected files are made available for suspense handling.

- Provides statistics related to the file-level transaction. The statistics are displayed in the **Transaction Details** table in the Node Performance screen.

## Configuring the File-Level Transaction Threshold

When processing input files, you can configure the file-level transaction threshold in the node manager configuration file. The file-level transaction threshold determines what percentage of records in a file can be suspended before the entire input file is rejected. This is an optional parameter. The default is 100%.

For example, if you have configured the file-level threshold to be at 40%, if a file contains 10 records, the file will be rejected when the fourth error record is encountered.

To configure the file-level transaction threshold:

1. Stop the node manager.
2. Open the `OMC_home/bin/nodemgr.cfg` file in a text editor.
3. Add the following entry to the file:

```
FILE_LEVEL_TRANSACTION_THRESHOLD 'value'
```

where *value* specifies the threshold percentage.

For example:

```
FILE_LEVEL_TRANSACTION_THRESHOLD '50'
```

4. Save and close the file.
5. Start the node manager.

## Viewing the File-Level Transaction Performance

To view the file-level transaction performance:

1. Log on to Offline Mediation Controller Administration client.  
The Node Hosts & Nodes (logical view) screen appears.
2. From the **Administrative Function** list, select **Node Performance**.
3. In the **Mediation Hosts** table, select the mediation host that you want to monitor.

The **Transaction Details** table displays the file-level transactions currently running under the selected mediation host.

## Improving EP Performance by Using Multi-Threaded Processing

By default, an EP node processes records in sequential order, using a single processing thread. When you use a single-threaded EP node to enrich the data records with a multi-threaded collection cartridge (CC) node and a multi-threaded distribution cartridge (DC) node, the records processing is slower. When multi-threading is enabled for Record Processing EP nodes, multiple files are processed in parallel, reducing the processing time.



## Improving CC Node Processing Performance with Multi-Threaded Processing

By default, file-based CC node and file-based DC node input files are processed in sequential order, using one processing thread. When multi-threading is enabled for file-based CC and DC nodes, multiple input files are processed in parallel, reducing the processing time.

When multi-threading is enabled, the order is not maintained. If you require your file-based CC nodes and file-based DC nodes to output files in the order that the files are read and processed, even when multi-threading is enabled, select the **Enable Ordering** check box.

If **Multi Threaded** and **Enable Ordering** are enabled, the maximum number of records set in the **Max Records Per File** field in the **File Output** node tab is disregarded. Files retain the original number of records received by the ASCII DC node.

## Improving Record Archive File Performance with Bulk Read and Write Processing

You can configure to cache the bulk processing of network accounting record (NAR) data. By default, each intermediate NAR archive file is created and read line by line by each processing node. When the **Enable bulk read/write** check box is selected in a node configuration, each NAR archive file is read and stored in memory for processing by the respective node, and the NAR output files are written into memory before writing to the cache output file, reducing the processing time.

The bulk read and write function:

- Reads and stores each intermediate NAR archive file as a whole into memory for processing by the respective node.
- Writes the NAR output file as a whole into memory, until either the maximum number of NARs for a file is reached or the idle write time has expired, before writing the output data to a cache file.

Bulk read and write operations process and pass data records using memory, reducing the processing time.



### Note:

If you have system memory constraints, use the **Enable bulk read/write** function only for your critical path node chains.

## Discarding Records

You can use the following EP nodes to discard records:

- Use the Discarding EP node to discard records. You can discard records based on a value or by date.
- Use the Record Filter EP node to delete records according to multiple selection criteria. Filtering is based on:

- CDR Type: G-CDR, S-CDR, M-CDR, S-SMO-CDR, S-SMT-CDR.
- List of APNs
- List of SMS-Cs

The Record Filter EP node also suppresses certain optional fields if they are not required.

# 4

## Suspending and Recycling Records

You configure node chains to suspend and recycle records and batches of records in Oracle Communications Offline Mediation Controller.

Topics in this document:

- [About Suspending and Recycling Records and Batches](#)
- [About Suspense and Recycle Node Chains](#)
- [About Searchable Fields](#)
- [Configuring the File-Level Transaction Threshold](#)
- [About record Field Mapping](#)
- [Working with SQL Files to Restore a Recycled CDR](#)
- [Configuring Suspense Errors](#)

### About Suspending and Recycling Records and Batches

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

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

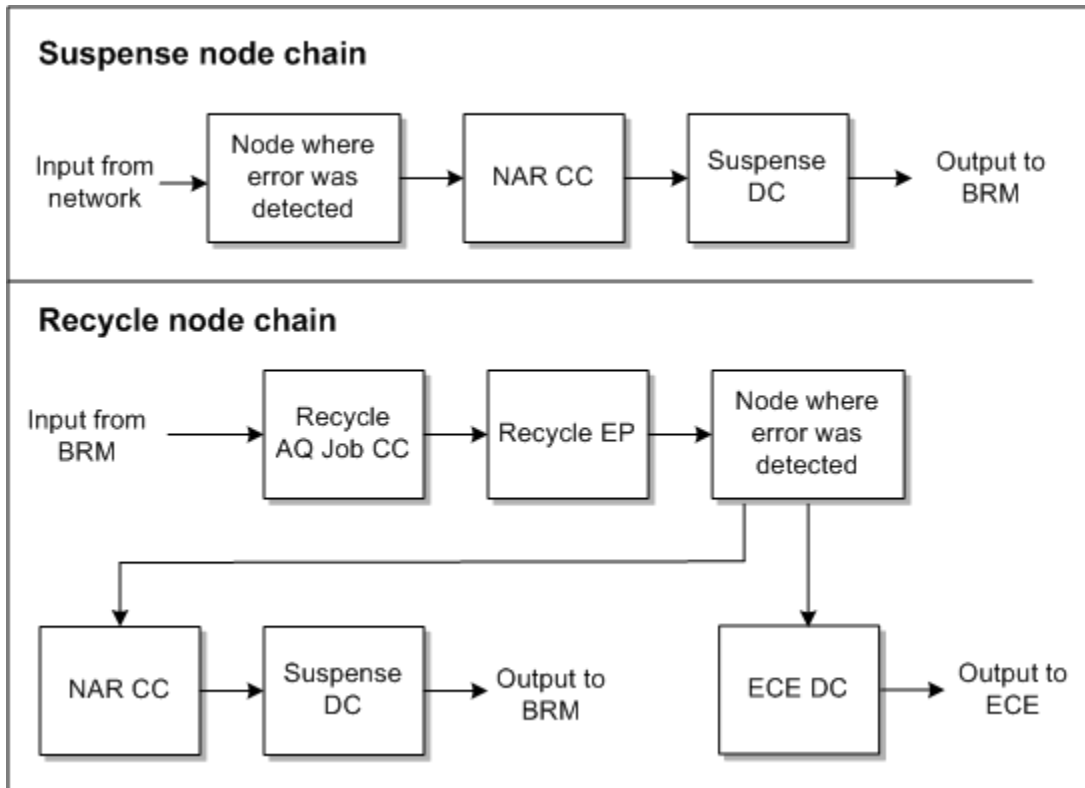
You can suspend and recycle individual records or a batch of records.

Failed records and batches are found during processing in Offline Mediation Controller. They are sent to the BRM server where you can analyze and correct the problems, and then send them back to Offline Mediation Controller to be processed again.

### About Suspense and Recycle Node Chains

To suspend and recycle records and batches in Offline Mediation Controller, you create a suspense node chain and a recycle node chain. [Figure 4-1](#) shows the suspense and recycle node chains.

Figure 4-1 Suspense and Recycle Node Chains



The suspense node chain does the following:

1. A node, such as an EP node, finds an error in a record or batch.
2. The node where the error was found sends the record or batch to the NAR CC node.
3. The NAR CC node reads the record or batch from the suspense directory and prepares the record or batch for further processing.
4. The NAR CC node sends the record or batch to the Suspense CD node.
5. The Suspense CD node sends the record or batch to BRM to be processed by BRM Suspense Manager. To do so, the Suspense CD node generates a Create file.

The recycle node chain does the following:

1. The Recycle Job AQ CC node receives a record or batch from BRM and prepares the record or batch for further processing. To identify the record or batch, the Recycle Job AQ CC node receives the job ID message from the Oracle AQ queue.
2. The Recycle Job AQ CC node sends the record or batch to the Recycle EP node.
3. The Recycle EP node uses the job ID message to send the record or batch to the node that reported the original error.
4. If no errors are found, the record or batch is sent to two nodes:
  - The next node in the node chain. In [Figure 4-1](#) this is the ECE DC node.

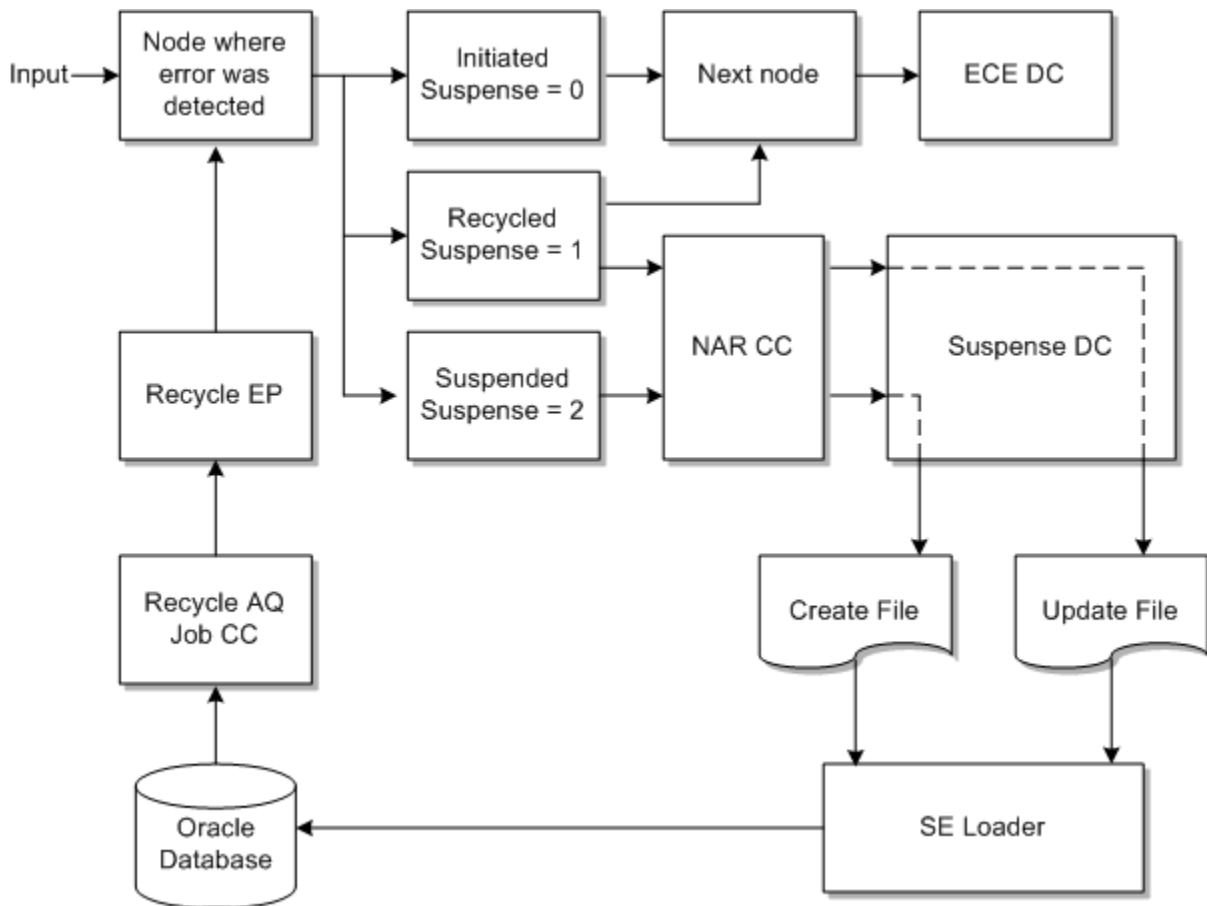
- The NAR CC node, which sends the record or batch to the Suspend DC node. The Suspend DC node sends the output to BRM to record that the record or batch was processed successfully. To do so, the Suspend DC node generates an Update file.

To manage suspending and recycling records and batches, configure nodes to add and update a flag that tracks the status of suspended and recycled records and batches. The status values are:

- **Initialized (0).** This flag is set when the node processes the record or batch. If no errors are found, the record or batch is passed to the next node in the node chain. If the status remains at 0, the record or batch is never suspended.
- **Failed (2).** This flag is set when an error is found in the record or batch. The record or batch is sent to the NAR CC node, and then to the Suspend DC node, but not sent to the next node in the node chain. The Suspend DC node generates a Create file and sends it to BRM.
- **Successful (1).** This flag is set when a record or batch is recycled, and no new error is found by the node that originally found the error. The record or batch is sent to two places:
  - The next node in the node chain, such as the ECE DC node.
  - The NAR CC node, and then to the Suspend DC node. The Suspend DC node generates an Update file and sends it to BRM to update the record or batch status in BRM Suspend Manager.

Figure 4-2 shows how the suspend flag is used in node chains.

Figure 4-2 Flags Used in Suspense and Recycle Node Chains



## About Searchable Fields

When you configure Suspense Manager in BRM, you can specify which record and batch fields can be searched on and displayed in Suspense Management Center. These fields are called *searchable fields*. For more information, see "Using Custom Data With Suspense Manager" in *BRM Suspending and Recycling Event Records*.

To configure searchable fields in Offline Mediation Controller, you edit the Suspense DC node rule file. You specify the fields by using the BRM table names, for example:

```
QueryableTables "SUSP_USAGE_TELCO_INFO_T,SUSP_USAGE_TELCO_GSM_INFO_T";
```

For information about editing the Suspense DC node rule file, see *Administration Client Online Help*.

## Configuring the File-Level Transaction Threshold

The file-level transaction threshold determines what percentage of records in a file can be suspended before the entire input file is rejected. This is an optional parameter.

To configure the file-level transaction threshold:

1. Stop the node manager that you want to reconfigure.
2. Open the `OMC_home/bin/nodemgr.cfg` file in a text editor.
3. Add the following entry to the file:

```
FILE_LEVEL_TRANSACTION_THRESHOLD 'value'
```

where *value* specifies the threshold percentage.

For example:

```
FILE_LEVEL_TRANSACTION_THRESHOLD '50'
```

4. Save and close the file.
5. Start the node manager.

## About record Field Mapping



### Note:

Oracle recommends record field mapping if you use Suspense Manager.

In Offline Mediation Controller, the NPL rules for a node allow values from the fields in the input stream to be assigned to the fields in the output stream, which are sent to the downstream nodes or external medium for processing.

The Suspense DC node serializes the CDR based on the output field names from the NPL mapping. Shorter designations may be used in place of the output field names by using record field mapping, which reduces the length of field names by replacing the field name with an ID. Whatever mapping is defined in the Suspense DC node, the reverse is required for the Recycle EP node.

The **EDR Field Mapping Name** field in the **Mode** tab of the Suspense DC node, refers to the database `EDR_FIELD_MAPPING_T.NAME` and the record field mapping is stored in the `EDR_FLD_MAP_BUF_T` table. Both tables are installed during a BRM installation or when you run the **load\_edr\_field\_mapping** utility. Before using record field mapping, verify that these tables exist.

A record field mapping XML file is loaded into the database during installation of BRM Suspense Manager. You use the **load\_edr\_field\_mapping** utility, a BRM component, to add or modify existing record field names and load new versions of the record field mapping XML file.

For more information, see "Integrating Event Record Field Mapping" in *BRM Suspending and Recycling Event Records*.

## Working with SQL Files to Restore a Recycled CDR

The Recycle EP node restores the recycled CDRs based on SQL statements. Two types of SQL statements are used by the Recycle EP node:

- Detail SQL, which is required.
- Header SQL, which is optional.

The following sample SQL statement files are provided with the Recycle EP JAR file:

- To retrieve serialized suspended CDRs in ASCII format, use the **RecycleDetail.sql** file.
- To retrieve serialized suspended CDRs in records, use the **RecycleEdr.sql** file.
- To retrieve serialized suspended CDRs in IMS format, which includes selected record common fields, use the **RecycleIMS.sql** file.
- To retrieve header information, use the **RecycleHeader.sql** file, which is a generic SQL statement.

The data from Header SQL is set in every recycled CDR retrieved by the detail SQL statement.

## Installing the Sample Detail and Header SQL Files

To install the sample detail and header SQL files:

1. Go to the *OMC\_home/web/htdocs* directory, where *OMC\_home* is the directory in which Offline Mediation Controller is installed.
2. Extract the SQL files from the JAR file:

```
jar xvf Recycle.jar sql/RecycleDetail.sql sql/RecycleEdr.sql sql/
RecycleIMS.sql sql/RecycleHeader.sql
```

The SQL files are placed in a **sql** directory.

## Updating and Applying a Sample SQL File in the Recycle EP Node

To update and apply a sample SQL file in the Recycle EP node:

1. Choose one of the sample SQL statement files that best represents the format of the CDR before suspension, and update it according to your business requirements.

### Note:

The following variables may be used in the SQL statement:

- `${RECYCLE_JOB_ID}`, which is replaced by the job ID message sent by the AQ listener.
- `${PIPELINE_CATEGORY}`, which is replaced by the name entered into the **Category** field in the **SQL** configuration tab of the Recycle EP node. A NULL value is given if the **Category** field is left blank.

2. In the **SQL Header File** and **SQL Detail File** fields in the **SQL Files** tab of the Recycle EP node, enter the full path and file name to the SQL header and SQL detail files.

## SQL Statement Examples

The following example shows a detail SQL statement without editable fields:



```

SELECT  b.poid_id0, b.pipeline_name, r.recycle_mode, '', b.batch_id,
        b.suspended_from_batch_id, b.recycle_key, e.edr_buf
FROM    suspended_usage_t b,
        susp_usage_edr_buf e,
        susp_usage_recycle_t r
WHERE   b.poid_id0 = e.obj_id0
        AND b.recycle_obj_id0 = r.obj_id0
        AND b.recycle_obj_id0 = ${RECYCLE_JOB_ID}
        AND NVL(b.pipeline_category, 'NULL') = ${PIPELINE_CATEGORY}
ORDER BY b.poid_id0

```

The following example shows a detail SQL statement with editable fields:

```

SELECT  a.poid_id0, a.pipeline_name, a.recycle_mode, a.override_reasons,
        a.batch_id, a.suspended_from_batch_id, a.recycle_key, a.edr_buf,
        a.editable_flds
FROM    (
        SELECT b.poid_id0 , b.pipeline_name, r.recycle_mode,
               r.override_reasons, b.batch_id, b.suspended_from_batch_id,
               b.recycle_key, e.edr_buf,
               'CalledId='|| t.calling_from|| CHR (9)||
               'calling_number='||t.called_to|| CHR (9)||
               'duration='|| t.call_duration|| CHR (9)||
               'start_time='|| t.start_time|| CHR (9)||
               'product_type='|| t.usage_type|| CHR (9)||
               'requestedInputVolume='|| t.bytes_in|| CHR (9)||
               'requestedOutputVolume='|| t.bytes_out|| CHR (9)||
               'cell_id='|| g.cell_id as editable_flds
        FROM  suspended_usage_t b,
               susp_usage_telco_info_t t,
               susp_usage_telco_gsm_info_t g,
               susp_usage_edr_buf e,
               susp_usage_recycle_t r
        WHERE b.poid_id0 = t.obj_id0
               AND b.poid_id0 = e.obj_id0
               AND b.recycle_obj_id0 = r.obj_id0
               AND b.poid_id0 = g.obj_id0
               AND b.recycle_obj_id0 = ${RECYCLE_JOB_ID}
               AND NVL(b.pipeline_category, 'NULL') = ${PIPELINE_CATEGORY}) a
ORDER BY a.poid_id0

```

## Configuring Suspense Errors

You can configure the following:

- Which errors cause a record or batch to be recycled. You configure this in the nodes where errors occur.
- The reasons and subreasons that are displayed in BRM Suspense Management Center for each of the errors. You configure this in BRM reason files. See "Changing the List of Suspense Reasons and Subreasons" in *BRM Suspending and Recycling Event Records*.

To configure a node to suspend a record or batch, you edit the node's rule file. You need to use Java hooks. This example shows the Java hook code for identifying an error:

```

if (in.product_type != "TEL" && in.product_type!="SMS") {
    // bad product type detected
    // 1) prepare output stream
    // 2) set suspended routing flag
    logInfo("inside wrong product type in.product_type="
            "+in.product_type);
}

```

```
suspHandler.assemble(in, out, 5002, "ProductCheck",  
"ASCII");
```

In this example:

- An error is recognized if the product is not a telco or SMS product.
- The suspense flag is set to 2 (Suspended).
- The error code is 5002.

The Offline Mediation Controller error codes are stored in the *OMC\_home/config/nodemgr/ocomc\_errorcodes.xml* file.

For more information, see "[Suspense Error Configuration Examples](#)".

## Configuring New Error Codes

You incorporate error codes for errors that you want to detect into your node. Error codes are mapped to suspense reasons and subreasons in the **Create** and **Update** output files by the Suspense DC node. You can search for suspended event CDRs and batch CDR files using the error, reason, and subreason codes in Suspense Management Center.

To configure new Offline Mediation Controller error codes for suspended CDRs:

1. Stop the node manager.
2. Go to the *OMC\_home/config/nodemgr* directory, where *OMC\_home* is the directory in which Offline Mediation Controller is installed.
3. Open the **ocomc\_errorcodes.xml** file in a text editor.
4. Add suspended error codes according to your specific business requirements by using the following XML format:

```
<OCOMCErrorCodes>  
  <error id="1">  
    <code>60001</code>  
    <message>CANCEL_NOT_ALLOWED</message>  
    <ReasonCode>13</ReasonCode>  
    <SubReasonCode>0</SubReasonCode>  
  </error>  
</OCOMCErrorCodes>
```

5. Save and close the file.
6. Restart the node manager.

 **Note:**

Any new Offline Mediation Controller error codes must be:

1. Added to the *BRM\_home/sys/data/config/pin\_suspend\_reason\_code* and *BRM\_home/sys/msgs/suspend\_reason\_code/suspend\_reason\_code.en\_US* files.
2. Loaded into the BRM database using the **load\_pin\_suspend\_reason\_code** and **load\_localized\_strings** utilities, which are components of BRM.

For more information, see "Changing the List of Suspend Reasons and Subreasons" in *BRM Suspending and Recycling Event Records*.

## Working with the Offline Mediation Controller Error Code API

The **OCOMCErrorCode** Java API is used to access the errors defined in the **ocomc\_errorcodes.xml** file. The following methods can be used:

- **OCOMCErrorCode.get(String errorMsg)**, this method will return the error code associated with the error message.
- **public static synchronized OCOMCError getError(String errorMessage)**, this method will return an instance of the **OCOMCError** object, which contains all the fields associated with the error.

 **Note:**

You do not need to create an instance of the **OCOMCErrorCode** class. The node manager will add the class when it has been restarted.

# 5

## Sending Records to Elastic Charging Engine

Learn how to configure Offline Mediation Controller to send records to Oracle Communications Billing and Revenue Management (BRM) Elastic Charging Engine (ECE).

Topics in this document:

- [About the ECE Cartridge Pack](#)
- [Rule File](#)
- [Mapping the Input Record Attribute to the Output Record Attribute](#)

### About the ECE Cartridge Pack

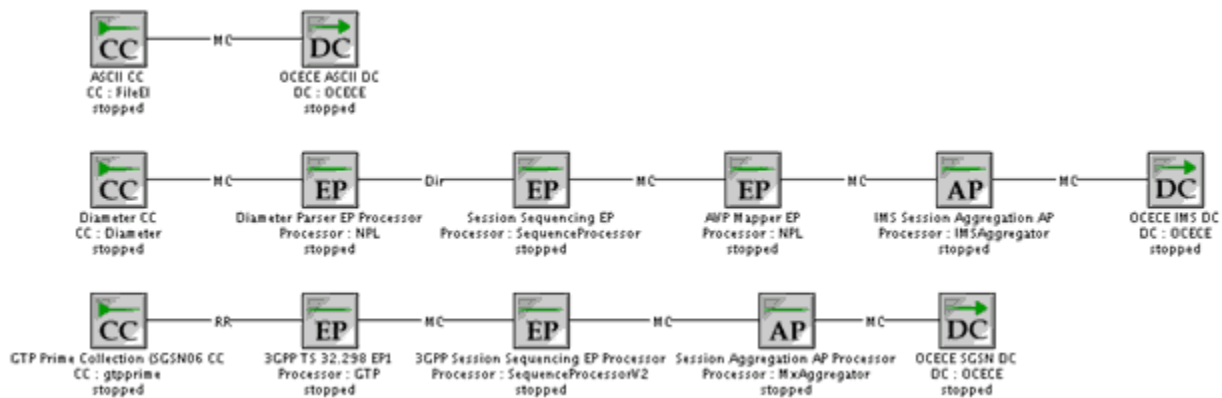
The ECE cartridge pack includes the ECE DC node, which enables Offline Mediation Controller to submit offline charging requests to ECE.

The ECE cartridge pack is used with the following existing cartridge packs to process the input records:

- Simple ASCII
- IMS
- SGSN

Figure 5-1 shows the node chain configuration for submitting offline charging requests to ECE.

Figure 5-1 ECE DC Node Configuration Architecture



### Rule File

When creating and configuring the ECE Distribution Cartridge (DC) node, you configure an NPL rule file containing the mapping data from the network accounting record (NAR) format

to an output format for use by ECE. For more information about NPL, see *Offline Mediation Controller Cartridge Development Kit NPL Reference Guide*.

When configuring the ECE DC node, define the following in the NPL rule files:

- [Configuration Block](#)
- [Input Record Block](#)
- [Output Record Block](#)
- [Expose Block for PAYLOAD](#)
- [Expose Block for Usage Object](#)
- [Expose Block for Payload Data Type](#)

## Configuration Block

Use a configuration block to set configuration values required by the node. You can define only one configuration block in an NPL rule file.

### Syntax

```
Config {
  PRODUCTTYPES "product_type[, ...]";
  PAYLOAD "payload_type[, ...]";
  PAYLOADDATATYPE "datatype";
  USAGEOBJECTS "usage_object";
  EVENT "event_type[, ...]";
  VERSION "version_number[, ...]";
  MODE "usage_request_mode";
  TIME_ZONE "time_zone";
  DATE_FORMAT "date_format";
  USAGE_REQUEST_BUILDER "request_format";
}
```

### Configuration Settings

**PRODUCTTYPES** "product\_type[, ...]";

where *product\_type* is a product type defined in ECE. This parameter is specific to the product types the ECE DC node processes.

**PAYLOAD** "payload\_type[, ...]";

where *payload\_type* is the payload type defined in ECE.

**PAYLOADDATATYPE** "datatype";

where *datatype* specifies the data type used in payloads.

**USAGEOBJECTS** "usage\_object";

where *usage\_object* specifies the usage objects type. The default value is **USAGEBUILDER**.

**EVENT** "event\_type[, ...]";

where *event\_type* specifies the types of events. Add the *event\_type* values separated by commas.

**VERSION** "version\_number[, ...]";

where *version\_number* specifies the version for each record corresponding to the product type. Add the *version\_number* values separated by commas.

**MODE** "*usage\_request\_mode*";

where *usage\_request\_mode* specifies the mode in which Offline Mediation Controller sends usage requests to ECE. Set the parameter to one of the following values:

- **CUMULATIVE**: Specifies that the balance impact in ECE is cumulative.
- **INCREMENTAL**: Specifies that the balance impact in ECE is incremental.

**TIME\_ZONE** "*time\_zone*";

where *time\_zone* specifies the time zone used by the ECE DC node to send the session start time and the session end time to ECE. The default value is **UTC**. The time zone is set at the record level or at the cartridge level:

- **Record level**: If the NPL output record block contains the **timeZone** field, the time zone is set at record level. For record level time zone, each call detail record (CDR) can have a different time zone defined by the **timeZone** field in the CDR.
- **Cartridge level**: If the configuration block contains **TIME\_ZONE** field, the time zone is set at cartridge level. All the CDRs processed by the cartridge will contain the same time zone as defined by the **TIME\_ZONE** field.



**Note:**

If the time zone is defined at record level and at cartridge level, the record level time zone is used.

**DATE\_FORMAT** "*date\_format*";

where *date\_format* specifies the date format used by ECE DC node to send the session start time and the session end time to ECE. The default value is **yyyy-MM-dd HH:mm:ss**.

**USAGE\_REQUEST\_BUILDER** "*request\_format*";

where *request\_format* defines the usage request builders. This parameter is used to create the builders for the corresponding product type, event, and version.

For example:

```
Config {
  PRODUCTTYPES "VOICE,DATA,SMS";
  PAYLOAD "VOICE_USAGE_Terminate_PAYLOAD,VOICE_USAGE_Update_PAYLOAD,
  VOICE_USAGE_Debit_unit_PAYLOAD,VOICE_USAGE_Refund_unit_PAYLOAD,
  VOICE_USAGE_Refund_amount_PAYLOAD,VOICE_USAGE_Debit_amount_PAYLOAD,
  DATA_DATA_USAGE_Terminate_PAYLOAD,DATA_DATA_USAGE_Update_PAYLOAD,
  DATA_DATA_USAGE_Debit_unit_PAYLOAD,DATA_DATA_USAGE_Refund_unit_PAYLOAD,
  DATA_DATA_USAGE_Refund_amount_PAYLOAD,DATA_USAGE_Debit_amount_PAYLOAD,
  SMS_SMS_USAGE_Terminate_PAYLOAD,SMS_SMS_USAGE_Debit_unit_PAYLOAD,
  SMS_SMS_USAGE_Refund_unit_PAYLOAD,SMS_SMS_USAGE_Refund_amount_PAYLOAD,
  SMS_SMS_USAGE_Debit_amount_PAYLOAD";
  PAYLOADDATATYPE "PAYLOADDATATYPE";
  USAGEOBJECTS "USAGEBUILDER";
  EVENT "USAGE,DATA_USAGE,SMS_USAGE";
  VERSION "1.0";
  MODE "CUMULATIVE";
```

```

TIME_ZONE "UTC";
DATE_FORMAT "yyyy-MM-dd HH:mm:ss";
USAGE_REQUEST_BUILDER "VOICE@USAGE@1.0,DATA@DATA_USAGE@1.0,SMS@SMS_USAGE@1.0";
}

```

## Input Record Block

Use an input record block to define which fields in the input data record are passed into the node.

### Syntax

```

InputRec {
    datatype parameter;
} in;

```

where:

- *datatype* is the data type of the input field, such as **String**.
- *parameter* is the input field name.

For example:

```

InputRec {
    String calling_number;
    Integer seq_no;
    Integer duration;
    String start_time;
    String product_type;
    String session_id;
    String CalledId;
    String end_time;
    String cell_id;
    String requestedInputVolume;
    String requestedOutputVolume;
    String requestedTotalVolume;
    String usedUnitsInputVolume;
    String usedUnitsOutputVolume;
    String usedUitsTotalVolume;
    String operationType;
    String correlation_identifier;
    String balance_element_id;
    String amount;
    // String timeZone;
} in;

```

## Output Record Block

Use an output record block to define the format of a node's output data record.

### Syntax

```

OutputRec {
    datatype parameter;
} out;

```

where:

- *datatype* is the data type of the output field, such as **String**.

- *parameter* is the output field name.

For example:

```
OutputRec {
    String calling_number;
    Integer seq_no;
    String duration;
    String start_time;
    String end_time;
    String product_type;
    String session_id;
    String CalledId;
    String npl_type;
    String cdr_service;
    String cell_id;
    String requestedInputVolume;
    String requestedOutputVolume;
    String requestedTotalVolume;
    String usedUnitsInputVolume;
    String usedUnitsOutputVolume;
    String usedUitsTotalVolume;
    Integer usedUnitsSpecificUnit;
    String eventType;
    String version;
    String operationType;
    String correlation_identifier;
    Integer balance_element_id;
    Double amount;
    // String timeZone;
} out;
```

## Expose Block for PAYLOAD

For every payload type in the configuration block, define an Expose block. In the Expose block, map the NPL fields to the ECE Payload fields defined in the ECE request specifications.

For example:

```
Expose for VOICE_PAYLOAD{out.CalledId "CALLED_ID";out.duration
"REQUESTED_UNITS[0].DURATION";}
Expose for DATA_PAYLOAD{out.cell_id "CELL_ID";
out.requestedInputVolume "REQUESTED_UNITS[0].INPUT_VOLUME";
out.requestedOutputVolume "REQUESTED_UNITS[0].OUTPUT_VOLUME";
out.requestedTotalVolume "REQUESTED_UNITS[0].TOTAL_VOLUME";
out.usedUnitsInputVolume "USED_UNITS[0].INPUT_VOLUME";
out.usedUnitsOutputVolume "USED_UNITS[0].OUTPUT_VOLUME";
out.usedUitsTotalVolume "USED_UNITS[0].TOTAL_VOLUME";}
```

## Expose Block for Payload Data Type

For every payload data type in the configuration block, define an Expose block. In the Expose block, define the data type for the fields that are passed in payload data type.

For example:

```
Expose for PAYLOADDATATYPE{
    out.duration "SECONDS";
    out.requestedInputVolume "VOLUME";
```



```

out.requestedOutputVolume "VOLUME";
out.requestedTotalVolume "VOLUME";
out.usedUnitsInputVolume "VOLUME";
out.usedUnitsOutputVolume "VOLUME";
out.usedUitsTotalVolume "VOLUME";
}

```

## Expose Block for Usage Object

For every usage object type in the configuration block, define an Expose block. In the Expose block, map the NPL fields that are sent as parameters in the create usage objects method.

For example:

```

Expose for USAGEBUILDER{
  out.calling_number "PARAM1";
  out.session_id "PARAM2";
  out.start_time "PARAM3";
  out.end_time "PARAM4";
  out.seq_no "PARAM5";
}

```

## Mapping the Input Record Attribute to the Output Record Attribute

When mapping the input record attribute to the output record attribute, set the following output fields based on the specific product type for which the CDR is received:

- **cdr\_service**: Set this field to the product types for each CDR record.
- **eventType**: Set this field to the event type for each CDR record corresponding to the product type.
- **version**: Set this field to the version for each CDR record corresponding to the product type.
- **operationType**: Set this field to the usage type for each CDR record corresponding to the product type.

See "Sample Mapping for ECE Cartridge Pack" in *Offline Mediation Controller Cartridge Development Kit NPL Reference Guide* for more information about the supported usage types.

For example:

```

out.cdr_service="VOICE"out.eventType="USAGE"out.version="1.0"out.operationType="Terminate"

```

# A

## Suspense Error Configuration Examples

This appendix provides two examples of how to configure Oracle Communications Offline Mediation Controller nodes to suspend records and batches.

Topics in this document:

- [About These Examples](#)
- [Product Type Error Event NPL Example](#)
- [Batch CDR File Error NPL Example](#)

For more information, see "[Suspending and Recycling Records](#)".

### About These Examples

The following examples describe the NPL rule files for cartridges that detect event and batch CDR errors. Both examples use Java hook methods. For more information about Java hooks, see "Java Hooks" in *Offline Mediation Controller Cartridge Development Kit Developer's Guide*.

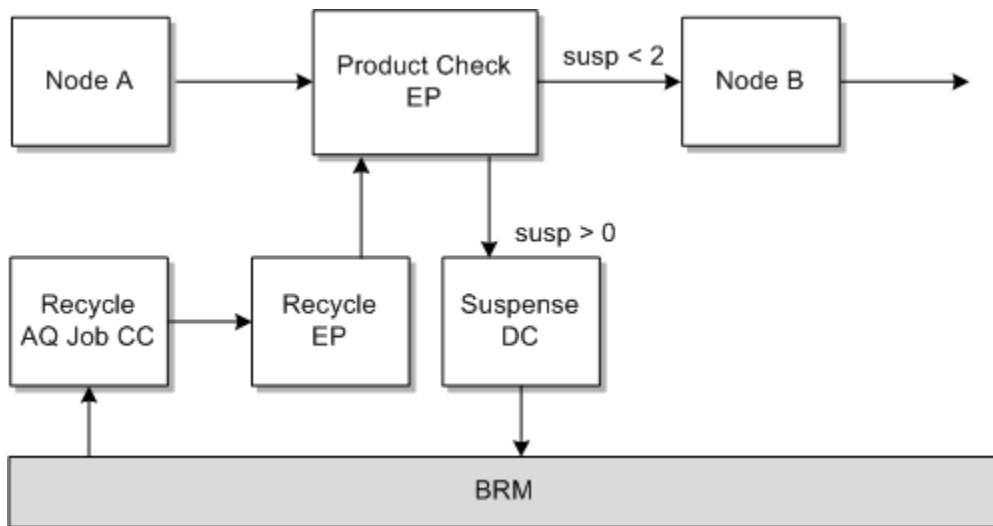
### Product Type Error Event NPL Example

This example describes an NPL file for a node that detects and suspends event CDRs if the product type is incorrect.

[Figure A-1](#) shows the processing flow of a CDR in a node chain that contains three nodes:

1. Node A sends its output stream to the Product Check EP node.
2. The Product Check EP node checks whether the product type is TEL or SMS.
3. If the Product Check EP node does not detect any errors, it sends its output stream to Node B.

**Figure A-1 Product Type Error Detection Flow**



In the Product Check EP node, configure the NPL file with a flag value that outputs the CDR to one of the following:

- If the product type is not valid, the CDR is made available for the Suspense DC node, which changes the status to a suspended state and generates a **Create** file for the SE Loader.
- If the product type is valid, the CDR outputs to Node B.
- If the product type is valid and the CDR is recycled, the CDR is made available for the Suspense DC node, which changes the status to a succeeded state and generates an **Update** file for SE Loader.

Table A-1 describes the NPL configuration for the Product Check EP node.

**Table A-1 Product Check EP NPL**

NPL	Description
<pre> JavaHook suspHandler=oracle.communications.brm.nm.nplhook.SuspenseMethodHandlerImpl;           </pre>	Java hook declaration.
<pre> InputRec {   String product_type;   String calling_number; } in;           </pre>	Input stream from Node A.
<pre> OutputRec {   String cdr_service;   String eventType;   String A_NUMBER;           </pre>	Output stream to Node B.

**Table A-1 (Cont.) Product Check EP NPL**

NPL	Description
<pre>// internal flag Integer pSUSPENDED; } out;</pre>	<p>The routing Flag.</p>
<pre>Expose for Routing {   out.pSUSPENDED "susp"; }</pre>	<p>Expose the routing flag and set the display name.</p>
<pre>// initialize routing flag to 0 out.pSUSPENDED=0;</pre>	<p>Initialize the routing flag to <b>0</b>, which if no errors are found sends the CDR to Node B.</p>
<pre>if (in.product_type != "TEL" &amp;&amp; in.product_type!="SMS") {   // bad product type detected   // 1) prepare output stream   // 2) set suspended routing flag   logInfo("inside wrong product type in.product_type=     "+in.product_type);   suspHandler.assemble(in, out, 5002, "ProductCheck",     "ASCII"); }</pre>	<p>If the product type is incorrect, call the <b>assemble</b> Java hook method to clone the input stream to the output stream and add suspense fields for the given error code of <b>5002</b>, node name <b>ProductCheck</b>, and node category <b>ASCII</b>.</p>
<pre>out.pSUSPENDED=2;</pre>	<p>Set the routing flag to <b>2</b>, which directs the suspended CDR to the Suspense DC for suspense handling.</p>
<pre>} else {   // good product type   logInfo("good product type in.product_type=     "+in.product_type);    // prepare output stream by assigning input values   if(in.product_type=="TEL"){     out.cdr_service="VOICE";     out.eventType="USAGE";   }else if(in.product_type=="SMS"){     out.cdr_service="SMS";     out.eventType="SMS_USAGE";   }   out.A_NUMBER=in.calling_number;</pre>	<p>If the product type is correct, prepare the output stream based on the input stream.</p>
<pre>// if this NAR was previously suspended (from   recycle) // 1) add suspense fields to output stream // 2) set suspended routing flag if (suspHandler.isRecycled(in)&gt;0) {   logInfo("key suspense fields found,   copying..." );   suspHandler.append(in, out);</pre>	<p>If the product type is correct and any suspense fields exist, such as, <b>SUSPENSE_ID</b>, append the suspense fields to the output stream.</p>

Table A-1 (Cont.) Product Check EP NPL

NPL	Description
<pre> out.pSUSPENDED=1; } </pre>	Set the routing flag to 1, which directs the CDR to: <ol style="list-style-type: none"> <li>Node B to continue processing.</li> <li>Suspense DC, which updates the status to Successful.</li> </ol>
<pre> } write(out); </pre>	Writes out the stream.

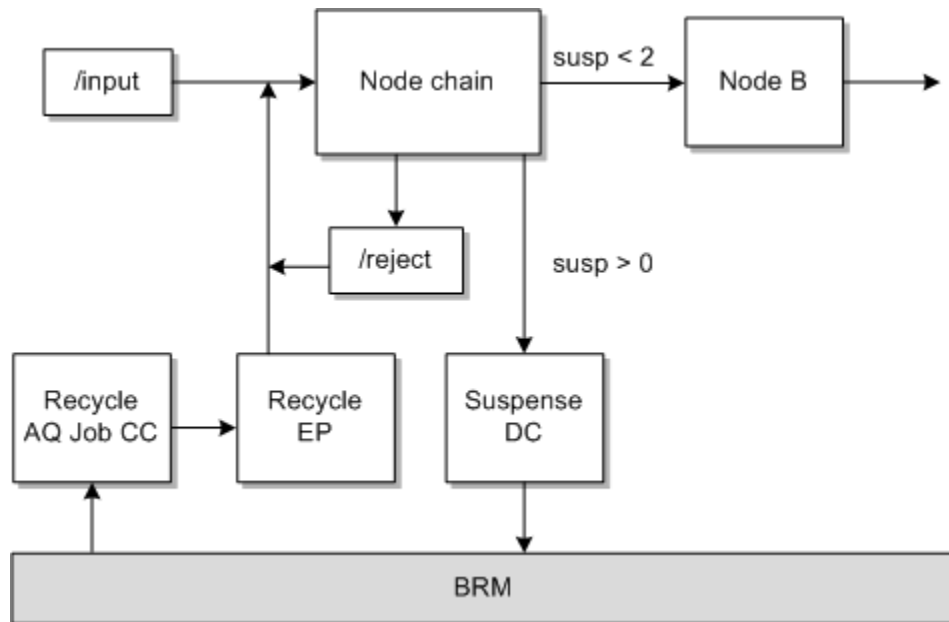
## Batch CDR File Error NPL Example

This example describes an NPL file for a node that detects and suspends a batch CDR file if it contains an error.

Figure A-2 shows the processing flow of a batch CDR file in a node chain that contains two nodes:

1. CDR CC node, which is responsible for processing input files received from the **input** directory.
2. Node B, which if no errors are detected, receives the input stream from the CDR CC node.

Figure A-2 Batch CDR File Error Detection Flow



In the node chain you configure the NPL with a flag value that outputs the CDR to one of the following:

- If no suspended batch CDR file information exists, the CDR CC node moves the batch CDR file to an **error** directory and makes information on the suspended batch CDR file available for the Suspense DC node. The Suspense DC node changes the status to a suspended state and generates a **Create** file for SB Loader.
- If the batch CDR file is valid, data processing continues to Node B.
- If suspended batch CDR file information exists, the Suspense DC node changes the status to a succeeded state and generates an **Update** file for SB Loader.

Table A-2 describes the NPL configuration for the nodes in the node chain.

**Table A-2 NPL Configuration**

NPL	Description
<pre>JavaHook suspHandler=oracle.communications.brm.nm.nplhook.SuspenseMethodHandlerImpl;</pre>	<p>Java hook declaration.</p>
<pre>InputRec {   String sourceFileName;   Byte rejected;   Byte resubmitted; } in;</pre>	<p>Input stream from the <b>input</b> directory. <b>Note:</b> Not all the fields in the input stream are declared.</p>
<pre>OutputRec {   Integer pSUSPENDED; } out;</pre>	<p>Declares the routing Flag. <b>Note:</b> Not all the fields in the output stream are declared.</p>
<pre>Expose for Routing {   out.pSUSPENDED "susp"; }</pre>	<p>Expose the routing flag and set the display name.</p>
<pre>/// local variables String pipelineName; String pipelineCategory; String sourceDir; String errorDir; String sourcePrefix; String sourceSuffix; String targetPrefix; String targetSuffix; Integer count;  pipelineName="CDR Pipeline"; pipelineCategory="Wireless"; sourceDir="/input"; errorDir="batch/error"; sourcePrefix="test_";sourceSuffix="edr"; targetPrefix="err_";targetSuffix="bad";</pre>	<p>Declare and initialize local variables.</p>
<pre>// initialize routing flag to 0 out.pSUSPENDED=0;</pre>	<p>Initialize the routing flag to <b>0</b>.</p>

**Table A-2 (Cont.) NPL Configuration**

NPL	Description
<pre> if (in.rejected &gt; 0) {   // file was rejected   logInfo("in rejected");   // 1) move files from input directory to error   directory   count=SuspHandler.move(sourceDir, sourcePrefix,     sourceSuffix, errorDir, targetPrefix,     targetSuffix);   // 2) create batch nar   SuspHandler.assemble(out,     sourcePrefix+in.sourceFileName+sourceSuffix   ,     pipelineName,     127,     errorDir,     targetPrefix+in.sourceFileName+targetSuffix   ,     "Acme Wireless",     "control10001",     "01",     "tap processing info",     pipelineCategory); </pre>	<p>If the batch CDR file is invalid, move the batch CDR file to an reject directory and call the <b>assemble</b> Java hook method to add the suspended batch file information to the output stream.</p>
<pre> // 3) set suspended flag out.pSUSPENDED=2; </pre>	<p>Sets the suspended Flag to <b>2</b>.</p>
<pre> } else {   // file is good, output input stream, and set flag   logInfo("in good");   out=clone(in); </pre>	<p>If the input file is valid, prepare the output stream based on the input stream.</p>
<pre> // if this file was previously suspended (from resubmit/recycle), // 1) add suspense fields to output stream // 2) set suspended routing flag if (in.resubmitted &gt; 0) { </pre>	<p>If the input file is valid and was previously suspended, append the suspense fields to the output stream.</p>
<pre> // send 0 error code to flag success SuspHandler.assemble(out,   sourcePrefix+in.sourceFileName+sourceSuffix,   pipelineName,   0);   out.pSUSPENDED=1; } </pre>	<p>Set the suspended flag to <b>1</b>.</p>
<pre> } write(out); </pre>	<p>Writes out the stream.</p>

# Glossary

**AP**

aggregation processor

**BSS**

billing support system

**CC**

collection cartridge

**DC**

distribution cartridge

**EP**

enhancement processor

**GPRS**

General Packet Radio Service

**GUI**

graphical user interface

**IP**

Internet Protocol

**LDAP**

lightweight directory access protocol



**LDM**

local data manager

**NAR**

network accounting record

**NM**

node manager or Offline Mediation Controller

**OAM**

operations and management system

**OSS**

operational support system

**PDA**

personal digital assistant

**PSTN**

public switched telephone network

**RDM**

remote data manager

**SLA**

service level agreement

**SNMP**

simple network management protocol

**VoIP**

voice over IP

**VPN**

virtual private network