Oracle® Communications Offline Mediation Controller

Oracle CDR Format Cartridge User's Guide

Release 6.0

E65826-01

August 2015

ORACLE®

Oracle Communications Offline Mediation Controller Oracle CDR Format Cartridge User's Guide, Release 6.0

E65826-01

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. 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 and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon 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, Opteron, the AMD logo, and the AMD Opteron 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

This document describes how to use the Oracle Communications Offline Mediation Controller Oracle CDR Format Collection Cartridge (CC) to handle the CDR input files in the Oracle CDR format.

Audience

This document is intended for solution designers who configure Offline Mediation Controller.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Help Center:

http://docs.oracle.com

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:

https://edelivery.oracle.com

Related Documents

For more information, see the following documents:

- *Offline Mediation Controller Cartridge Development Kit Developer's Guide*: For information about how to develop a cartridge.
- *Offline Mediation Controller Cartridge Development Kit NPL Reference Guide*: For information about how to use the Node Programming Language for developing or extending a cartridge.

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.

1

Oracle CDR Format Cartridge Overview

This chapter provides an overview of the Oracle Communications Offline Meditation Controller Oracle CDR Format Collection Cartridge (CC), which parses the call detail record (CDR) data conforming to the Oracle CDR format.

Before reading this chapter, you should be familiar with Offline Mediation Controller cartridge concepts. For more information, see *Offline Mediation Controller Cartridge Development Kit Developer's Guide*.

About the Oracle CDR Format

The Oracle CDR format is the CDR file format used by Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager to process CDRs for offline charging. For more information about Oracle CDR format, see BRM documentation.

Offline Mediation Controller Oracle CDR Format CC collects the CDR input files conforming to the Oracle CDR format and parses the Oracle CDR format data into a network accounting record (NAR).

2

Creating and Configuring the Oracle CDR Format Cartridge Node

This chapter describes how to create and configure the Oracle Communications Offline Mediation Controller Oracle CDR Format Collection Cartridge (CC) cartridge node to process the call detail record (CDR) data conforming to the Oracle CDR format.

Before reading this chapter, you should be familiar with Offline Mediation Controller cartridge concepts and Node Programming Language (NPL).

Creating an Oracle CDR Format CC Node

To create an Oracle CDR Format CC node:

1. Log on to Offline Mediation Controller Administration Client.

   The Node Hosts & Nodes (logical view) screen appears.

2. In the **Mediation Hosts** table, select a host.
3. In the **Nodes on Mediation Host** section, click **New**.

   The Create a Node dialog box appears.

4. Select **Cartridge Kit** and click **Next**.
5. Select **Collection Cartridge (CC)** and click **Next**.
6. Select **Oracle CDR Format File Collection Cartridge** and click **Finish**.

   The New Node dialog box appears.

7. In the **Name** field, enter a name for the node.
8. From the **Rule File** list, select the rule file that matches the type of input file processed by the CC node.

   To edit the rule file, see "Configuring the NPL Rule File for Oracle CDR Format".

9. Click the **General** tab and do the following:

   a. From the **Debug** list, select one of the following:

      To log the informational messages in the node log file, select **OFF**.

      To log detailed debug messages in the node log file, select **ON**.

   b. In the **Max Log File Size** field, enter the maximum size in bytes for the log file. When the log file reaches its limit, the node closes the file and opens a new file. The minimum value is **50000** and the maximum value is **2000000000**.

   c. To enable node statistics, select the **Enable Statistics** check box.

d. To enable the node to read or write files in bulk, select the **Enable bulk read/write** check box.

e. In the **NARs Per File** field, enter the maximum number of network accounting records (NARs) allowed in an output file. The minimum value is **1** and the maximum value is **10000**.

f. In the **Idle Write Time** field, enter the number of seconds the node waits before moving the NAR output file to the output directory of the processing node, whether or not it has reached its maximum size. The minimum value is **1** and the maximum value is **3600**.

10. Click the **File Location** tab and do one of the following:

- To collect files using File Transport Protocol (FTP), select **Files are pulled (via FTP)**, and do the following:

  In the **Local Directory** field, enter the path and name of the directory where the CDR input files after collection are stored.

  In the **Prefix of Files after Collection** field, enter the prefix to rename the CDR input file after collection.

  In the **Suffix of Files after Collection** field, enter the suffix to rename the CDR input file after collection. The default is **.done**.

  To delete the CDR files after processing, select the **Delete Files After Processing** check box.

- To collect files from a directory, select **Files are pushed to this node**, and do the following:

  In the **Check for New Files Period** field, enter the duration of time that the node waits before checking for new files in the input directory. You also select the time unit: **Seconds**, **Minutes**, or **Hours**.

  In the **Local Directory** field, enter the path and name of the directory that contains the input files to process.

  Configure a file pattern: To configure a file pattern for the input files, select **Prefix** and enter the prefix in the **Prefix** field and suffix in the **Suffix** field. The default for **Suffix** is **.complete**; to configure a regular expression file pattern, select **Reg.Expr** and enter a file name pattern. To test the regular expression, click **Test**.

  In the **Prefix of Files after Collection** field, enter the prefix to rename the CDR input file after collection.

  In the **Suffix of Files after Collection** field, enter the suffix to rename the CDR input file after collection. The default is **.done**.

  To delete the CDR files after processing, select the **Delete Files After Processing** check box.

11. If you selected **Files are pulled (via FTP)**, click the **FTP Settings** tab and do the following:

a. From the **FTP Interval** list, select the interval in minutes or hours the node waits before checking for incoming data.

b. From the **Interrupt Timer Delay** list, select the default timeout for the FTP client when opening a socket.

c. To delete the CDR files from the remote location after the files are collected, select the **Delete remote files** check box.

d. To rename the CDR files in the remote location after the files are collected, select the **Rename remote files** check box.

e. Click **Add**.

The FTP Configuration dialog box appears.

f. From the **FTP Type** list, select one of the following:

If the client initiates the control connection with the server and the server initiates the data connection with the client, select **Regular**.

If both the data and the commands are transferred in specially formatted packets through a single connection, select **Secure (SFTP)**. When using SFTP, all data sent between the client and the server is encrypted using an encryption cipher.

If the client initiates all connections with the server, select **Passive**. The server informs the client about the port to be used for the data connection.

g. In the **Host** field, enter the IP address of the FTP server.

h. In the **Username** field, enter a valid user name for accessing the FTP server.

i. In the **Password** field, enter the password for the user name.

j. In the **Verify Password** field, re-enter the password used in the **Password** field.

k. In the **Directory** field, enter the name of the remote directory that contains the input files to process.

l. Configure a file pattern: To configure a file pattern for the input files, select **Prefix** and enter the prefix in the **Prefix** field and suffix in the **Suffix** field; to configure a regular expression file pattern, select **Reg.Expr** and enter a file name pattern. To test the regular expression, click **Test**.

m. Click **OK**.

The FTP configuration is saved and the FTP Configuration dialog box closes.

12. Click the **Advanced** tab and do the following:

a. To enable file-level transactions, select the **File Level Transaction** check box.

b. To enable validating the name of the input file for duplicate file names, select the **Reject Files With Duplicate File Name** check box.

The **Expiry Time to Reject Duplicate Files** field is enabled.

c. In the **Expiry Time to Reject Duplicate Files** field, enter the time in minutes after which the file names are flushed from memory. The minimum value is **1**, and the maximum value is **129600**. The default is **1**.

d. Do one of the following:

To enable validating the order of the sequence number in the input file, select the **Sequence Ordering by File Name** check box.

To enable multithreading to process multiple files in parallel, select the **Multi Threaded** check box and enter the number of processing threads you require in the **Processing Threads** field. The maximum value is **20**. If you require the order of the output data across all threads to be processed in the same order as the input data, select the **Enable Ordering** check box.

13. Click the **Schema File** tab and do the following:

a. From the **Schema file** list, select the schema file that is used to parse the Oracle CDR format data files.

14. Click the **Destination** tab and do the following:

a. Select the **Enable** check box, which enables the connection between the CC node and any destination cartridge node.

b. From the **Routing** list, select one of the following:

If the **Enable** check box is not selected, select **None**.

To enable multicast routing between the CC node and the destination cartridge node, select **Multicast**.

To enable round-robin routing between the CC node and the destination cartridge node, select **Round Robin**.

15. Click **Save**.

Configuring the NPL Rule File for Oracle CDR Format

To configure the NPL rule file for Oracle CDR format:

1. Log on to Offline Mediation Controller Administration Client.

The Node Hosts & Nodes (logical view) screen appears.

2. In the **Mediation Hosts** table, select the mediation host that contains the Oracle CDR Format CC node.

3. In the **Nodes on Mediation Host** section, select the Oracle CDR format CC node that you want to configure, and click **Edit**.

The Node dialog box appears.

4. From the **Rule File** list, select one of the sample NPL rule files.

5. Click **Edit** for the selected rule file.

The NPL Editor dialog box appears.

6. In the configuration block, do the following:

- Add the following entry, which defines the Oracle CDR format header record fields written to the NAR output file:

```
AddHeaderFields "headerFieldName1, headerFieldName2,...";
```

where *headerFieldNameX* is the name of the field in the Oracle CDR format header record.

If you want all the fields in the header record to be part of every NAR written by the Oracle CDR format CC node, specify **ALL**.

For example:

```
AddHeaderFields "RECORD_TYPE, RECORD_NUMBER";
```

- Add the following entry, which defines the Oracle CDR format trailer record fields written to the NAR output file:

```
AddTrailerFields " trailerFieldName1, trailerFieldName2,...";
```

where *trailerFieldNameX* is the name of the field in the Oracle CDR format trailer record.

For example:

```
AddTrailerFields "RECORD_TYPE";
```

- Add the following entry, which defines the order of the records:

  `RecordOrder "HEADER,DETAIL*,TRAILER";`

  For the DETAIL* record field, define the associated type of records and sub-records. For example:

```
RecordOrder "HEADER,DETAIL*,TRAILER";
DETAIL "ASSOCIATED_CHARGE*,ASSOCIATED_ZONE,ASSOCIATED_INFRANET*,ASSOCIATED_
GPRS,ASSOCIATED_WAP,ASSOCIATED_GSMW*,ASSOCIATED_CAMEL,ASSOCIATED_
MESSAGE,ASSOCIATED_SMS,ASSOCIATED_MMS,ASSOCIATED_SUSPENSE,ASSOCIATED_
ROAMING,ASSOCIATED_CIBER";
ASSOCIATED_CHARGE "RUM_MAP*,CHARGE_PACKET*,DISCOUNT_PACKET*,TAX_PACKET*";
DISCOUNT_PACKET "DISCOUNT_SUBBALANCE*";
ASSOCIATED_ZONE "ZONE_PACKET*";
ASSOCIATED_INFRANET "BALANCE_PACKET*,SUB_BAL_IMPACT*,MONITOR_
PACKET*,MONITOR_SUB_BAL_IMPACT*";
SUB_BAL_IMPACT "SUB_BAL*";
MONITOR_SUB_BAL_IMPACT "MONITOR_SUB_BAL*";
ASSOCIATED_GSMW "SS_EVENT_PACKET*";
```

7. Compile and save the file.
8. Close the NPL Editor dialog box.
9. Click **Save**.

   The configuration is saved.

Configuring a New Oracle CDR Format Schema for the Oracle CDR Format CC Node

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 configure a new Oracle CDR format schema for the Oracle CDR Format CC node:

1. Log on to Offline Mediation Controller Administration Client.

   The Node Hosts & Nodes (logical view) screen appears.
2. In the **Mediation Hosts** table, select the mediation host that contains the Oracle CDR Format CC node.
3. Stop the Oracle CDR Format CC node.
4. Copy your new schema file into the *OMC_Home*/**config/sol42/schema** directory.
5. In the Offline Mediation Controller Administration Client **Mediation Hosts** table, select the mediation host that contains the Oracle CDR Format CC node.
6. In the **Nodes on Mediation Host** section, select the Oracle CDR Format CC node for which you want to change the schema file, and click **Edit**.

   The Node dialog box appears.
7. In the **Node Configuration** section, click the **Schema File** tab.

8. From the **Schema file** list, select the new schema file, which contains the schema information used to parse the Oracle CDR format data files.
9. Click **Save**.

   The configuration is saved.
10. Start the Oracle CDR Format CC node.

3

Working with Oracle CDR Format Java Hooks in NPL

This chapter lists and describes the Java hooks available for the Oracle Communications Offline Mediation Controller Oracle CDR Format Collection Cartridge (CC).

About Oracle CDR Format Java Hooks

Java hooks are an advanced feature of NPL (Node Programming Language) that enable Offline Mediation Controller to call a Java method from an NPL program. For more information on using Java hooks with NPL, see the discussion on Java hooks in *Offline Mediation Controller Cartridge Development Kit NPL Reference Guide*.

Table 3–1 lists the Oracle CDR Format Java hooks methods.

Table 3–1 Oracle CDR Format Java Hooks Method Summary

Modifier and Type	Method and Description
IntField	hasHeaderFields(DCFieldContainer *in*) Verifies if the record contains header fields.
IntField	hasTrailerFields(DCFieldContainer *in*) Verifies if the record contains trailer fields.
IntField	hasAnyAssociated(DCFieldContainer *in*) Verifies if the record contains any associated block.
IntField	hasAssociatedData(DCFieldContainer *in*, StringField *associatedBlockName*) Verifies if the record contains the associated block *associatedBlockName*.
ListField	getAssociatedData(DCFieldContainer *in*, StringField *associatedBlockName*) Returns the *associatedBlockName* as a Listfield
IntField	getAssociatedIntField(DCFieldContainer *in*, StringField *associatedBlockName*, StringField *fieldName*) Returns the integer field *fieldName* from the associated block *associatedBlockName*.
DoubleField	getAssociatedDoubleField(DCFieldContainer *in*, StringField *associatedBlockName*, StringField *fieldName*) Returns the double field *fieldName* from the associated block *associatedBlockName*.
LongField	getAssociatedLongField(DCFieldContainer *in*, StringField *associatedBlockName*, StringField *fieldName*) Returns the long field *fieldName* from the associated block *associatedBlockName*.

Table 3–1 (Cont.) Oracle CDR Format Java Hooks Method Summary

Modifier and Type	Method and Description
StringField	getAssociatedStringField(DCFieldContainer *in*, StringField *associatedBlockName*, StringField *fieldName*) Returns the string field *fieldName* from the associated block *associatedBlockName*.
IntField	getIntFieldFromList(ListField *lstFld*, StringField *fieldName*) Returns the integer field *fieldName* from *lstFld*.
LongField	getLongFieldFromList(ListField *lstFld*, StringField *fieldName*) Returns the long field *fieldName* from *lstFld*.
DoubleField	getDoubleFieldFromList(ListField *lstFld*, StringField *fieldName*) Returns the double field *fieldName* from *lstFld*.
StringField	getStringFieldFromList(ListField *lstFld*, StringField *fieldName*) Returns the string field *fieldName* from *lstFld*.

Oracle CDR Format Java Hook Method Details

The section describes the Oracle CDR Format Java hook methods.

hasHeaderFields

```
IntField hasHeaderFields(DCFieldContainer in)
```

Usage

This function verifies if the record contains header fields.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

Returns

1 (true) if the record contains header fields.

0 (false) if the record does not contain header fields.

hasTrailerFields

```
IntField hasTrailerFields(DCFieldContainer in)
```

Usage

This function verifies if the record contains trailer fields.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

Returns

1 (true) if the record contains trailer fields.

0 (false) if the record does not contain trailer fields.

hasAnyAssociated

```
IntField hasAnyAssociated(DCFieldContainer in)
```

Usage

This function verifies if the record contains any associated block.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

Returns

1 (true) if the record contains the associated block.

0 (false) if the record does not contain the associated block.

hasAssociatedData

```
IntField hasAssociatedData(DCFieldContainer in, StringField associatedBlockName)
```

Usage

This function verifies if the record contains the associated block *associatedBlockName*.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

Returns

1 (true) if the record contains the associated block.

0 (false) if the record does not contain the associated block.

getAssociatedData

```
ListField getAssociatedData(DCFieldContainer in, StringField associatedBlockName)
```

Usage

This function returns the *associatedBlockName* as a Listfield.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

Returns

The *associatedBlockName* as a List field if the value is found. Returns an empty List field if the value is not found.

getAssociatedIntField

```
IntField getAssociatedIntField(DCFieldContainer in,StringField
associatedBlockName,StringField fieldName)
```

Usage

This function returns the integer field *fieldName* from the associated block *associatedBlockName*.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

fieldName is the field name for which the value is to be returned.

Returns

The integer field matching the search criteria.

getAssociatedDoubleField

```
DoubleField getAssociatedDoubleField(DCFieldContainer in,StringField
associatedBlockName,StringField fieldName)
```

Usage

This function returns the double field *fieldName* from the associated block *associatedBlockName*.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

fieldName is the field name for which the value is to be returned.

Returns

The double field matching the search criteria.

getAssociatedLongField

```
LongField getAssociatedLongField(DCFieldContainer in,StringField
associatedBlockName,StringField fieldName)
```

Usage

This function returns the long field *fieldName* from the associated block *associatedBlockName*.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

fieldName is the field name for which the value is to be returned.

Returns

The long field matching the search criteria.

getAssociatedStringField

```
StringField getAssociatedStringField(DCFieldContainer in,StringField
associatedBlockName,StringField fieldName)
```

Usage

This function returns the string field *fieldName* from the associated block *associatedBlockName*.

Parameters

in is the DCFieldContainer that contains the field names and their associated values.

associatedBlockName is the name of the associated block.

fieldName is the field name for which the value is to be returned.

Returns

The string field matching the search criteria.

getIntFieldFromList

```
IntField getIntFieldFromList(ListField lstFld, StringField fieldName)
```

Usage

This function returns the integer field *fieldName* from *lstFld*.

Parameters

lstFld is the ListField that contains the field names and values.

fieldName is the field name for which the value is to be returned.

Returns

The integer field matching the search criteria.

getLongFieldFromList

```
LongField getLongFieldFromList(ListField lstFld, StringField fieldName)
```

Usage

This function returns the long field *fieldName* from *lstFld*.

Parameters

lstFld is the ListField that contains the field names and values.

fieldName is the field name for which the value is to be returned.

Returns

The long field matching the search criteria.

getDoubleFieldFromList

```
DoubleField getDoubleFieldFromList(ListField lstFld, StringField fieldName)
```

Usage

This function returns the double field *fieldName* from *lstFld*.

Parameters

lstFld is the ListField that contains the field names and values.

fieldName is the field name for which the value is to be returned.

Returns

The double field matching the search criteria.

getStringFieldFromList

```
StringField getStringFieldFromList(ListField lstFld, StringField fieldName)
```

Usage

This function returns the string field *fieldName* from *lstFld*.

Parameters

lstFld is the ListField that contains the field names and values.

fieldName is the field name for which the value is to be returned.

Returns

The string field matching the search criteria.