Contents

Preface .......................................................................................................................................................... v
Audience ......................................................................................................................................................... v
Downloading Oracle Communications Documentation ................................................................. v
Documentation Accessibility ....................................................................................................................... v

1 Managing Balances on the Intelligent Network

About Diameter Balance Manager ........................................................................................................ 1-1
Diameter Balance Manager Architecture .............................................................................................. 1-2
  Pre-Processing Pipeline ..................................................................................................................... 1-3
  Post-Processing Pipeline ................................................................................................................... 1-4
  Time-Out Pipeline .............................................................................................................................. 1-5
About the Diameter Balance Manager Input and Output Grammar ................................................... 1-5
  Capability Exchange with the IN ...................................................................................................... 1-6
Communication Between the CM, Diameter Balance Manager, and the IN ...................................... 1-6
  About the Diameter Balance Manager High-Availability ................................................................ 1-6
    Specifying the Interval for Checking Connection to the IN ........................................................... 1-7
About Using Diameter Balance Manager ........................................................................................... 1-7
  Configuring Diameter Balance Manager .......................................................................................... 1-7
  Configuring Diameter Balance Manager to Connect to the IN ..................................................... 1-8
  Configuring the CM to Connect to Diameter Balance Manager .................................................... 1-8
  Configuring the Input and Output Grammar Modules ........................................................................ 1-8
  Configuring a Time-Out Value for Each Input Request .................................................................... 1-10
Starting and Stopping Diameter Balance Manager ............................................................................... 1-10
Monitoring Diameter Balance Manager ............................................................................................ 1-10

2 Diameter Balance Manager Protocol Translation

About Diameter Balance Manager Protocol Support ........................................................................... 2-1
About Converting Messages Between Flist and EDR Formats ............................................................ 2-2
  Flist Input Grammar .......................................................................................................................... 2-2
    Flist Field Blocks ......................................................................................................................... 2-3
    Error Handling ............................................................................................................................... 2-3
  Flist Input Mapping ........................................................................................................................... 2-3
  Flist Stream Format Description ..................................................................................................... 2-3
  Flist Output Grammar ....................................................................................................................... 2-4
    Error Handling ............................................................................................................................... 2-4
3 Accounting EDR Container Description

About the Accounting EDR Container Description ................................................................. 3-1
HEADER Content Type for the Accounting EDRs ................................................................. 3-1
DETAIL Content Type for Accounting EDRs ........................................................................ 3-2
Function Module Fields ....................................................................................................... 3-2
ASSOCIATED_PROTOCOL_INFO Block ............................................................................. 3-3
Preface

This book provides an overview of the Oracle Communications Billing and Revenue Management (BRM) Diameter Balance Manager.

Audience

This document is intended for developers and system administrators.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Technology Network:

http://docs.oracle.com

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

https://edelivery.oracle.com

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at


Access to Oracle Support

Oracle customers 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.
Managing Balances on the Intelligent Network

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Diameter Balance Manager architecture and how to configure Diameter Balance Manager.

**Note:** Diameter Balance Manager is an optional feature that requires a separate license.

Before reading this chapter, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

**Note:** By default, Diameter Balance Manager is not integrated into the BRM business logic flow. To include it in the business logic flow, you must write and configure a custom Facilities Module (FM) that sends messages to the Diameter Balance Manager. For information on creating custom FMs, see the discussion of writing custom Facilities Module in *BRM Developer’s Guide*.

**About Diameter Balance Manager**

Diameter Balance Manager facilitates communication between BRM Connection Manager (CM) and the Intelligent Network (IN) using the Diameter protocol. Diameter Balance Manager manages the balances maintained on the IN by using the credit control capabilities of the IN. For example, Diameter Balance Manager sends usage information for prepaid services, which the IN uses to rate and impact the customer’s balance. For real-time services, Diameter Balance Manager sends the authorization requests to the IN and, based on the available resources, the IN allows or rejects a request. When accounting requests are sent, the IN credits or debits the account balances.

Diameter Balance Manager performs the following operations to maintain communication between the CM and the IN:

- Converts the request messages from the CM in flist format to Diameter protocol before sending them to the IN and converts the Diameter protocol response messages to flists before sending them to the CM. See "Diameter Balance Manager Architecture" for more information.

- Converts messages from synchronous to asynchronous and back. It communicates with the CM in a synchronous mode using the Portal Communication Protocol (PCP) and with the IN in an asynchronous mode using the Diameter protocol.
"Communication Between the CM, Diameter Balance Manager, and the IN" for more information.

- Handles the Capability Exchange with the IN by performing the initial handshake between Diameter Balance Manager and the IN. The request goes as Capability Exchange Request (CER) message to the IN and the expected response comes as Capability Exchange Answer (CEA) message from the IN. See "Capability Exchange with the IN" for more information.

- Checks the status of the IN by sending keep-alive messages to the IN, when there is no message traffic between the CM and the IN for a specified amount of time. The keep-alive message goes as Device Watchdog Request (DWR) to the IN, and the response comes as Device Watchdog Answer (DWA) from the IN to Diameter Balance Manager. See "About Using Diameter Balance Manager" for more information.

- Handles IN failover by switching from the active IN instance to the passive IN instance and restoring the connection to the active instance when it recovers. See "About the Diameter Balance Manager High-Availability" for more information.

---

**Note:** All time-out values you specify for the various processes in Diameter Balance Manager must be in milliseconds.

---

**Diameter Balance Manager Architecture**

Diameter Balance Manager is a set of pipelines run by BRM Pipeline Manager. Each instance of Diameter Balance Manager consists of the following set of pipelines, each of which runs in its own thread and has its own input buffer:

- Pre-Processing Pipeline
- Post-Processing Pipeline
- Time-Out Pipeline

Figure 1–1 illustrates the Diameter Balance Manager architecture.
Figure 1–1 Diameter Balance Manager Architecture

Pre-Processing Pipeline

The pre-processing pipeline receives requests in flist format from the CM and performs the following tasks to send the requests to the IN, the Diameter server:

- Converts the flist request to an EDR by using the input grammar that you can customize. See "About Converting Messages Between Flist and EDR Formats" for more information.

- Specifies a time-out value for the EDR by using "FCT_iScript". See BRM Configuring Pipeline Rating and Discounting.

You can customize the default iScript to specify a different time-out value for different types of messages.

- Schedules a timer for each CM request by using FCT_Timer and creates a duplicate EDR, which it sends to the time-out queue, if the request times out. For more information, see "Time-Out Pipeline".

FCT_Timer sets a unique timer identifier in the EDR field after successfully scheduling the timer. It also checks the heartbeat of the IN by sending DWR messages to the IN.
Inserts correlation data for each request, which consists of a unique session identifier, a CM socket identifier, and a unique timer identifier in the DAT_Correlation module.

The FCT_SetCorrelation iScript module performs this function.

When the responses come from the IN in an asynchronous mode, Diameter Balance Manager uses this information to determine the CM connection to which it needs to send the responses.

The FCT_GetCorrelation iScript module performs this function.

Converts EDRs into Diameter messages containing attribute-value pairs (AVP) by using the output grammar. See "About Converting Messages Between Flist and EDR Formats" for more information.

Sends the requests to the active IN by using the DAT_ConnectionManager module to get the connection context. See the discussion about DAT_ConnectionManager in BRM Configuring Pipeline Rating and Discounting for more information.

Sends system-level messages and commands, such as DWR and CER to the IN. DWR messages are generated by a timer event and CER messages are generated by connection and reconnection events in Diameter Balance Manager. You can customize them by using the output grammar in the pre-processing pipeline. They do not have a timer scheduled.

### Post-Processing Pipeline

The post-processing pipeline receives the response from the IN. If the responses from the IN are received within the time specified for the type of request, the post-processing pipeline performs the following tasks:

- Converts the Diameter message to an EDR by using the input grammar that you can customize. See "Diameter Input Grammar" for more information on the input grammar.
- Copies the CM socket identifier and timer identifier fields from the DAT_Correlation module to the EDR and deletes the corresponding correlation data from DAT_Correlation module. See the description of DAT_Correlation in BRM Configuring Pipeline Rating and Discounting for more information.
- Cancels the timer by using the unique timer ID from the EDR in the FCT_CancelTimer module.
- Converts the EDR into an flist message using the output grammar and sends the responses to the correct CM using the socket identifier in the EDR.
- Processes responses to system messages and commands, such as DWA and CEA from the IN.

If the responses, with successful operation, are received after the timer expires, the post-processing pipeline performs the following tasks:

- Converts the Diameter message to an EDR by using the input grammar that you can customize. See "Diameter Input Grammar" for more information on the input grammar.
- Sends the response to a stream that you specify in the registry file, other than the stream that sends responses to the CM, for example, a file or dev_null.

You need to write an iScript or create a processing pipeline to process the EDRs in the file.
Note: For delayed responses, the post-processing pipeline does not handle the correlation data, which are removed by the time-out pipeline, or cancel the timer, because it is already expired.

Delayed responses with failed operations are ignored by the pipeline.

Time-Out Pipeline

When a request times out (that is, it is not processed within the time you specified in the FCT_iScript) FCT_Timer in the pre-processing pipeline sends a duplicate EDR to the time-out queue. The time-out pipeline performs the following tasks to process the timed-out requests:

- Reads the duplicate EDR from the time-out queue.
- Copies the CM socket identifier and timer identifier fields from the DAT_Correlation module to the EDR and deletes the corresponding correlation data from the DAT_Correlation module.
- Adds the output data you specify in the output data block and information about the error in the error buffer data block in the EDR. See "PCM_OP_EBUF Block" in BRM AAA Gateway Manager for more information.
- Converts the EDR into an flist message by using the output grammar and sends the response to the correct CM by using the socket identifier in the EDR.

The CM needs to take appropriate action for this request, for example, send the request to the IN again.

About the Diameter Balance Manager Input and Output Grammar

Diameter Balance Manager input and output grammar define the mapping between data in the flist format, an EDR and the Diameter protocol. You can customize the grammar and mapping files. See "Diameter Balance Manager Protocol Translation" for more information on the grammar and mapping files.

Diameter Balance Manager uses the following modules to translate data to the correct format required by the processing modules or components:

- **InputSocketFlistManager**: Translates the incoming requests in flist format from the CM to an EDR that the modules in the pre-processing pipeline can process. This module is a server for the CM.
- **OutputSocketDiameterManager**: Translates the EDR to a Diameter request before sending it to the IN. This module is a client to the IN.
- **InputSocketDiameterManager**: Translates the response returned from the IN in the Diameter protocol to an EDR for the post-processing pipeline modules to process. This module is a client to the IN.
- **OutputSocketFlistManager**: Translates the EDR into an flist before sending the response to the CM. This module is a server for the CM.

Note: You must configure each of the input and output modules as a client or server in the Diameter Balance Manager registry file. See "Configuring the Input and Output Grammar Modules" for more information.
Capability Exchange with the IN

As soon as Diameter Balance Manager connects to the IN through a connection in the DAT_ConnectionManager module, before it starts any business transaction with the IN, it performs an initial handshake with the IN instances and establishes its identity and capabilities. It sends a CER message to the IN and receives a CEA response. When this message exchange is successful, Diameter Balance Manager can send CM requests to the IN.

Diameter Balance Manager establishes a connection to the primary IN server. When the connection to the primary server fails, it connects to the next available server.

Communication Between the CM, Diameter Balance Manager, and the IN

The CM sends a request in flist format to Diameter Balance Manager in a synchronous mode and then waits for a response without performing any other operation.

Diameter Balance Manager receives the synchronous messages from the CM and sends them to the IN in an asynchronous mode by using a single connection to the IN in the DAT_ConnectionManager module. A Diameter Balance Manager thread keeps sending the messages to the IN and another thread keeps receiving the messages from IN.

Diameter Balance Manager uses the DAT_Correlation module to determine the CM to which to send a response, based on the session ID that maps to the CM socket ID. It removes the socket ID and the session ID for the response from the DAT_Correlation module and cancels the timer for the response. Then it forwards the response to the CM and starts processing the next request in the queue.

If the request times out, the FCT_iScript module in the time-out pipeline retrieves the correlation data and deletes it from the DAT_Correlation module memory. The timer is expired and therefore not deleted.

The Diameter Balance Manager gets a connection context from the DAT_ConnectionManager module during startup and contacts the DAT_ConnectionManager module only when there is a connection failure.

About the Diameter Balance Manager High-Availability

You can set up Diameter Balance Manager to fail over to a secondary server when communication with the primary server fails. You configure the DAT_ConnectionManager module with connections to two IN instances, a primary and a secondary. See "Configuring Diameter Balance Manager to Connect to the IN" for more information.

If there is no message traffic between the CM and the IN within a period of time you specify, the KeepAlive handler in the FCT_Timer module generates an EDR. Because the EDR does not contain an Flist block, the FlistToDiameter_iScript module interprets it as a DWR message and sends it to the IN. The DWA that is returned from the server contains information about the server failure or the server shutting down. See "Specifying the Interval for Checking Connection to the IN" for information on setting the interval for checking communication between the CM and IN.

Diameter Balance Manager sends that information to the DAT_ConnectionManager module, which fails over to the secondary IN instance. The Diameter Balance Manager connects to the secondary server in the DAT_ConnectionManager module. The KeepAlive handler in the FCT_Timer module resets the timer to start the next keep-alive time period.
If there is message exchange within the time period that you specify, the KeepAlive handler resets the timer to start the next period.

---

**Note:** You can customize the DWR messages by using the pre-processing pipeline output grammar.

---

**Specifying the Interval for Checking Connection to the IN**

To specify the time period for Diameter Balance Manager to check the CM-to-IN traffic, enter a value in milliseconds in the FCT_Timer `KeepAliveInterval` registry entry. For example,

```
KeepAliveInterval = 180000
```

---

**About Using Diameter Balance Manager**

To use Diameter Balance Manager:

1. Install Diameter Balance Manager. See *BRM Installation Guide* for instructions on installing Diameter Balance Manager.
2. Customize the iScripts and iRules in the processing pipelines.
3. Write any new iScript or iRules modules and configure them in the pipelines.
4. Customize the mapping between the flist, EDR, and Diameter formats and the corresponding grammar files and specify the location of the files in the appropriate registry entries.
5. *(Optional)* Customize the DWR and CER messages.
6. Configure Diameter Balance Manager by editing the `diameter_balance.reg` file. See "Configuring Diameter Balance Manager" for more information.

---

**Configuring Diameter Balance Manager**

You need to perform the following tasks to configure the Diameter Balance Manager:

- Configure all the pipeline modules in the pre-processing and post-processing pipelines. See the sample `diameter_balance.reg` file.
- Configure the FCT_Timer module in the `diameter_balance.reg` file to specify time-out values for the EDRs. See the description of FCT_Timer in *BRM Configuring Pipeline Rating and Discounting* for more information.
- Specify the path to the various input and output grammar, mapping, data description, and log files for the pipelines.
- Configure the input and output grammar modules. See "Configuring the Input and Output Grammar Modules" for more information.
- Configure a timeout value for each input request. See "Configuring a Time-Out Value for Each Input Request" for more information.
- Configure Diameter Balance Manager to connect to the IN. See "Configuring Diameter Balance Manager to Connect to the IN" and the description of DAT_ConnectionManager in *BRM Configuring Pipeline Rating and Discounting* for more information.
- Configure the CM to connect to Diameter Balance Manager. See "Configuring the CM to Connect to Diameter Balance Manager" for more information.
Configure the DAT_Correlation module. See the description of DAT_Correlation in BRM Configuring Pipeline Rating and Discounting for more information.

Configuring Diameter Balance Manager to Connect to the IN

To connect Diameter Balance Manager to the IN, configure the DAT_ConnectionManager module.

To configure the DAT_ConnectionManager module, you enter the IP address and port number for two IN server instances, a primary and a secondary. When the primary server is not responsive, it fails over to the secondary server and routes all the messages to the secondary server. It checks the primary server at regular intervals and switches back to it as soon as it becomes active.

DAT_ConnectionManager also initializes the Global Data Dictionary (GDD) during startup. By default, it uses a file containing the data dictionary flist. You can customize it to use a CM to initialize the GDD by accessing the database.

See the description of DAT_ConnectionManager in BRM Configuring Pipeline Rating and Discounting for more information.

Configuring the CM to Connect to Diameter Balance Manager

To connect the CM to Diameter Balance Manager, in the CM pin.conf file, add a DM pointer entry with the host name and port number of the system where Diameter Balance Manager is installed. For example,

```cm dm_pointer 0.0.0.1 DM_Diameter 11950```

Configuring the Input and Output Grammar Modules

To configure the input and output grammar modules:

1. For the InputSocketFlistManager, which connects to the CM, specify the CM Port number in the Diameter Balance Manager registry file. For example:

   ```
   Input {
     InputModule {
       ...
       Module {
         ...
         InputStream {
           ...
           Module {
             Port = 27228
           }
         }
       }
     }
   }
   ```

2. For the OutputSocketFlistManager, do not specify anything.

   ```
   Output {
     OutputModule
   }
   ```
{  
  ...
  Module
  {
    ...
    OutputStream
    {
      ...
      Module
      {
        
        }
      }
    }
  }
}

3. For the **OutputSocketDiameterManager** in the pre-processing pipeline, which connects to the IN, specify a connection to the DAT_ConnectionManager module. For example:

```plaintext
Output
{
  ...
  OutputCollection
  {
    EdrOutput
    {
      ...
      Module
      {
        ...
        OutputStream
        {
          ...
          Module
          {
            ConnectionManagerDataModule = ifw.DataPool.INConnectionManager.Module
          }
        }
      }
    }
  }
}
```

4. For the **InputSocketDiameterManager** in the post-processing pipeline, which connects to the IN, specify a connection to the DAT_ConnectionManager module. For example:

```plaintext
Input
{
  ...
 InputModule
  {
    ...
    Module
    {
      ...
      InputStream
      {
        ...
      }
    }
  }
}
```
Configuring a Time-Out Value for Each Input Request

To configure a time-out value for each input request:

1. Open the IRL_Router.data file in the Pipeline_home/iScriptLib/AAA directory.
2. For each type of request, specify the time-out value in milliseconds by using the following syntax:
   
   OpcodeNumber;OutputStream;TimeoutOffset
   
   For example,
   
   1;EDR_OUT;6000
   
3. Save and close the file.

Starting and Stopping Diameter Balance Manager

To start the Diameter Balance Manager, use the pin_ctl utility. See "Starting Pipeline Manager components with pin_ctl" in BRM System Administrator’s Guide for more information.

To stop Diameter Balance Manager, use the following command:

   pin_ctl stop diameter

Monitoring Diameter Balance Manager

You can monitor Diameter Balance Manager by checking the following pipeline, process, and stream log files:

- ./log/pipeline/PreProcessingPipe.log
- ./log/stream/PreProcessingStream.log
- ./log/pipeline/PostProcessingPipe.log
- ./log/stream/streamPostProcessingStream.log
- ./log/pipeline/TimeoutPipe.log
- ./log/pipeline/CERGeneratorPipe.log
- ./log/pipeline/CEADisplayPipe.log
- ./log/process/processDiameter.log

Note: The module that connects to the IN is a client, and you specify the connection for the IN server.
This chapter describes the Oracle Communications Billing and Revenue Management (BRM) message conversion process and the files used to convert messages from the Connection Manager (CM) in flist format to EDR containers that the pipeline modules can process and then to the Diameter protocol that the Intelligent Network (IN) requires.

See "Managing Balances on the Intelligent Network" for more information on Diameter Balance Manager and how to use it.

**About Diameter Balance Manager Protocol Support**

To exchange messages between the CM and the IN, Diameter Balance Manager performs the following tasks:

- Converts flist messages from the CM to EDRs and back from EDRs to flists. See "About Converting Messages Between Flist and EDR Formats" for more information.

- Converts EDRs to Diameter messages and back from Diameter messages to EDRs. See "About Converting Messages Between EDR and Diameter Formats" for more information.

**Figure 2–1** illustrates the conversion process between the flist, EDR, and Diameter formats:
About Converting Messages Between Flist and EDR Formats

To convert flist messages to EDR containers and back, Diameter Balance Manager uses these files:

- **Flist Input Grammar**
- **Flist Input Mapping**
- **Flist Stream Format Description**
- **Flist Output Grammar**
- **Flist Output Mapping**

### Flist Input Grammar

The flist input grammar file (*Pipeline_home*/Flist/Flist_v01_InGrammar.dsc) contains instructions for generating EDR containers and filling them with data from incoming flist-based messages. At startup, Diameter Balance Manager uses this grammar file to create a parser that implements these instructions.

By default, the flist input grammar supports the following opcodes:
- PCM_OP_CREATE_OBJ
- PCM_OP_DELETE_OBJ
- PCM_OP_TEST_LOOPBACK

**Flist Field Blocks**

Each incoming flist message begins with a wire header. This header has a fixed length and is mapped to a block in the input grammar named WIRE_HEADER. The grammar is driven by the opcode number included in this header.

Because fields following the wire header in flist-based messages can be arranged in any order, each field is defined as a separate block in the grammar.

Flist blocks contain a field descriptor in the following format (the field descriptor is defined in the "Flist Stream Format Description" file):

FlistType("{flist_field_name}");

where
- *Type* is the data type of the flist field (see "Flist Stream Format Description" for more information).
- *flist_field_name* is the name of the flist field.

**Flist Arrays and Substructs**

The headers for flist arrays and substructs contain two important values: index and size. To retrieve both values, the parser must treat them as two fields. Thus, the grammar defines a separate field for each value:

- The *FlistIndex* field ("PIN_FLD_ARRAY") contains the index of an array.
- The *FlistContainer* field ("PIN_FLD_ARRAY/SUBSTRUCT_NAME") contains the number of fields in a substruct or the number of elements in an array.

**Error Handling**

The grammar calls the edrAddError function to report errors, such as the failure to create a data block.

These errors are recorded in the Diameter Balance Manager stream log (`Pipeline_home/log/stream/`). The block in which the error occurred is specified in the log.

**Flist Input Mapping**

To transfer data from fields in incoming flist messages to the appropriate fields in EDR containers, the parser follows the mapping in the flist input mapping file (`Pipeline_home/formatDesc/Formats/Flist/Flist_v01_InMap.dsc`).

Because flist field names are not unique, this file contains a mapping block for each context in which the field name appears. For example, in one opcode’s input flist, PIN_FLD_POID can be included at every level in every array. In addition, it can be included in the input flists of multiple opcodes.

**Flist Stream Format Description**

The flist stream format description file (`Pipeline_home/formatDesc/Formats/Flist/Flist_v01.dsc`) specifies the data type of the fields in flist messages that are supported by Diameter Balance Manager. For example, this entry specifies that the data type of the PIN_FLD_POID field is FlistPoid:
PIN_FLD_POID FlistPoid('PIN_FLD_POID');

The input parser uses this file to identify the data type of fields in incoming messages from the CM. The output parser uses this file to format responses from EDR containers to flists.

To process flist messages, Diameter Balance Manager supports several flist data types. See "Understanding Flists and Storable Classes" in *BRM Developer’s Guide* for more information.

**Flist Output Grammar**

The flist output grammar file (*Pipeline_home/formatDesc/Formats/Flist/Flist_v01_OutGrammar.dsc*) contains instructions for generating outgoing flist messages and filling them with data from EDR containers. At startup, Diameter Balance Manager uses this grammar to create a parser that implements the instructions.

The output grammar file contains a block for each Diameter Balance Manager function; only the function blocks associated with the opcode specified in the flist wire header are incorporated into the parser. Each function block contains the following blocks:

- **Mandatory fields**: The `edrOutputMapToBuffer` function transfers the data in mandatory fields to a buffer. When BRM finishes processing a request, the `outputWrite()` function transfers the data in all such function-block buffers into an outgoing flist message.

- **Optional fields**: Before data in optional fields is transferred to a buffer, the `edrInternalState` function verifies that the fields are included in the output flists of the opcodes that implemented the request or in the protocol-specific data in the `ASSOCIATED_PROTOCOL_INFO` block.

**Error Handling**

The `logPipeline` function writes all errors to the pipeline log.

**Flist Output Mapping**

To transfer data from fields in the EDR containers to the appropriate fields in outgoing flist messages, the parser follows the mapping in the flist output mapping file (`*Pipeline_home/formatDesc/Formats/Flist/Flist_v01_OutMap.dsc*`).

**About Converting Messages Between EDR and Diameter Formats**

To convert EDR containers to Diameter messages and back, Diameter Balance Manager uses these files:

- Diameter Input Grammar
- Diameter Input Mapping
- Diameter Stream Format Description
- Diameter Output Grammar
- Diameter Output Mapping
Diameter Input Grammar

The Diameter input grammar file (Pipeline_home/formatDesc/Formats/Diameter/Diameter_v01_InGrammar.dsc) contains instructions for generating EDR containers and filling them with data from Diameter messages returned from the IN. At startup, Diameter Balance Manager uses this grammar to create a parser that implements the instructions.

By default, the Diameter input grammar supports these messages:

- DIA_CC_REQUEST
- DIA_CE_REQUEST
- DIA_DW_REQUEST

Diameter Field Blocks

Diameter messages consist of a message header followed by a list of attribute-value pairs (AVPs). The AVPs can be listed in any order. Each header contains a command code to identify the message type, as listed in Table 2–1.

See "Supported Diameter Messages" for more information about these message types.

In the Diameter input grammar, the Diameter header is defined as a fixed block, and each field that corresponds to an AVP is defined as a separate block. Each AVP block contains a field descriptor in one of the following formats (the field descriptor is defined in the "Diameter Stream Format Description" file):

- Diameter Type1 (AVP_Code, "Flags", Vendor_ID)
- Diameter Type2 (AVP_Code, "Flags", Vendor_ID, Size)
- Diameter Type3 (AVP_Code, "Flags", Vendor_ID, Zone)

where

- Type1 is data type String or Grouped
- Type2 is data type Integer or Decimal
- Type3 is data type Time
- AVP_Code is a value defined by the Internet Assigned Numbers Authority (IANA) that uniquely identifies the message type. See "Supported Diameter Messages" for more information.

- Flags is one or more of the flags listed in Table 2–2.

Table 2–1 Diameter Message Command Codes

<table>
<thead>
<tr>
<th>Diameter Message Type</th>
<th>Command Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Control Request (CCR)</td>
<td>272</td>
</tr>
<tr>
<td>Capability Exchange Request (CER)</td>
<td>257</td>
</tr>
<tr>
<td>Device Watchdog Request (DWR)</td>
<td>280</td>
</tr>
</tbody>
</table>

See "Supported Diameter Messages" for more information about these message types.

Table 2–2 Flag Descriptions

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Indicates that the AVP must be supported. This flag is presumed to apply to all AVPs in incoming messages.</td>
</tr>
<tr>
<td>P</td>
<td>Indicates that the AVP must be encrypted for end-to-end security.</td>
</tr>
</tbody>
</table>
Vendor_ID is a value used with AVP_Code to create a vendor-specific Diameter AVP. For all incoming messages, Vendor_ID is presumed to be 0.

- Size is 32 or 64
- Zone is local or utc

Error Handling

The grammar calls the edrAddError function to report errors, such as the failure to create a data block.

These errors are recorded in the Diameter Balance Manager stream log (Pipeline_home/log/stream/InOutputlog_timestamp_transactionID.log). The block in which the error occurred is specified in the log.

Diameter Input Mapping

To transfer data from fields in Diameter messages from the IN to the appropriate fields in EDR containers, the parser follows the mapping in the Diameter input mapping file (Pipeline_home/formatDesc/Formats/Diameter/Diameter_v01_InMap.dsc).

Some AVPs are mapped to different fields in different types of Diameter request messages. For such AVPs, the input mapping file includes multiple subblocks. For example, the AUTH_APPLICATION_ID AVP block has four request-specific subblocks: CC, CE, CE_VSAI.

AUTH_APPLICATION_ID
{
    CC
    {
        AUTH_APPLICATION_ID -> DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.DIAMETER_DETAIL.DIAMETER_CC.INPUT.AUTH_APPLICATION_ID;
    }
    CE
    {
        AUTH_APPLICATION_ID -> DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.DIAMETER_DETAIL.DIAMETER_CE.INPUT.AUTH_APPLICATION_ID;
    }
    CE_VSAI
    {
        AUTH_APPLICATION_ID -> DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.DIAMETER_DETAIL.DIAMETER_CE.INPUT.VENDOR_SPECIFIC_APPLICATION_ID.AUTH_APPLICATION_ID;
    }
}

When processing a Diameter message, the input grammar uses the AVP subblock that corresponds to the response type identified in a message header.

See "Supported Diameter Messages" for information about Diameter messages.
Diameter Stream Format Description

The Diameter stream format description file (`Pipeline_home/formatDesc/Formats/Diameter/Diameter_v01.dsc`) contains descriptions of the fields in Diameter messages. For example, the following entry specifies that the data type of the SESSION_ID field is `DiameterString`, its AVP code is 263, it must be supported per the mandatory flag `M`, and its vendor ID is 0:

```
SESSION_ID DiameterString(263, "M", 0);
```

See "Diameter Field Blocks" for more information about field descriptors.

The input parser uses this file to identify the data type of fields in the responses from the IN. The output parser uses this file to format the request message sent to the IN.

To process Diameter-based network messages, Diameter Balance Manager supports several Diameter data types. For more information, see the discussion of Diameter data types in `BRM Configuring Pipeline Rating and Discounting`.

Diameter Output Grammar

The Diameter output grammar file (`Pipeline_home/formatDesc/Formats/Diameter/Diameter_v01_OutGrammar.dsc`) contains instructions for generating Diameter messages and filling them with data from the EDR containers. At startup, Diameter Balance Manager uses this grammar to create a parser that implements the instructions.

The output grammar file contains a block for each function; only the function blocks associated with the request identified in the message header are incorporated into the parser. Each function block contains the following blocks:

- **Mandatory fields**: The `edrOutputMapToBuffer` function transfers the data in mandatory fields to a buffer. When BRM finishes processing a request, the `outputWrite()` function transfers the data in all such function-block buffers into an outgoing Diameter-based answer message.

- **Optional fields**: Before data in optional fields is transferred to a buffer, the `edrInternalState` function verifies that the fields are included in the output flists of the opcodes that implemented the request or in the protocol-specific data in the `ASSOCIATED_PROTOCOL_INFO` block.

Error Handling

The `logPipeline` function writes all errors to the pipeline log.

Diameter Output Mapping

To transfer data from fields in EDR containers to the appropriate fields in outgoing Diameter messages, the parser follows the mapping in the Diameter output mapping file (`Pipeline_home/formatDesc/Formats/Diameter/Diameter_v01_OutMap.dsc`).

Supported Diameter Messages

By default, Diameter Balance Manager supports these Diameter application messages:

- **Capability Exchange**
  - **Capability Exchange Request**: Initial handshake with the IN. If it succeeds, the IN sends a Capability Exchange Answer (CEA) message with the result-code AVP set to `DIAMETER_SUCCESS` to the Diameter Balance
Manager. If the handshake fails, the result-code AVP is set to the appropriate error code

- **Capability Exchange Answer.** Answer from the IN for the initial handshake.

  ■ **Device Watchdog**
  - **Device Watchdog Request.** Request to check the status of the IN when there is no message exchange between the CM and the IN.
  - **Device Watchdog Answer.** Answer from the IN for the Device Watchdog Request.

  ■ **Credit Control**
  - **Credit Control Request.** Request for a credit control operation.
  - **Credit Control Answer.** Answer to the request for a credit control operation.
This chapter provides an overview of the accounting event data record (EDR) container description for Oracle Communications Billing and Revenue Management (BRM) Diameter Balance Manager.


See "Diameter Balance Manager Protocol Translation" for information about the grammar and mapping files used to translate requests from the Connection Manager (CM) flist format to the Diameter protocol requests and from Diameter protocol to flist format.

See the discussion of EDRs in BRM Configuring Pipeline Rating and Discounting for general information about EDR container descriptions.

### About the Accounting EDR Container Description

Diameter Balance Manager converts requests from the CM into the EDR format that the pipeline modules can process before converting the requests to Diameter protocol for the Intelligent Network (IN) to process. It converts the responses in Diameter protocol from the IN to the EDR format for the pipeline modules to process. The EDR format is defined in the accounting EDR container description, `dmDiameterContainerDesc.dsc`, in the `Pipeline_home/formatDesc/Formats/Portal` directory.

The accounting EDR container description includes the following EDR content types:

- **HEADER.** See "HEADER Content Type for the Accounting EDRs" for more information.
- **DETAIL.** See "DETAIL Content Type for Accounting EDRs" for more information.

### HEADER Content Type for the Accounting EDRs

The HEADER content type for an accounting EDR is defined by the HEADER record in the container description. The information in this record is used by the OUT.Serialize module to write a serialized representation of an accounting EDR to a file in binary format.

The accounting EDR HEADER record contains the fields shown in Table 3–1:
About the Accounting EDR Container Description

The DETAIL content type for accounting EDRs is defined by the DETAIL record in the container description. The information in this record is used to perform the Diameter Balance Manager accounting activities.

The accounting EDR DETAIL record contains these fields and blocks:

- **Function Module Fields**
- **ASSOCIATED_PROTOCOL_INFO Block**

### Function Module Fields

The fields shown in Table 3–2 are required by function modules such as FCT_Timer, FCT_CancelTimer, and the iScripts for setting and getting the correlation data.

#### Table 3–1 Accounting EDR Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATION_PROCESS</td>
<td>String</td>
<td>Specifies the process that created the input file. Must be present for the OUT.Serialize module to process the EDR container. No value is assigned to this field.</td>
</tr>
<tr>
<td>EVENT_TYPE</td>
<td>String</td>
<td>Specifies the event type. Must be present for the OUT.Serialize module to process the EDR container. No value is assigned to this field.</td>
</tr>
</tbody>
</table>

#### DETAIL Content Type for Accounting EDRs

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE_NODE</td>
<td>String</td>
<td>Specifies the fields to map to the input flist of the opcode specified in the OPCODE_NUM field.</td>
</tr>
<tr>
<td>OPCODE_NUM</td>
<td>Integer</td>
<td>Specifies the number of the BRM opcode that performs the requested action. Opcode numbers are defined in header (*.h) files in the BRM_home/include/ops directory.</td>
</tr>
<tr>
<td>OPCODE_FLAG</td>
<td>Integer</td>
<td>Specifies the flag to include in the call to the opcode that performs the requested action.</td>
</tr>
<tr>
<td>SOCKET_ID</td>
<td>Integer</td>
<td>Specifies the CM network socket number to which Diameter Balance Manager connects.</td>
</tr>
<tr>
<td>CURRENT_TIME</td>
<td>Integer</td>
<td>Specifies the time when the EDR container is created. Required by FCT_Timer.</td>
</tr>
<tr>
<td>TIMEOUT_OFFSET</td>
<td>Integer</td>
<td>Number of seconds from the time it is created in which an EDR times out. Required by FCT_Timer.</td>
</tr>
</tbody>
</table>
ASSOCIATED_PROTOCOL_INFO Block

Table 3–3 lists the fields for the ASSOCIATED_PROTOCOL_INFO block.

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASS_DIAMETER_INFO</td>
<td>ASSOCIATED_DIAMETER_INFO</td>
<td>Contains all the attribute-value pairs (AVPs) in a Diameter-based request.</td>
</tr>
<tr>
<td>ASS_FLIST_INFO</td>
<td>ASSOCIATED_FLIST_INFO</td>
<td>Contains all the fields in an flist-based request.</td>
</tr>
</tbody>
</table>

For a sample list of fields in the flist and Diameter protocol blocks, see the `Pipeline_home/formatDesc/Formats/Portal/dmDiameterContainerDesc.dsc` file.

ASS_DIAMETER_INFO

This block contains all the fields in the following types of Diameter messages from the IN:

- Credit Control
- Capability Exchange
- Device Watchdog

Note: The INTERNAL fields block at the end of this block are BRM internal fields that you must not change.

ASS_FLIST_INFO

This block contains the input and output flist fields, which you can customize, for the following supported opcodes:

- PCM_OP_CREATE_OBJ
- PCM_OP_DELETE_OBJ
- PCM_OP_TEST_LOOPBACK

There is a block each for the input and output flist of each opcode.
Note: The fields at the beginning of this block definition, up to the FLIST_DETAIL field, are BRM internal fields that you must not change.