Copyright

Copyright © 2013, 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, 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 on 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. 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.
Installation and Removal Overview
Checking the Installation

NCC Glossary of Terms
Index
About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the Diameter Charging Driver application.

Audience

This guide was written primarily for system administrators and persons installing, configuring and administering the Diameter Charging Driver application. However, sections of the document may be useful to anyone requiring an introduction to the application.

Prerequisites

A solid understanding of UNIX and a familiarity with IN concepts are an essential pre-requisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

A familiarity with the Diameter protocol is also required. Refer to the following:

- RFC 3588 - Diameter Base Protocol
- RFC 4006 - Diameter Credit-Control Application

Although it is not a prerequisite to using this guide, familiarity with the target platform would be an advantage.

This manual describes system tasks that should only be carried out by suitably trained operators.

Related documents

The following documents are related to this document:

- Oracle Communications Network Charging and Control Advanced Control Services Technical Guide
- Oracle Communications Network Charging and Control Charging Control Services Technical Guide
- Oracle Communications Network Charging and Control Charging Control Services User's Guide
- Oracle Communications Network Charging and Control Diameter Charging Driver Alarms Guide
- Oracle Communications Network Charging and Control Service Management System Technical Guide
- Oracle Communications Network Charging and Control Service Management System User's Guide
- Oracle Communications Network Charging and Control Service Logic Execution Environment Technical Guide
## Document Conventions

### Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
</table>
| **Special Bold**      | Items you must select, such as names of tabs.  
                        | Names of database tables and fields.          |
| **Italics**           | Name of a document, chapter, topic or other publication.  
                        | Emphasis within text.                         |
| **Button**            | The name of a button to click or a key to press.  
                        | **Example:** To close the window, either click **Close**, or press **Esc**. |
| **Key+Key**           | Key combinations for which the user must press and hold down one key and then press another.  
                        | **Example:** **Ctrl+P**, or **Alt+F4**.        |
| **Monospace**         | Examples of code or standard output.            |
| **Monospace Bold**    | Text that you must enter.                       |
| **variable**          | Used to indicate variables or text that should be replaced. |
| **menu option > menu option >** | Used to indicate the cascading menu option to be selected, or the location path of a file.  
                        | **Example:** **Operator Functions > Report Functions**  
                        | **Example:** `/IN/html_SMS/Helptext/` |
| **hypertext link**    | Used to indicate a hypertext link on an HTML page. |

Specialized terms and acronyms are defined in the **Glossary** at the end of this guide.
Overview

Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

In this chapter

This chapter contains the following topics.

What is the Diameter Charging Driver

What is the Diameter Charging Driver

Overview

Diameter is a protocol that focuses on network access and accounting. The Diameter base protocol provides the minimum requirements needed for Authentication, Authorization and Accounting (AAA) (RFC 3588).

This base protocol can be extended via the addition of new commands or AVPs. RFC 4006 specifies such an extension for applications that can be used to implement real-time credit-control.

The Diameter Charging Driver (DCD) product provides functionality that allows the Prepaid Charging product to interface with applications using the RFC 3588 and RFC 4006 protocol. Typically it is expected that Prepaid Charging will interface with a third-party convergent real-time charging system.

The DCD product contains several components:

- Diameter protocol stack. Implements the RFC 3588/4006 protocol
- Dynamically loadable library (DLL), diamActions.so. Implements the required Prepaid Charging functionality
- Diameter client. Implements the network interface to the Diameter

Diameter Credit Control

The Prepaid Charging product uses the Universal-Attempt-Termination-with-Billing (UATB) node for credit-control of telephony (voice) calls. There are a number of other CCS nodes that also use Diameter credit control actions.

RFC 4006 defines credit-control in the following way:

- Credit-control is a process of checking whether credit is available, credit-reservation, deduction of credit from the end user account when service is completed and refunding of reserved credit that is not used.
The Diameter terminology defines an “interrogation” as the request/answer transaction between the client and server.

RFC 4006 defines session based credit-control as:

- A credit-control process that makes use of several interrogations:
  - the first - used to reserve money from the user's account and to initiate the process
  - a possible intermediate - may be needed to request new quota while the service is being rendered.
  - the final - used to exit the process.

The credit-control server is required to maintain session state for session-based credit-control.

Telephony requires session based credit-control. A new session is created when the CCS product detects that an end-user is trying to establish a new telephony call.

Other nodes may use the DCD to send event based (rather than session based) credit control messages for one-time events, for example, SMS (text message).

**Process**

Prepaid Charging uses the dcdBeClient (Diameter Charging Driver) to send a first interrogation to the Diameter Server. The server rates the request, reserves a suitable amount of money from the user’s account, and returns the corresponding amount of credit resources. Prepaid Charging connects the telephony call and monitors the usage of the granted resources.

Prepaid Charging may send an intermediate interrogation to request a new quota of resources when the granted resources have been consumed. When the telephony call ends, Prepaid Charging sends a final interrogation to inform the Diameter Server of the actual amount of resources used. At this point the session is terminated.

**Credit Control messages**

RFC 4006 defines two new commands used for credit-control encapsulated in the following messages:

- Credit-Control-Request (CCR). Used by the credit-control client to request credit authorization from the credit-control server.
- Credit-Control-Answer (CCA). Used by the credit-control server to acknowledge a CCR from the credit-control client.

**AVPs**

A detailed list of AVPs for the CCR and CCA messages is given in RFC 4006 and copied in the next section of this document. Note the CC-Request-Type - an enumeration with the following values:

- INITIAL_REQUEST - First interrogation
- UPDATE_REQUEST - Intermediate interrogation
- TERMINATION_REQUEST - Final interrogation
- EVENT_REQUEST - Event based (not session based)

**Note:** DCD can be configured to support certain vendor specific applications that add AVPs to the accounting commands of Diameter base protocol. For more information, refer to vendor specific AVPs under **DCD parameters** (on page 36).

**Attribute Value Pairs**

In the Diameter protocol message, parameters are specified as Attribute-Value Pairs (AVPs).

An AVP consists of a Code, Flags, Length, optional Vendor-ID and Data fields. The AVP Code, combined with the Vendor-Id field, identifies the attribute uniquely. The type (format) of the Data field is implied by the Code/Vendor-Id field combination. The following Data formats are specified:
- OctetString
- Integer32
- Integer64
- Unsigned32
- Unsigned64
- Float32
- Float64
- Address
- Time
- UTF8String
- DiameterIdentity
- DiameterURI
- Enumerated
- Grouped

**Additional EDR tags**

Resolved values for AVPs can be written to the ACS EDR under a configured tag. These tags are not intended to be used to amend existing, pre-defined ACS tags.

The feature is intended for situations where the customer wishes to add some new tag to the EDR. Conditions can be attached to the writing of the EDR value:

- Replace it unconditionally, after removing any existing tags of same name.
- Append a new value instance unconditionally.
- Leave the EDR alone if the tag is present, append the new instance if the tag is not present.
Diagram

Here is a high level diagram showing the Diameter Charging Driver.

ccsConcepts

List of ccsConcepts

Here is the list of all of the ccsConcepts that can be used in the AVPs section of the eserv.config file.

Introduction

To match AVPs to variables in CCS, the DCD has “ccsConcepts”. These can be a specific parameter of the CCS action, a general CCS variable, some of the call's context, or even an ACS profile value.

The DCD provides functionality to scale values by a factor, and also allows a mapping of one set of integers to another while reading/writing to CCS concepts. The specific formatting of the value field is quite configurable, see AVPs parameters for formatting details.

Note: The availability of each concept depends upon the action involved, and the previous actions of the control plan.
**ACS Action handler**
Here are the CCS concepts from the ACS action handler.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>cascade</td>
<td>After a setCascade Override.</td>
<td>Also can be set by previous responses. The integer ID of the cascade to apply.</td>
</tr>
<tr>
<td>chargeInfoBalanceSystemValue</td>
<td>After a chargeInfo response</td>
<td>The balance Unit for the current item of the Charge structure. This is in units of the system currency.</td>
</tr>
<tr>
<td>chargeInfoBalanceType</td>
<td>After a chargeInfo response</td>
<td>The CCS ID of the balance Type for the current balance of the Charge structure.</td>
</tr>
<tr>
<td>chargeInfoBalanceUnitType</td>
<td>After a chargeInfo response</td>
<td>The CCS of the balance Unit for the current balance of the Charge structure. This is not necessary for a ChargeInfo in a response, it can be derived from the balance type.</td>
</tr>
<tr>
<td>chargeInfoBalanceUserValue</td>
<td>After a chargeInfo response</td>
<td>The balance Unit for the current item of the Charge structure. This is in units of the user's currency. Note that the system currency value is mandatory, while this entry is optional.</td>
</tr>
<tr>
<td>discountMaxCharge</td>
<td>After a setDiscount</td>
<td>Present after a setDiscount node or a response that has the discountMaxCharge present.</td>
</tr>
<tr>
<td>discountPeriod</td>
<td>After a setDiscount</td>
<td>Present after a setDiscount node or a response that has the discountPeriod present.</td>
</tr>
<tr>
<td>tariffCugName</td>
<td>After a setTariffPlan</td>
<td>The Closed User Group Name.</td>
</tr>
<tr>
<td>tariffPlan</td>
<td>After a setTariffPlan</td>
<td>Integer representing the tariff Plan.</td>
</tr>
<tr>
<td>terminationCause</td>
<td>After a call is terminated</td>
<td>The esg values in the configuration for the ACS callEndReasons that map to specific termination cause values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = reasonNotSet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = precallAnnouncementFailure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = firstEventACRAbort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = firstEventATAbort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 = secondEventACRAbort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 = secondEventATAbort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 = abortWaitingForBEResponse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 = releasedOnTCPExpiry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 = releasedNoFunds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 = disconnectedLegBNoFunds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = calledPartyBusy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 = routeSelectFailure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 = callingPartyAbandon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 = noAnswer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 = callingPartyDisconnected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 = calledPartyDisconnected</td>
</tr>
<tr>
<td>Concept Label</td>
<td>Available</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>walletInfoActivationDate</td>
<td>After a walletInfo response</td>
<td>time_t of the wallet's activation date. The DCD handles conversion from time_t to DIAMETER times.</td>
</tr>
<tr>
<td>walletInfoBalanceExpiry</td>
<td>After a walletInfo response</td>
<td>The expiry date (in time_t) of the current balance. The DCD handles conversion from time_t to DIAMETER times.</td>
</tr>
<tr>
<td>walletInfoBalanceExponent</td>
<td>After a walletInfo response</td>
<td>An exponent to apply to the balance system value.</td>
</tr>
<tr>
<td>walletInfoBalanceLimitType</td>
<td>After a walletInfo response</td>
<td>The balances limit type: An integer representing one of: limitedPostpaid, postpaid, prepaid, singleUsePrepaid</td>
</tr>
<tr>
<td>walletInfoBalanceMaxCredit</td>
<td>After a walletInfo response</td>
<td>The maximum amount of credit allowed for this subscriber.</td>
</tr>
<tr>
<td>walletInfoBalanceSystemValue</td>
<td>After a walletInfo response</td>
<td>The balance Unit for the current item of the balance structure. This is in units of the system currency.</td>
</tr>
<tr>
<td>walletInfoBalanceType</td>
<td>After a walletInfo response</td>
<td>The CCS ID of the balance Type for the current balance of the Wallet structure.</td>
</tr>
<tr>
<td>walletInfoBalanceUnitType</td>
<td>After a walletInfo response</td>
<td>The CCS ID of the balance Unit for the current balance item of the Wallet structure. This is not necessary for a balance in a response, it can be derived from the balance type.</td>
</tr>
<tr>
<td>walletInfoBalanceUserValue</td>
<td>After a walletInfo response</td>
<td>The balance Unit for the current item of the Blance structure. This is in units of the user's currency. Note that the system currency value is mandatory, while this entry is optional.</td>
</tr>
<tr>
<td>walletInfoExpiry</td>
<td>After a walletInfo response</td>
<td>The expiry date (in time_t) of the wallet. The DCD handles conversion from time_t to DIAMETER times.</td>
</tr>
<tr>
<td>walletInfoLastAccess</td>
<td>After a walletInfo response</td>
<td>time_t of the wallet's last access. The DCD handles conversion from time_t to DIAMETER times.</td>
</tr>
<tr>
<td>walletInfoMaxConcurrent</td>
<td>After a walletInfo response</td>
<td>The maximum number of concurrent users allowed for this wallet.</td>
</tr>
<tr>
<td>walletInfoState</td>
<td>After a walletInfo response</td>
<td>A single character representing the wallet's state. One of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'A' = Active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'D' = Dormant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'F' = Frozen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'P' = Pre-Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'S' = Suspended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'T' = Terminated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that conversion to different representations is possible.</td>
</tr>
<tr>
<td>walletInfoSystemCurrency</td>
<td>After a walletInfo response</td>
<td>The system currency.</td>
</tr>
<tr>
<td>walletInfoUserCurrency</td>
<td>After a walletInfo response</td>
<td>The CCS_ACCT.CURRENCY value for</td>
</tr>
</tbody>
</table>
## ACS service context
Here are the CCS concepts from the ACS service context.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>acsCallID</td>
<td>always</td>
<td>The call ID from the SLEE</td>
</tr>
<tr>
<td>acsChargingDomain</td>
<td>always</td>
<td>The destined billing domain (logical collection of wallets) for this request.</td>
</tr>
<tr>
<td>acsProductType</td>
<td>always</td>
<td>The ACS product type ID</td>
</tr>
<tr>
<td>acsProfile</td>
<td>always</td>
<td>An ACS profile buffer from the Call plan. If the buffer is not set, then the AVP is not included.</td>
</tr>
<tr>
<td>acsServiceProvider</td>
<td>always</td>
<td>The ACS service provider ID</td>
</tr>
<tr>
<td>acsSubscriber</td>
<td>always</td>
<td>The CCS subscriber ID</td>
</tr>
<tr>
<td>acsSubscriberReference</td>
<td>always</td>
<td>The CCS subscriber number (i.e., their MSISDN)</td>
</tr>
<tr>
<td>acsTariffCode</td>
<td>After an initial reservation.</td>
<td>Tariff Code string returned in the Initial Reservation Response (if present).</td>
</tr>
<tr>
<td>acsUnnormalisedCalledNumber</td>
<td>always</td>
<td>The called party number digits from the IDP, without any attempt at normalization.</td>
</tr>
<tr>
<td>acsWallet</td>
<td>always</td>
<td>The CCS wallet ID (BE_WALLET.ID)</td>
</tr>
<tr>
<td>acsWalletReference</td>
<td>always</td>
<td>The CCS wallet Reference (the Billing System's reference to the wallet)</td>
</tr>
<tr>
<td>acsWalletType</td>
<td>always</td>
<td>The CCS wallet type. (CCS_WALLET_TYPE.ID)</td>
</tr>
</tbody>
</table>

## CCS time reservation
Here are the CCS concepts from CCS time reservation.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>callAnsweredTime</td>
<td>ConfirmTimeReservation</td>
<td></td>
</tr>
<tr>
<td>callDurationDelta</td>
<td>Any Time Charging Action</td>
<td></td>
</tr>
<tr>
<td>callDurationTotal</td>
<td>Any Time Charging Action</td>
<td></td>
</tr>
<tr>
<td>callerTimeZone</td>
<td>After a DirectTimeCharge or InitialTimeReservation</td>
<td></td>
</tr>
<tr>
<td>cli</td>
<td>After a DirectTimeCharge or InitialTimeReservation</td>
<td></td>
</tr>
<tr>
<td>confirmTimeReservationStatus</td>
<td>After set from a response</td>
<td>Usually part of an confirmTimeReservationResponse.</td>
</tr>
<tr>
<td>destinationNumber</td>
<td>After a</td>
<td></td>
</tr>
<tr>
<td>Concept Label</td>
<td>Available</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>discountPercentage</td>
<td>DirectTimeCharge or InitialTimeReservation After a setDiscount or DirectNamedEvent or NamedEventReservation</td>
<td>Present after a setDiscount node or a response that has the discountPeriod present.</td>
</tr>
<tr>
<td>eventClass</td>
<td>NamedEvent Actions</td>
<td>A string representing the CCS event Class.</td>
</tr>
<tr>
<td>eventName</td>
<td>NamedEvent Actions</td>
<td>A string of the CCS event name.</td>
</tr>
<tr>
<td>eventType</td>
<td>NamedEvent Actions</td>
<td>An integer representing the type of CCS named event.</td>
</tr>
<tr>
<td>expectedReservationDelta</td>
<td>InitialTimeReservation and ExtendTimeReservation</td>
<td></td>
</tr>
<tr>
<td>expectedReservationTotal</td>
<td>InitialTimeReservation and ExtendTimeReservation</td>
<td></td>
</tr>
<tr>
<td>extraInformation</td>
<td></td>
<td>Usually call information for adding to Billing CDRs. Content varies for each action.</td>
</tr>
<tr>
<td>freeCallDisposition</td>
<td>After set from a response</td>
<td>Usually part of an InitialTimeReservationResponse.</td>
</tr>
<tr>
<td>ignoreBalanceLimit</td>
<td>DirectNamedEvent, DirectTimeCharge, NamedEventReservation</td>
<td></td>
</tr>
<tr>
<td>initialLowBalanceAnnouncement</td>
<td>After set from a response</td>
<td>Usually part of an InitialTimeReservationResponse. The Announcement ID of the announcement to play.</td>
</tr>
<tr>
<td>initialLowBalanceIndicator</td>
<td>After set from a response</td>
<td>Usually part of an InitialTimeReservationResponse. If present and non zero the indicated pre call warning announcement should be played to the subscriber.</td>
</tr>
<tr>
<td>lowCreditBuffer</td>
<td>After set from a response</td>
<td>Usually part of an InitialTimeReservationResponse. Number of seconds from the end of the last good reservation period until a low credit beep should be played</td>
</tr>
<tr>
<td>maxCallLength</td>
<td>After set from a response</td>
<td>Usually part of an InitialTimeReservationResponse.</td>
</tr>
<tr>
<td>maxSeconds</td>
<td>After set from a response</td>
<td>Session Time left. Usually part of an xxxTimeReservationResponse.</td>
</tr>
<tr>
<td>maxUnitsRequested</td>
<td>NamedEvent Actions</td>
<td></td>
</tr>
<tr>
<td>minUnitsRequested</td>
<td>NamedEvent Actions</td>
<td></td>
</tr>
<tr>
<td>numUnitsGranted</td>
<td>After set from a response</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 1

#### System Overview

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>numUnitsUsed</td>
<td>ConfirmNamedEventReservation</td>
<td></td>
</tr>
<tr>
<td>reservedLengthDelta</td>
<td>After set from a response</td>
<td>Usually part of an xxxTimeReservationResponse.</td>
</tr>
<tr>
<td>reservedLengthTotal</td>
<td>After set from a response</td>
<td>Usually part of an xxxTimeReservationResponse.</td>
</tr>
<tr>
<td>retrieveLCRNumbers</td>
<td>After set from a response</td>
<td>Usually part of an initialTimeReservationResponse.</td>
</tr>
<tr>
<td>revokeTimeReservationStatus</td>
<td>After set from a response</td>
<td>Usually part of an revokeTimeReservationResponse.</td>
</tr>
<tr>
<td>scpAction</td>
<td></td>
<td>This AVP is an enumeration with the following known values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 Supervise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 Do not supervise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3 Release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4 Send message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5 Play announcement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6 Supervise without controlling</td>
</tr>
<tr>
<td>singleReservation</td>
<td>After set from a response</td>
<td>Usually part of an initialTimeReservationResponse.</td>
</tr>
<tr>
<td>timeReservationStatus</td>
<td>After set from a response</td>
<td>Usually part of an xxxTimeReservationResponse.</td>
</tr>
<tr>
<td>validityPeriod</td>
<td>After set from a response</td>
<td></td>
</tr>
</tbody>
</table>

#### Charge details

Here are the CCS concepts from charge details.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>balanceTypeFilter</td>
<td>WalletInfo</td>
<td>Request the Billing Engine to only return balances of this type.</td>
</tr>
<tr>
<td>balanceUnitFilter</td>
<td>WalletInfo</td>
<td>Request the Billing Engine to only return balances of this unit.</td>
</tr>
</tbody>
</table>

#### Direct time charge

Here are the CCS concepts from direct time charge.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>callDate</td>
<td>DirectTimeCharge</td>
<td></td>
</tr>
<tr>
<td>ratingPrecision</td>
<td>InitialTimeReservation</td>
<td>Integer representing one of seconds, tenths-of-a-second or hundredths-of-a-second</td>
</tr>
</tbody>
</table>
### Others
Here are the CCS concepts from others.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>freeform</td>
<td>always</td>
<td>Uses/updates the concept previously defined by setFreeform.</td>
</tr>
<tr>
<td>setFreeform</td>
<td>always</td>
<td>The next AVP of concept “freeform” will instead use/update the concept indexed by the value of this AVP.</td>
</tr>
</tbody>
</table>

### Voucher details
Here are the CCS concepts from voucher details.

<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>voucherInfoBalanceExpiryExtension</td>
<td>WalletInfoRequest</td>
<td>The expiry extension period for adjusting the balance expiry date of the voucher.</td>
</tr>
<tr>
<td>voucherInfoBalanceExpiryExtensionPolicy</td>
<td>WalletInfoRequest</td>
<td>Indicates how to apply the balance expiry extension period to the balance expiry date.</td>
</tr>
<tr>
<td>voucherInfoBalanceExpiryExtensionType</td>
<td>WalletInfoRequest</td>
<td>The unit of the extension value available for this balance (example: hours or months).</td>
</tr>
<tr>
<td>voucherInfoBalanceType</td>
<td>WalletInfoRequest</td>
<td>The CCS ID of the balance type for the current balance of the voucher structure.</td>
</tr>
<tr>
<td>voucherInfoMissingBalancePolicy</td>
<td>WalletInfoRequest</td>
<td>Indicates what to do if the specified balance type is missing from the list of existing balances for the voucher.</td>
</tr>
<tr>
<td>voucherInfoNewBucket</td>
<td>WalletInfoRequest</td>
<td>If this value is set to true, the voucher value will be added to the balance as a new bucket.</td>
</tr>
<tr>
<td>voucherInfoReplaceBalance</td>
<td>WalletInfoRequest</td>
<td>If this value is set to true, all existing buckets of the balance will removed, and a new bucket is created with the specified voucher value.</td>
</tr>
<tr>
<td>voucherInfoValue</td>
<td>WalletInfoRequest</td>
<td>The voucher balance recharge details.</td>
</tr>
<tr>
<td>voucherInfoVoucher</td>
<td>WalletInfoRequest</td>
<td>The database key of the voucher being redeemed.</td>
</tr>
<tr>
<td>voucherInfoVoucherNumber</td>
<td>WalletInfoRequest</td>
<td>The voucher number of the voucher being redeemed.</td>
</tr>
<tr>
<td>voucherInfoWalletExpiryExtension</td>
<td>WalletInfoRequest</td>
<td>The extension period to apply to the wallet expiry date of the recharged wallet.</td>
</tr>
<tr>
<td>voucherInfoWalletExpiryExtensionPolicy</td>
<td>WalletInfoRequest</td>
<td>Indicates how to apply the wallet expiry extension period to the wallet expiry date.</td>
</tr>
<tr>
<td>voucherInfoWalletExpiryExtensionType</td>
<td>WalletInfoRequest</td>
<td>The unit of the expiry extension for the wallet that the voucher will recharge (example: hours or months).</td>
</tr>
<tr>
<td>voucherTypeName</td>
<td>WalletInfoRequest</td>
<td>Returns the name of the type of voucher being redeemed.</td>
</tr>
</tbody>
</table>

**Note:** Voucher type name is only available if a positive value is defined for `voucherTypeCacheSize` in the
<table>
<thead>
<tr>
<th>Concept Label</th>
<th>Available</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ccsActions section. When this cache is configured, CCS will be able to use the batch of the voucher being redeemed to lookup the voucher type, and that name will then be available to the ccsConcept voucherTypeName.</td>
</tr>
<tr>
<td>voucherRechargeFailureDateTime</td>
<td>WalletRechargeRequest</td>
<td>Returns the timestamp of any previous voucher recharge failure. If there has not been a previous voucher recharge failure, then zero (0) is returned.</td>
</tr>
<tr>
<td>voucherRechargeFailureFlag</td>
<td>WalletRechargeRequest</td>
<td>Returns the value of one (1) if the voucher is not redeemed and a failed voucher redeem attempt has been made. Returns zero (0) for all other voucher states. For example, if a redeem attempt has never been made for the voucher or if the voucher has been redeemed successfully.</td>
</tr>
</tbody>
</table>
Chapter 2

Split Charging and Voucher Domains

Overview

Introduction

This chapter explains how the DCD components handle split charging and voucher redemption when wallets and vouchers are hosted by different billing domains.

In this chapter

This chapter contains the following topics.

| Wallets and Vouchers Split Recharging | 13 |
| Bad PIN | 16 |

WALLETS & VOUCHERS SPLIT RECHARGING

Introduction

CCS supports charging services for redeeming vouchers and updating wallets when they reside on the same billing domain. The DCD product provides functionality that allows the Prepaid Charging product to extend this support to separate voucher and wallet domains in CCS.

DCD can be configured to support a CCS-based convergent billing solution where separate billing systems are used to host:

- Vouchers and redemption functions
  **Example:** Oracle VWS-Voucher Management
- Wallets and charging functions
  **Example:** A third-party billing system.

In order to achieve this, DCD can be extended using ccsConcepts required to support voucher redemption.

Key components

The key components that enable this split-domain architecture with DCD are as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS voucherDelegator</td>
<td>Configures the diamActions which support different billing domains for recharges.</td>
<td>voucherDelegator (on page 14)</td>
</tr>
<tr>
<td>ccsConcepts</td>
<td>Support voucher redemption variables.</td>
<td>Voucher details</td>
</tr>
</tbody>
</table>
voucherDelegator

The voucherDelegator is a slee_acs process which implements the following diamActions enabling split domain recharging over DCD:

<table>
<thead>
<tr>
<th>Action</th>
<th>Expected Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPIN</td>
<td>• CCR named BadPINRequest, and</td>
<td>Sends a CCR to the BE, notifying that a given MSISDN has failed to redeem a</td>
</tr>
<tr>
<td></td>
<td>• CCA named BadPINResponse</td>
<td>voucher.</td>
</tr>
<tr>
<td>CreateEDR</td>
<td>• CCR named CreateEDRRequest, and</td>
<td>Sends a list of tags and values as AVPs to the third-party BE which will be</td>
</tr>
<tr>
<td></td>
<td>• CCA named CreateEDRResponse</td>
<td>added to the BE EDR.</td>
</tr>
<tr>
<td>WalletRecharge</td>
<td>• CCR named WalletRechargeRequest, and</td>
<td>Sends a CCR to the third-party BE with a wallet recharge request, and expects</td>
</tr>
<tr>
<td></td>
<td>• CCA named WalletRechargeResponse</td>
<td>a CCA with a wallet recharge response.</td>
</tr>
</tbody>
</table>
Diagram

Here is an example of the split-domain wallet and voucher recharging process. Note that the third-party billing engine server is also the Diameter Server in this example.

Split recharging process

This table describes the stages involved in redeeming a voucher using VWS-Voucher Management and recharging a wallet on a third-party domain.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Voucher redemption is triggered using any of the following methods:  
  - IVR feature nodes in a control plan  
  - Interaction with a customer services representative (who uses the Voucher Management screen)  
  - (if MM is installed) Short Messages sent from the subscriber’s handset, and  
  - (if USSD GW is installed) menus and fast access.  
  The information from the request initiator is passed to the voucherDelegate, which |
<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The BeClientIF sends a Voucher Reserve (VR_Req) request to VWS-Voucher Management.</td>
</tr>
<tr>
<td>3</td>
<td>VWS-Voucher Management checks whether:</td>
</tr>
<tr>
<td></td>
<td>• this VWS holds the details for the requested voucher</td>
</tr>
<tr>
<td></td>
<td>• the voucher PIN number is correct, and</td>
</tr>
<tr>
<td></td>
<td>• if the voucher can be redeemed.</td>
</tr>
<tr>
<td></td>
<td>If the voucher can be redeemed, VWS-Voucher Management reserves the voucher and passes a Voucher Reserve acknowledgment (VR_Ack) back to the voucherDelegator.</td>
</tr>
<tr>
<td>4</td>
<td>The voucherDelegator processes the message and sends a Wallet Recharge request to the dcdBeClientIF in an attempt to recharge the wallet on the Diameter Server (in this case the third-party BE).</td>
</tr>
<tr>
<td>5</td>
<td>The dcdBeClientIF constructs a CCR with a WalletRecharge action and interrogates the Diameter Server for wallet recharge.</td>
</tr>
<tr>
<td>6</td>
<td>The Diameter Server checks whether:</td>
</tr>
<tr>
<td></td>
<td>• the details for the requested wallet, and</td>
</tr>
<tr>
<td></td>
<td>• whether the wallet state allows it to be updated.</td>
</tr>
<tr>
<td></td>
<td>If the wallet can be recharged, the Diameter Server sends a CCA response back to the dcdBeClientIF with a Wallet Recharge acknowledgment, which is reported back to the voucherDelegator.</td>
</tr>
<tr>
<td>7</td>
<td>The voucherDelegator then sends a Commit Voucher Redeem (CVR_Req) request to the BeClientIF which is sent to VWS-Voucher Management for redeeming the voucher.</td>
</tr>
<tr>
<td></td>
<td>If the voucher redeem succeeds, VWS-Voucher Management responds with a Commit Voucher Redeem acknowledgement (CVR_Ack) to the BeClientIF which is reported back to the voucherDelegator.</td>
</tr>
<tr>
<td>8</td>
<td>The voucherDelegator processes the message and informs the request initiator of the successful voucher redemption wallet recharge.</td>
</tr>
<tr>
<td>9</td>
<td>The voucherDelegator then initiates the createEDR action for the relevant EDRs to be produced on the Diameter Server.</td>
</tr>
<tr>
<td>10</td>
<td>If either the voucher redemption or the wallet recharge failed, appropriate Not Acknowledgment (Nack) messages at each stage and this is recorded in the final set of EDRs generated.</td>
</tr>
</tbody>
</table>

**Bad PIN**

**Description**

The ACS voucherDelegator also handles BadPIN processing and notifies the BE that controls wallet recharging if a subscriber (MSISDN) has failed to redeem a voucher.
Diagram

Here is an example of Bad PIN processing on a split-domain wallet and voucher network. Note that the third-party billing engine server is also the Diameter Server in this example.

Checking voucher bad PIN

This table describes an unsuccessful voucher redemption due to a Bad PIN.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | When VWS-Voucher Management receives a Voucher Reserve (VR_Req) request, it checks whether:  
\- the VWS holds the details for the requested voucher  
\- the voucher PIN number is correct, and  
\- if the voucher can be redeemed.  
If the voucher PIN is incorrect, VWS-Voucher Management updates the bad PIN counter and returns a Bad PIN Increase (BPIN) count to the BeClientIF which informs the voucherDelegator. |
<p>| 2     | On successive Bad PIN attempts, the voucherDelegator cancels the voucher |</p>
<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VWS-Voucher Management responds with a Revoke Voucher Redeem (RVR_Ack) acknowledgement which means that the voucher redemption request stands cancelled.</td>
</tr>
<tr>
<td>2</td>
<td>The voucherDelegator then sends a Bad PIN request to the dcdBeClientIF in an attempt to cease any transactions on the wallet domain.</td>
</tr>
<tr>
<td>3</td>
<td>The dcdBeClientIF constructs a CCR with a BadPINRequest action and notifies the Diameter Server.</td>
</tr>
<tr>
<td>4</td>
<td>The Diameter Server confirms and sends a CCA back to the dcdBeClientIF with a BadPINResponse, which is reported to the voucherDelegator.</td>
</tr>
<tr>
<td>5</td>
<td>The voucherDelegator processes the message and informs the request initiator that the voucher redemption was unsuccessful.</td>
</tr>
<tr>
<td>6</td>
<td>The voucherDelegator then initiates the createEDR action for the relevant EDRs to be produced on the Diameter Server.</td>
</tr>
</tbody>
</table>
Overview

Introduction

This chapter explains the summary of changes that affect DCD to make it SCAP compliant.

In this chapter

This chapter contains the following topics.

SCAP

SCAP

Introduction

Ericsson SCAP is a vendor specific protocol that utilizes the Draft version 8 of the Diameter Base Protocol (the predecessor to RFC 3588).

The NCC Diameter protocol is based upon the final version of RFC 3588. The base protocol is further extended to support the use of CCR (Credit-Control-Request) / CCA (Credit-Control-Answer) messages, described by RFC 4006.

SCAP, on the other hand, adds Attribute Value Pair (AVP)s to Diameter base protocol's Accounting-Request (ACR) and Accounting-Answer (ACA) commands.

DCD may be reconfigured to permit it to interact with an Ericsson SCAP compliant server as a SCAP client. Variances between the SCAP and normal Credit-Control charging approaches available within DCD are described below.

Note: The necessary changes will take effect only if the enableDraft8 (on page 38) is set to true.

Application-identifiers values

This table lists the different Application-identifier values between the Diameter Base Protocol Draft 8 and RFC 3588 final version.

<table>
<thead>
<tr>
<th>Draft 8</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Diameter Common Messages 0</td>
</tr>
<tr>
<td>NASREQ 1</td>
<td>NASREQ 1</td>
</tr>
<tr>
<td>CMS Security 2</td>
<td>Mobile-IP 2</td>
</tr>
<tr>
<td>Mobile IP 4</td>
<td>-</td>
</tr>
<tr>
<td>Relay 0xffffffff</td>
<td>Relay 0xffffffff</td>
</tr>
</tbody>
</table>
Message header

This table describes the changes to message header values.

<table>
<thead>
<tr>
<th>Section Heading</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor-Id</td>
<td>This should be changed in outbound messages (at the Oracle Diameter stack level). See Vendor-Id (on page 42) for SCAP specific changes.</td>
</tr>
<tr>
<td>T-flag</td>
<td>The T-flag (retransmit) is used by a Diameter client to indicate to the server that a message has been retransmitted (possibly due to loss of message). This flag is not supported by Draft 8, and hence it is not set while retransmitting duplicate DCD messages to SCAP.</td>
</tr>
</tbody>
</table>

Supported AVPs

This table describes the changes to the supported AVPs to enable SCAP compliance.

<table>
<thead>
<tr>
<th>Supported AVP</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event-Timestamp</td>
<td>DCD to send vendor specific version for Ericsson SCAP.</td>
</tr>
<tr>
<td>Inband-Security-Id</td>
<td>This AVP is not sent during capabilities exchange, if SCAP support is configured, that is, when enableDraft8 (on page 38) is set to true.</td>
</tr>
</tbody>
</table>

Result-Codes

This table describes the changes to the Result-Codes to enable SCAP compliance.

<table>
<thead>
<tr>
<th>Result Code</th>
<th>[DRAFT8] Value</th>
<th>[3588] Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAMETER_UNSUPPORTED_TRANSFORM</td>
<td>5010</td>
<td>-</td>
</tr>
<tr>
<td>DIAMETER_NO_COMMON_APPLICATION</td>
<td>5011</td>
<td>5010</td>
</tr>
<tr>
<td>DIAMETER_UNSUPPORTED_VERSION</td>
<td>5012</td>
<td>5011</td>
</tr>
<tr>
<td>DIAMETER_UNABLE_TO_COMPLY</td>
<td>5013</td>
<td>5012</td>
</tr>
<tr>
<td>INVALID_BIT_IN_HEADER</td>
<td>5014</td>
<td>5013</td>
</tr>
<tr>
<td>INVALID_AVP_LENGTH</td>
<td>5015</td>
<td>5014</td>
</tr>
<tr>
<td>INVALID_MESSAGE_LENGTH</td>
<td>5016</td>
<td>5015</td>
</tr>
<tr>
<td>INVALID_AVP_BIT_COMBO</td>
<td>5017</td>
<td>5016</td>
</tr>
<tr>
<td>DIAMETER_NO_COMMON_SECURITY</td>
<td>-</td>
<td>5017</td>
</tr>
</tbody>
</table>
Chapter 4

Configuration

Overview

Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Configuration Overview</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>eserv.config Configuration</td>
<td>22</td>
</tr>
<tr>
<td>CCS eserv.config Configuration</td>
<td>23</td>
</tr>
<tr>
<td>SLEE.cfg Configuration</td>
<td>24</td>
</tr>
<tr>
<td>acs.conf Configuration</td>
<td>24</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction

This topic provides a high level overview of how the DCD interface is configured.

There are configuration options which are added to the configuration files that are not explained in this chapter. These configuration options are required by the application and should not be changed.

Configuration components

The Diameter Charging Driver is configured by the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>eserv.config</td>
<td>all SLC machines</td>
<td>DCD is configured by the Diameter section of eserv.config.</td>
<td>eserv.config Configuration (on page 22).</td>
</tr>
<tr>
<td>eserv.config</td>
<td>all SLC machines</td>
<td>The WalletInformation cache is configured in the CCS section of eserv.config.</td>
<td>CCS eserv.config Configuration (on page 23).</td>
</tr>
<tr>
<td>SLEE.cfg</td>
<td>all SLC machines</td>
<td>The SLEE interface is configured to include the DCD service.</td>
<td>SLEE.cfg Configuration (on page 24) and the SLEE Technical Guide.</td>
</tr>
<tr>
<td>acs.conf</td>
<td>all SLC machines</td>
<td>Configures the diamActions library.</td>
<td>acs.conf Configuration (on page 24)</td>
</tr>
<tr>
<td>oracleConfig.xsd</td>
<td>SMS</td>
<td>Defines acceptable structure for XML.</td>
<td></td>
</tr>
<tr>
<td>oracleConfigWorking.xml</td>
<td>SMS</td>
<td>The editable configuration.</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>Component</td>
<td>Locations</td>
<td>Description</td>
<td>Further Information</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>oracleConfigMaster.xml</td>
<td>SMS</td>
<td>The deployed (live) configuration used to generate the eserv.config.</td>
<td>Configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Management</td>
</tr>
</tbody>
</table>

Note: The .xsd and .xml files are only present if the Configuration Management editor has been installed. Refer to Configuration Management.

**eserv.config Configuration**

**Introduction**

The eserv.config file is a shared configuration file, from which many NCC applications read their configuration. Each NCC machine (SMS, SLC, and VWS) has its own version of this configuration file, containing configuration relevant to that machine. The eserv.config file contains different sections; each application reads the sections of the file that contains data relevant to it.

The eserv.config file is located in the /IN/service_packages/ directory.

The eserv.config file format uses hierarchical groupings, and most applications make use of this to divide up the options into logical groupings.

**Configuration file format**

To organize the configuration data within the eserv.config file, some sections are nested within other sections. Configuration details are opened and closed using either { } or [ ].

- Groups of parameters are enclosed with curly brackets - { }
- An array of parameters is enclosed in square brackets - [ ]
- Comments are prefaced with a # at the beginning of the line

To list things within a group or an array, elements must be separated by at least one comma or at least one line break. Any of the following formats may be used, as in this example:

```plaintext
{ name="route6", id = 3, prefixes = [ "00000148", "0000473"] } 
{ name="route7", id = 4, prefixes = [ "000001049" ] }
```

or

```plaintext
{ name="route6" 
  id = 3 
  prefixes = [ 
    "00000148" 
    "0000473" 
  ] 
} 
{ name="route7" 
  id = 4 
  prefixes = [ 
    "000001049" 
  ] 
} 
```

or

```plaintext
{ name="route6" 
  id = 3 
  prefixes = [ "00000148", "0000473" ] 
} 
{ name="route7", id = 4 
  prefixes = [ "000001049" ] 
} 
```
eserv.config files delivered

Most applications come with an example eserv.config configuration in a file called eserv.config.example. The example file for DCD is:

```
/IN/service_packages/DCD/etc/eserv.config.dcd.example
```

Editing the file

There are two ways to edit the eserv.config file, depending upon whether or not you are using the Configuration Management editor. If you are not, you can edit the file using a text editor.

**Warning:** If you are using the Configuration Management editor, you cannot use a text editor to edit the DCD section.

Text editor

Open the configuration file on your system using a standard text editor. Do not use text editors, such as Microsoft Word, that attach control characters. These can be, for example, Microsoft DOS or Windows line termination characters (for example: ^M), which are not visible to the user, at the end of each row. This will cause file errors when the application tries to read the configuration file.

Always keep a backup of your file before making any changes to it. This will ensure you have a working copy to which you can return.

Loading eserv.config changes

If you change the configuration file, then you must restart the appropriate parts of the service to enable the new options to take effect.

Diameter eserv.config configuration

The eserv.config file must be configured to enable the DCD to work. Most of the necessary DCD configuration in eserv.config is done at installation time by the configuration script. However, realms and hosts need to be configured.

**Note:** The DCD configuration options in eserv.config are explained in the section on the diameterBeClient background process (on page 25).

CCS eserv.config Configuration

Introduction

DCD maintains a WalletInformation cache. To enable updates to the WalletInformation cache, the service handles for the services using DCD must be mapped to the libdcdCcsSvcExtra.so library. You will need to configure the mapping in the CCS.ccsServiceLibaray.ccsPluginExtend of eserv.config.

ccsServiceLibrary configuration

Here is an example of the ccsServiceLibrary configuration for DCD in the CCS section of eserv.config.

```python
ccsServiceLibrary = {
    ccsPluginExtend = {
        library="libdcdCcsSvcExtra.so"
    }
}
```
SLEE.cfg Configuration

Introduction

The SLEE.cfg file must be configured to enable the DCD to work. All necessary SLEE configuration is done at installation time by the configuration script, this section is for information only.

The SLEE configuration file is located at:

/IN/service_packages/SLEE/etc/SLEE.cfg

Refer to SLEE Technical Guide for details on SLEE configuration.

DCD SLEE configuration

On installation, the following line is added to the SLEE.cfg

INTERFACE=dcdBeClient diameterBeClient.sh /IN/service_packages/DCD/bin EVENT

Note: It is essential for the correct operation of this application that the SLEE Interface type is always set to EVENT.

acs.conf Configuration

Introduction

The acs.conf file must be configured to enable the application to work. All necessary configuration is done at installation time by the configuration script; this section is for information only.

The ACS configuration file is located at /IN/service_packages/ACS/etc/acs.conf.

Refer to ACS Technical Guide for details on ACS configuration.

DCD acs.conf configuration

On installation, the following line is added to the acs.conf.

ChassisPlugin diamActions.so
Chapter 5

Background Processes

Overview

Introduction

This chapter explains the process which runs automatically as part of the Oracle Communications Network Charging and Control (NCC) application. This process is started automatically by the SLEE.

In this chapter

This chapter contains the following topics.

- diameterBeClient
- DCD
- DomainTypes
- PeerSchemes Configuration Section
- Routes
- NamedEventTypes
- HostSpecificData
- Statistics logged by diameterBeClient
- DCD EDRs

diameterBeClient

Purpose

The diameterBeClient takes SLEE messages from the diamActions library and converts them to Diameter messages. It also maintains all Diameter connections.

Startup

The diamActions library and diameterBeClient will load the configuration (from the eserv.config file and the SLC database) on startup and on receiving a SIGHUP, at which point it needs to reload the config.

Example DIAMETER section

Here is an example DIAMETER section configuration in the eserv.config file.

```ini
DIAMETER = {
    DCD = {
        serviceNameDomainInterfaceName = "dcdBeClient"
        loggedNotificationPeriod = 300
        loggedInvalidPeriod = 300
        databaseCacheValiditySeconds = 18000
        Origin-Host = "ocpc.oracle.com"
        Origin-Realm = "ocpc.oracle.com"
    }
}
```
serviceContextID = "ocpc@oracle.com"

Auth-Application-Id = 4

Vendor-Specific-Application-Id = [
  {
    Vendor-Id = 193
    Auth-Application-Id = 4
    Acct-Application-Id = 19302
  }
]

enableDraft8 = false

# scheme = "SchemeA"

honour_disconnect = true

Vendor-Id = 16247

Product-Name = ""

timeout_x = 30000000

transmit_limit = 2

trace_debug_flags = "all"

statistics_interval = 300

NotificationFilter = {
  CollectiveNotice = true
  PerPeerNotice = {
    initial_request = true
    update_request = true
    termination_request = true
    event_request = true
    unknown_request = true
    total_request = true
    request_timeout = true
    non_request_type_result_codes = true
  }
}

DomainTypes = [
{
  name = "DIAMETER"
  schemeName = "SchemeA"
  routing = "Round Robin"
  voidUnusedReservation = false
  defaultSessionFailover = 0
  defaultEventFailover = 0
  defaultFailureHandling = 0
  balanceEnquiryMethod = "balanceCheck"
includeDcdCdrFields = false

defaultFixedCostDuration = 86400
enableScap = false
overwriteZeroCallAnswerTime = false

Domains = [
  {
    name = "myDomainA"
    routing = "Round Robin"
    realmFailureWaitSeconds = 20
    weighting = 1
  },
  {
    name = "myDomainB"
    routing = "Failover"
    realmFailureWaitSeconds = 20
    weighting = 1
  }
]

AVPs = [
  {
    name = "ConfirmNamedEventReservationRequest"
    mandatoryContents = [
      "Subscription-Id",
      "Service-Identifier",
      "NE-Used-Service-Unit"
    ]
    optionalContents = []
  },
  {
    avpCode = 443
    name = "Subscription-Id"
    type = "Grouped"
    mandatoryContents = [
      "Subscription-Id-Type",
      "Subscription-Id-Data"
    ]
    optionalContents = []
  },
  {
    avpCode = 450
    name = "Subscription-Id-Type"
    type = "Enumerated"
    literal = "0"           # END_USER_E164
  },
  {
    avpCode = 444
    name = "Subscription-Id-Data"
    type = "UTF8String"
    ccsConcept = "acsProfile"
    profileBlock = 18
  },
profileTag = 327686
profileFormat = "LNSTRING"

conditionProfileBlock = 17
conditionProfileTag = 2009
conditionValue = 14

}  

avpCode = 439
name = "Service-Identifier"
type = "Unsigned32"
ccsConcept = "eventType"
cdrTag = "DIAMETER_SERVICE_ID"
cdrOperation = "replace"

}  

avpCode = 446
name = "NE-Used-Service-Unit"
type = "Grouped"

mandatoryContents = [  
    "CC-Service-Specific-Units-Used"
  ]

}  

avpCode = 417
name = "CC-Service-Specific-Units-Used"
type = "Unsigned64"
ccsConcept = "numUnitsUsed"

vendorId = 0
flags = 0
repeating = false
maxOccurrences = 10
conversion = [  
    {  
        esg = 1
        vendor = 2
        serviceProvider = 1
    }  
    {  
        esg = 3
        vendor = 2
        serviceProvider = 2
    }
  ]

conversionScale = 1
conversionRounding = "floor"
interpretBase = 10
octetLength = 0

}  

avpCode = 900
vendorId = 16247
name = "Custom-Scp-Action"
type = "Enumerated"
ccsConcept = "scpAction"
repeating = true

}  

name = "DirectNamedEventRequest-Money Refund"
mandatoryContents = [
    "Subscription-Id"
    "SOS-TopUp-RequestedAction"
    "Charging-Start-Timestamp"
]


name = "SOS-TopUp-RequestedAction"
type = "Enumerated"
value = "5"
avpCode = 436


name = "WalletRechargeRequest"
mandatoryContents = [
    "Subscription-Id"
    "TopUp-RequestedAction"
    "Charging-Start-Timestamp"
    "Topup-Amount"
    "Topup-Voucher-Number"
    "Topup-Voucher-Type"
    "Topup-Voucher-Id"
    "Source-System-Id"
]
optionalContents = [
    "Voucher-Recharge_Failed-Flag"
    "Voucher-Recharge_Failed_Date_Time"
]


name = "Voucher-Recharge_Failed-Flag"
type = "Integer32"
vendorId = 581
avpCode = 50998
ccsConcept = "voucherRechargeFailureFlag"
includeIf = true


name = "Voucher-Recharge_Failed-Date_Time"
type = "Integer32"
vendorId = 581
avpCode = 50999
ccsConcept = "voucherRechargeFailureDateTime"
excludeWhenIn = "0"


name = "TopUp-RequestedAction"
type = "Enumerated"
value = "4"
avpCode = 436


name = "Charging-Start-Timestamp"
type = "Integer32"
profileBlock = 18
profileTag = 327999
avpCode = 12000
{  
  name = "Topup-Voucher-Number"  
  avpCode = 12001  
  ccsConcept = "voucherInfoVoucher"  
  type = "Integer32"  
}

{  
  name = "Topup-Amount"  
  avpCode = 12002  
  type = "Grouped"  
  mandatoryContents = [  
    "Value-Digits"  
  ]  
  optionalContents = [  
    "Exponent"  
  ]  
}

{  
  name = "Value-Digits"  
  avpCode = 12003  
  ccsConcept = "voucherInfoValue"  
  type = "Integer32"  
}

{  
  name = "Topup-Voucher-Type"  
  avpCode = 12005  
  ccsConcept = "voucherTypeName"  
  type = "UTF8String"  
}

{  
  name = "Topup-Voucher-Id"  
  vendorId = 581  
  avpCode = 50026  
  ccsConcept = "voucherInfoVoucherId"  
  type = "Integer32"  
}

{  
  name = "WalletRechargeResponse"  
  mandatoryContents = [  
    "Topup-Balance-Information"  
    "Topup-Receipt-Number"  
  ]  
}

{  
  name = "Topup-Balance-Information"  
  vendorId = 581 # Intec  
  avpCode = 50030  
  type = "Grouped"  
  mandatoryContents = [  
    "Topup-Balance-Type-ID"  
  ]  
  optionalContents = [  
    "Topup-Balance-Expire-Date"  
    "Topup-Unit-Value"  
  ]  
}
{ name = "Topup-Balance-Type-ID" avpCode = 50020
vendorId = 581 # Intec
ccsConcept = "walletInfoBalanceType"
type = "Integer32"
}

{ name = "Topup-Balance-Expire-Date"
vendorId = 581 # Intec
avpCode = 50032
ccsConcept = "walletInfoBalanceExpiry"
type = "Integer32"
}

{ name = "Topup-Unit-Value"
vendorId = 581 # Intec
avpCode = 50099
type = "Grouped"
mandatoryContents = [
  "Topup-Value-Digits"
]
optionalContents = [
  "Topup-Exponent"
]
}

{ name = "Topup-Value-Digits"
avpCode = 50020
vendorId = 581 # Intec
ccsConcept = "walletInfoBalanceSystemValue"
type = "Integer32"
}

{ name = "Topup-Exponent"
avpCode = 50199 # the ICD doesn't define this but it's not an issue for testing
vendorId = 581 # Intec
type = "Integer32"
}

{ name = "Topup-Receipt-Number"
avpCode = 50024
type = "UTF8String"
}

{ name = "CreateEDRRequest"
}

{ name = "CreateEDRResponse"
}

{ name = "BadPINRequest"
{  
  name = "BadPINResponse"
}

{  
  name = "Termination-Cause"
  avpCode = 295
  type = "Enumerated"
  ccsConcept = "terminationCause"
  conversion = [
    
    
    {  
      esg = 8 # releasedNoFunds
      vendor = 6 # DIAMETER_AUTH_EXPIRED
    }
    
    
    {  
      esg = 9 # disconnectedLegBNoFunds
      vendor = 6 # DIAMETER_AUTH_EXPIRED
    }
    
    
    {  
      esg = 14 # callingPartyDisconnected
      vendor = 1 # DIAMETER_LOGOUT
    }
    
    
    {  
      esg = 15 # calledPartyDisconnected
      vendor = 1 # DIAMETER_LOGOUT
    }
  ]
}

ContextCopy = [
{  
  contextItem = "scpActionSupervise"
  profileBlock = 17
  profileTag = 6356992 # Hex 0x00610000
  profileFormat = "INTEGER"
}
{  
  contextItem = "scpActionDoNotSupervise"
  profileBlock = 17
  profileTag = 6356993 # Hex 0x00610001
  profileFormat = "INTEGER"
}
{  
  contextItem = "scpActionRelease"
  profileBlock = 17
  profileTag = 6356994 # Hex 0x00610002
  profileFormat = "INTEGER"
}
{  
  contextItem = "scpActionSendMessage"
  profileBlock = 17
  profileTag = 6356995 # Hex 0x00610003
  profileFormat = "INTEGER"
}
{  
  contextItem = "scpActionPlayAnnouncement"
  profileBlock = 17
  profileTag = 6356996 # Hex 0x00610004
  profileFormat = "INTEGER"
}
{  
    contextItem = "scpActionSuperviseWithoutControlling"
    profileBlock = 17
    profileTag = 6356997 # Hex 0x0061005
    profileFormat = "INTEGER"
}
{  
    contextItem = "callState"
    profileBlock = 17
    profileTag = 6356998 # Hex 0x0061006
    profileFormat = "INTEGER"
}
{  
    contextItem = "sendCount"
    profileBlock = 17
    profileTag = 6356999 # Hex 0x0061007
    profileFormat = "INTEGER"
}
{  
    contextItem = "preCallAnnouncementId"
    profileBlock = 17
    profileTag = 6357000 # Hex 0x0061008
    profileFormat = "INTEGER"
}
{  
    contextItem = "preCallLowBalance"
    profileBlock = 17
    profileTag = 6357001 # Hex 0x0061009
    profileFormat = "INTEGER"
}

TimeIn = {
    profileBlock = 17
    profileTag = 6357002 # Hex 0x006100a, continuing from example ContextCopy values
    profileFormat = "TIME"
}

TimeOut = {
    profileBlock = 17
    profileTag = 6357003 # Hex 0x006100b
    profileFormat = "TIME"
}

}  
] # End of DomainTypes array

PeerSchemes = [
{  
    schemeName = "SchemeA"

    Peers = [
    {  
        name = "host1"

        scheme = [ "scheme1", "schmeme2" ]

        permittedOriginHosts = [
            "host1.realm1.oracle.com"
        ]

        peer_group = "host1"
    }
    ]
}
```plaintext
transport = "tcp"
initiation = "connect"
RemoteAddresses = [
    "192.168.1.10"
]
remote_port = 3868
netmaskBits = 32
permittedInstances = 0
reqSctpInboundStreams = 8
reqSctpOutboundStreams = 8
sctp_hbinterval = 1000
watchdogPeriod = 30
connectionTimeout = 30
inBufferSize = 0
outBufferSize = 0
} # end of Peer host1

{
    name = "host2"
scheme = [ "scheme1", "schmeme2" ]
permittedOriginHosts = [
    "host1.realm1.oracle.com"
]
peer_group = "host1"
transport = "tcp"
initiation = "connect"
RemoteAddresses = [
    "192.168.1.11"
]
remote_port = 3868
netmaskBits = 32
permittedInstances = 0
reqSctpInboundStreams = 8
reqSctpOutboundStreams = 8
sctp_hbinterval = 1000
watchdogPeriod = 30
connectionTimeout = 30
inBufferSize = 0
outBufferSize = 0
```
Chapter 5

}{ # end of Peer host1

} # End of Scheme A

{ 
schemeName = "SchemeB"

Peers = [ 
{
    name = "host1"
    scheme = [ "scheme1", "scheme2" ]
    permittedOriginHosts = [ 
        "host1.realm1.oracle.com"
    ]
    peer_group = "host1"
    transport = "tcp"
    initiation = "connect"
    RemoteAddresses = [ 
        "192.168.1.10"
    ]
    remote_port = 3868
    netmaskBits = 32
    permittedInstances = 0
    reqSctpInboundStreams = 8
    reqSctpOutboundStreams = 8
    sctp_hbinterval = 1000
    watchdogPeriod = 30
    connectionTimeout = 30
    inBufferSize = 0
    outBufferSize = 0

} # end of Peer host1

} # End of Scheme B
} # End of PeerSchemes section

routes = [ 
{
    realm = "FirstRealm"
    host = "host1.realm1.oracle.com"
    priority = 1
    round_robin = 0
    direct = true
}
]

NamedEventTypes = [
Chapter 5

{  
  eventClass = "abc"
  eventName = "def"
  eventType = 123
  isDebit = true
}

{  
  eventClass = "ghi"
  eventName = "jkl"
  eventType = 456
}

HostSpecificData = [  
  {  
    name = "ocpc.oracle.com"
  }  
]

DCD = {  
  Origin-Host = "ocpc.oracle.com"
  Origin-Realm = "ocpc.oracle.com"
}

routes = [  
  {  
    realm = "myDomainA"
    host = "host1.realm1.oracle.com"
    priority = 1
    round_robin = 0
    direct = true
  }  
]

DCD

Introduction

The DCD section holds global configuration for the DCD application.

Note: This section is optional.

DCD parameters

Here are the parameters for the DCD section.

DCD = {  
  serviceDomainInterfaceName = "dcdBeClient"
  loggedNotificationPeriod = 300
  loggedInvalidPeriod = 300
  databaseCacheValiditySeconds = 18000
  Origin-Host = "ocpc.oracle.com"
  Origin-Realm = "ocpc.oracle.com"
}
serviceContextID = "ocpc@oracle.com"

Auth-Application-Id = 4

Vendor-Specific-Application-Id = {
    Vendor-Id = 193
    Auth-Application-Id = 4
    Acct-Application-Id = 19302
}

enableDraft8 = false

# scheme = "SchemeA"

honour_disconnect = true

Vendor-Id = 16247

Product-Name = ""

timeout_x = 30000000

transmit_limit = 2

trace_debug_flags = "all"

statistics_interval = 300

NotificationFilter = {
    CollectiveNotice = true
    PerPeerNotice = {
        initial_request = true
        update_request = true
        termination_request = true
        event_request = true
        unknown_request = true
        total_request = true
        request_timeout = true
        non_request_type_result_codes = true
    }
}

You may also set defaults for peer connection settings here. See Peer host parameters (on page 71) for details of the available options.

Auth-Application-Id

Syntax: 

Auth-Application-Id = ID

Description: This AVP value is set to the Credit-Control as DCD uses Credit-Control capability by default.

Type: Integer

Optionality: Optional (not sent if not set).

Allowed: Not sent

Default: Not sent
Notes:

- This AVP can be configured as a sub-AVP to support Vendor-Specific-Application-Id (on page 43).
- In case of vendor specific configuration, this value should NOT be set globally.

Example:  

Auth-Application-Id = 4

**Acct-Application-Id**

Syntax:  

Acct-Application-Id = ID

Description:  

This AVP value is configured as a sub-AVP to support Vendor-Specific-Application-Id (on page 43).

Type:  

Integer

Optionality:  

Optional (not sent if not set).

Allowed:  

Default:  

Notes:  

Either one, or both of Auth-Application-Id or Acct-Application-Id should be set in order to support Vendor-Specific-Application-Id (on page 43).

Example:  

Acct-Application-Id = 12300

**checkDefinedAvpFlags**

Syntax:  

checkDefinedAvpFlags = true|false

Description:  

Whether to check incoming AVPs for flags that are defined in the base Diameter protocol, or in any of the Diameter application specifications, so that unknown mandatory AVPs can be excluded.

Type:  

Boolean

Optionality:  

Optional (default used if not set)

Allowed:  

ture – Check flags of incoming AVPs
false – Do not check flags of incoming AVPs

Default:  

Example:  

checkDefinedAvpFlags = false

**databaseCacheValiditySeconds**

Syntax:  

databaseCacheValiditySeconds = seconds

Description:  

Defines how long to keep data from the database after loading it.

Type:  

Integer

Optionality:  

Optional

Allowed:  

Seconds

Default:  

18000 seconds (5 hours)

Example:  

databaseCacheValiditySeconds = 18000

**enableDraft8**

Syntax:  

enableDraft8 = true|false

Description:  

Indicates if Draft 8 version of the Diameter base protocol should be used. This is required to support the enableScap (on page 45) parameter.

Type:  

Boolean

Optionality:  

Optional (default used if not set).

Allowed:  

ture, false
Default: false
Notes:
Example: enableDraft8 = false

honour_disconnect
Syntax: honour_disconnect = true|false
Description: This is whether to use the RFC 3588 disconnect logic.
Type: Boolean
Optionality: Optional
Allowed: true, false
Default: false
Notes: By default the RFC3588 disconnect logic is enabled. If your server incorrectly sends DPR messages, then disable this by setting to false.
Example: honour_disconnect = true

loggedInvalidPeriod
Syntax: loggedInvalidPeriod = period
Description: This is how often to announce the number of invalid messages.
Type: Integer
Optionality: Optional
Allowed: Seconds
Default: 300
Example: loggedInvalidPeriod = 300

loggedNotificationPeriod
Syntax: loggedNotificationPeriod = period
Description: This is how often to announce the number of recent message parse errors.
Type: Integer
Optionality: Optional
Allowed: Seconds
Default: 300
Example: loggedNotificationPeriod = 300

NotificationFilter
Syntax: NotificationFilter = {
    CollectiveNotice =
    PerPeerNotice = {}
}
Description: This section sets flags allowing the appropriate notification to be enabled or suppressed.
Type: Section of boolean parameters
Optionality: Optional, default used if not set
Default: All values default to true (that is, statistics recording in DCD alarm log enabled).
Notes: Settings are only applicable to statistics that are recorded in the DCD alarm log.
Example: NotificationFilter = {

CollectiveNotice = true
PerPeerNotice = {
    initial_request = true
    update_request = true
    termination_request = true
    event_request = true
    unknown_request = true
    total_request = true
    request_timeout = true
    non_request_type_result_codes = true
}

CollectiveNotice
Syntax: collectiveNotice = true|false
Description: Enables or suppresses the recording of CCR request type statistics.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: 
    - true - enable CCR request type statistics
    - false - suppress CCR request type statistics
Default: true
Notes: Counts are collective across all peers and not separated on a per peer basis.
Example: collectiveNotice = true

PerPeerNotice
Syntax: PerPeerNotice = {request_type = true|false> list}
Description: Enables/suppresses statistic counts per peer for both CCR and CCA for the listed CC_Request_types.
Type: Section of boolean parameters
Optionality: Optional (default used if not set).
Allowed: Each request type can be set to:
    - true (statistic enabled)
    - false (statistic suppressed)
See example for list of request types.
Default: All values default to true (statistic enabled).
Notes:
Example: PerPeerNotice = {
    initial_request = true
    update_request = true
    termination_request = true
    event_request = true
    unknown_request = true
    total_request = true
    request_timeout = true
    non_request_type_result_codes = true
}

Origin-Host
Syntax: Origin-Host = "diameterId"
Description: The Diameter AVP Origin-Host - fully qualified domain name.
Type: String
<table>
<thead>
<tr>
<th>Optionality</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td>May be specified for each host. See HostSpecificData parameters (on page 78).</td>
</tr>
<tr>
<td>Example</td>
<td>Origin-Host = &quot;ocpc.oracle.com&quot;</td>
</tr>
</tbody>
</table>

**Origin-Realm**

- **Syntax:** Origin-Realm = "diameterId"
- **Description:** The Diameter AVP Origin-Realm.
- **Type:** String
- **Optionality:** Optional
- **Notes:** May be specified for each host. See HostSpecificData parameters (on page 78).
- **Example:** Origin-Realm = "ocpc.oracle.com"

**Product-Name**

- **Syntax:** Product-Name = "name"
- **Description:** The Product name for CER/CEA.
- **Type:** String
- **Optionality:** Optional
- **Default:** empty
- **Example:** Product-Name = ""

**scheme**

- **Syntax:** scheme = "type"
- **Description:** The global scheme to use.
- **Type:** String
- **Optionality:** If you use the peerSchemes section, then this is mandatory.
- **Example:** scheme = "schemeA"

**serviceDomainInterfaceName**

- **Syntax:** serviceDomainInterfaceName = "name"
- **Description:** The interface name of the Diameter BE client (in SLEE.cfg).
- **Type:** String
- **Optionality:** Optional
- **Default:** dcdBeClient
- **Example:** serviceDomainInterfaceName = "dcdBeClient"

**serviceContextID**

- **Syntax:** serviceContextID = "ContextID"
- **Description:** The Diameter AVP Service-Context-Id.
- **Type:** String
- **Optionality:** Mandatory
- **Example:** serviceContextID = "ocpc@oracle.com"

**statistics_interval**

- **Syntax:** statistics_interval = seconds
- **Description:** This is how often to record aggregate (average, min, max) latency statistics.
Type: Integer
Optionality: Mandatory
Allowed: In seconds.
Default: 300 (that is, 5 minutes)
Example: statistics_interval = 300

timeout_x
Syntax: timeout_x = timer
Description: The RFC 4006 Tx timer.
Type: Integer
Optionality: Optional
Allowed: In microseconds
Default: 30000000
Example: timeout_x = 30000000

trace_debug_flags
Syntax: trace_debug_flags = "flags"
Description: The debug flags to turn on if tracing (in the client) is requested by the actions library.
Type: String
Optionality: Optional
Default: "all"
Example: trace_debug_flags = "all"

transmit_limit
Syntax: transmit_limit = limit
Description: The maximum number of retransmits (including the original transmission) allowed for a message.
Type: Integer
Optionality: Optional
Example: transmit_limit = 2

Vendor-Id
Syntax: Vendor-Id = ID
Description: The Vendor ID for CER/CEA.
Type: Integer
Optionality: Optional
Default: 16247 (the Oracle Vendor-Id)
Notes:
- This AVP can be configured as a sub-AVP to support Vendor-Specific-Application-Id (on page 43).
- It is mandatory to specify vendorId when configuring vendor specific AVPs for SCAP.
Example: Vendor-Id = 16247
Vendor-Specific-Application-Id

Syntax: Vendor-Specific-Application-Id = [
    
    Vendor-Id = int
    #Auth-Application-Id = int
    Acct-Application-Id = int

]

Description: Lists the Vendor specific AVPs required for enabling SCAP.

Type: Parameter array

Optionality: Optional (default used if not set).

Allowed: Parameter array is not specified.

Notes: Do NOT set the global Auth-Application-Id (on page 37) if configuring this parameter array.

Example: Vendor-Specific-Application-Id = [
    
    Vendor-Id = 123
    Acct-Application-Id = 12345

]

DomainTypes

Introduction

The DomainTypes section lists all the DCD domain types and the associated configuration.

Note: This section is mandatory, and it must include one domain type, with a name and scheme defined.

DomainTypes parameters

Here is an example high-level structure showing the parameters for the DomainTypes section.

DomainTypes = [
    
    name = "DIAMETER"
    schemeName = "SchemeA"
    routing = "Round Robin"
    voidUnusedReservation = false
    defaultSessionFailover = 0
    defaultEventFailover = 0
    defaultFailureHandling = 0
    balanceEnquiryMethod = "balanceCheck"
    includeDcdCdrFields = false
    defaultFixedCostDuration = 86400
    enableScap = false
    overwriteZeroCallAnswerTime = false

    Domains = [}
Chapter 5

\[
\begin{align*}
&\{First\_Domain \\
&\{Next\_Domain \\
&\}
\]
\]
AVPs = [
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]
\]

balanceEnquiryMethod

**Syntax:**

\[
balanceEnquiryMethod = "method"
\]

**Description:**
The method to use to allow balance queries.

**Type:**
String

**Optionality:**
Optional (default used if not set).

**Allowed:**
- "balanceCheck" uses a Balance Check message with a Service Identifier set to "Information" to trigger the query, or
- "reqActionViewBalance" uses a special Requested-Action AVP with a value of VIEW_BALANCE(5) to trigger the query.

**Default:**
"balanceCheck"

**Notes:**
Example:
balanceEnquiryMethod = "balanceCheck"

defaultEventFailover

**Syntax:**

defaultEventFailover = failover

**Description:**
The default event failover.

**Type:**
Integer

**Optionality:**
Optional (default used if not set).

**Allowed:**
The failover corresponds to the values defined for Credit Control Failure Handling AVP in RFC 4006:
- 0 - Terminate
- 1 - Continue
- 2 - Retry and Terminate

**Default:**
0 (Failover not supported)

**Notes:**
Until the BE responds for the event, the failover behavior is determined by this and the defaultSessionFailover parameter.

**Example:**
defaultEventFailover = 0

44 NCC Diameter Charging Driver Technical Guide
defaultFailureHandling
Syntax: \texttt{defaultFailureHandling = number}
Description: How to behave until a Credit-Control-Failure-Handling AVP (as defined in RFC 4006) is received from the server.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 0 - Terminate
1 - Continue
2 - Retry and Terminate
Default: 0 (Terminate)
Notes:
Example: \texttt{defaultFailureHandling = 0}

defaultFixedCostDuration
Syntax: \texttt{defaultFixedCostDuration = duration}
Description: The default time to use for free and fixed cost calls.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: In seconds. May be any positive value.
Default: 86400 (one day)
Notes:
Example: \texttt{defaultFixedCostDuration = 86400}

defaultSessionFailover
Syntax: \texttt{defaultSessionFailover = failover}
Description: The default session failover.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Corresponds to the values defined for Credit Control Failure Handling AVP in RFC 4006:
\begin{itemize}
  \item 0 - Terminate
  \item 1 - Continue
  \item 2 - Retry and Terminate
\end{itemize}
Default: 0 (Failover not supported)
Notes: Until the BE responds in a session, the failover behavior for a CC session is determined by this.
Example: \texttt{defaultSessionFailover = 0}

enableScap
Syntax: \texttt{enableScap = true|false}
Description: If set to true, support for Ericsson SCAP (Service Charging Application Protocol) to DCD is enabled. This ensures SCAP type accounting messages are used for credit control, rather than CCR/CCA.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed:  
- true - enabled for SCAP compliance  
- false - disabled for SCAP compliance

Default:  false

Notes:

Example:  enableScap = false

forceWalletReload

Syntax:  forceWalletReload = true|false

Description:  Defines whether or not the cache may be used for balance inquiries.

Type:  Boolean

Optionality:  Optional (default used if not set).

Allowed:  true, false

Default:  false

Notes:

Example:  forceWalletReload = true

includeDcdCdrFields

Syntax:  includeDcdCdrFields = true|false

Description:  Whether or not to record the Result-Code in call EDRs.

Type:  Boolean

Optionality:  Optional (default used if not set).

Allowed:  true, false

Default:  false

Notes:

Example:  includeDcdCdrFields = false

name

Syntax:  name = "type"

Description:  The name of the Domain Type as defined in Prepaid Charging.

Type:  String

Optionality:  Mandatory


Example:  name = "DIAMETER"

overwriteZeroCallAnswerTime

Syntax:  overwriteZeroCallAnswerTime = true|false

Description:  Enables overwriting of the Charging-Start-Timestamp using the EventTimestamp.

Type:  Boolean

Optionality:  Optional (default used if not set).

Allowed:  
- true - overwrite charging start time  
- false - if a call is not answered (that is, CallAnswerTime is zero) then set the charging start timestamp AVP to 'Jan 1, 1970 00:00:00.0000 UTC'.

Default:  false

Notes:

Example:  overwriteZeroCallAnswerTime = false
routing

Syntax: \( \text{routing} = "\text{name}" \)

Description: The algorithm to use when picking domains within the domain type.

Type: String

Optionality: Optional

Allowed:
- "Round Robin" (a weighted round robin algorithm)
- "Failover"

Default: "Round Robin"

Example: \( \text{routing} = "\text{Round Robin}" \)

schemeName

Syntax: \( \text{schemeName} = "\text{name}" \)

Description: The name of the peer scheme to use with this domain type.

Type: String

Optionality: Mandatory

Example: \( \text{schemeName} = \text{"SchemeA"} \)

voidUnusedReservation

Syntax: \( \text{voidUnusedReservation} = \text{true|false} \)

Description: Whether or not to void unused reservations.

Type: Boolean

Optionality: Optional

Allowed: true, false

Default: false

Notes: Diameter states that the server and client should consider any previously reserved, but unused, time in a subsequent reservation as no longer reserved. This is counter to most telephony models, so is disabled by default. To enable (for this domain type), set this parameter to true.

Example: \( \text{voidUnusedReservation} = \text{false} \)

dynamicWalletReload

The \text{dynamicWalletReload} section defines the profile to use to determine whether or not to dynamically force wallet reloads. A wallet reload will be forced if the profileFormat value is one of the following:

- "INTEGER" and the data in the profile is 1 (stored as 4-bytes)
- "BOOLEAN" and the data in the profile is 1 (stored as a single byte)
- "STRING" and the data in the profile is one of:
  - "true"
  - "yes"
  - "y"
  - "1"

The parameters are defined in the \text{ContextCopy parameters} (on page 67) section.

Notes:
- Only the profile formats listed here will be supported, all other values will be ignored.
The `forceWalletReload` parameter must be set to false (or not specified).

**Example dynamicWalletReload**
Here are the example parameters.

```plaintext
dynamicWalletReload = {
    profileBlock = 17
    profileTag = 6357900
    profileFormat = "INTEGER"
}
```

**Domains parameters**
Here is an example of the Domains section.

```plaintext
Domains = [
    {
        name = "myDomainA"
        routing = "Round Robin"
        realmFailureWaitSeconds = 20
        weighting = 1
    }
    {
        name = "myDomainB"
        routing = "Failover"
        realmFailureWaitSeconds = 20
        weighting = 1
    }
]
```

**name**
- **Syntax:** `name = "name"`
- **Description:** The name of the domain as defined in Prepaid Charging.
- **Type:** String
- **Optionality:** Mandatory
- **Allowed:** Defined in Prepaid Charging from available DIAMETER domains on the Domain tab of the Service Management screen. Refer to *CCS User's Guide*.
- **Example:** `name = "myDomainA"

**realmFailureWaitSeconds**
- **Syntax:** `ArraySize = as`
- **Description:** How long we wait for a Realm that has had a failure before retrying it.
- **Type:** Integer
- **Optionality:** optional
- **Allowed:** in seconds
- **Default:** 30
- **Example:** `realmFailureWaitSeconds = 20`
Chapter 5

routing
Syntax:    routing = "type"
Description: The routing type to use when picking Realms within this Domain.
Type: String
Optionality: Optional
Allowed:
    • “Round Robin” (a weighted round robin algorithm)
    • “Failover”
Default: “Round Robin”
Example: routing = "Round Robin"

weighting
Syntax:    weighting = weight
Description: The domain's weighting, if our Domain Type is using Round Robin routing.
Type: Integer
Optionality: Optional
Allowed: 1
Default: 1
Notes: The weighting determines how many times the domain is used. The weightings of all the active domains are added together and the domain receives its percentage of usage.
For example:
    • myDomainA has a weighting of 1
    • myDomainB a weighting of 1
    • myDomainC a weighting of 2
myDomainA has 25% of the total usage. If myDomainC becomes unavailable, then myDomainA will have 50%.
Example: weighting = 1

AVPs
In the AVPs section, we define (as a tree-like structure) the configurable AVP part of the message for every action that uses Diameter requests and responses.
A worked example is provided in the eserv.config.default file.
The actions for which configuration is required are:
- BadPINRequest, BadPINResponse
- ConfirmNamedEventReservationRequest, ConfirmNamedEventReservationResponse
- ConfirmTimeReservationRequest, ConfirmTimeReservationResponse
- CreateEDRRequest, CreateEDRResponse
- DirectNamedEventRequest, DirectNamedEventResponse
- DirectTimeChargeRequest, DirectTimeChargeResponse
- ExtendTimeReservationRequest, ExtendTimeReservationResponse
- GetNamedEventRatesRequest, GetNamedEventRatesResponse
- InitialTimeReservationRequest, InitialTimeReservationResponse
- NamedEventReservationRequest, NamedEventReservationResponse
- RevokeNamedEventReservationRequest, RevokeNamedEventReservationResponse
Chapter 5

- `RevokeTimeReservationRequest`, `RevokeTimeReservationResponse`
- `WalletRechargeRequest`, `WalletRechargeResponse`
- `WalletInfoRequest`, `WalletInfoResponse`

### AVPs parameters

Here is an example of the AVPs section.

```plaintext
AVPs = [

    {
        name = "ConfirmNamedEventReservationRequest"
        mandatoryContents = [
            "Subscription-Id",
            "Service-Identifier",
            "NE-Used-Service-Unit"
        ]
        optionalContents = []
    },
    {
        avpCode = 443
        name = "Subscription-Id"
        type = "Grouped"
        mandatoryContents = [
            "Subscription-Id-Type",
            "Subscription-Id-Data"
        ]
        optionalContents = []
    },
    {
        avpCode = 450
        name = "Subscription-Id-Type"
        type = "Enumerated"
        literal = "0"  # END_USER_E164
    },
    {
        avpCode = 444
        name = "Subscription-Id-Data"
        type = "UTF8String"
        ccsConcept = "acsProfile"
        profileBlock = 18
        profileTag = 32768
        profileFormat = "LNSTRING"
        conditionProfileBlock = 17
        conditionProfileTag = 2009
        conditionValue = 14
    },
    {
        avpCode = 439
        name = "Service-Identifier"
        type = "Unsigned32"
        ccsConcept = "eventType"
        cdrTag = "DIAMETER_SERVICE_ID"
        cdrOperation = "replace"
    },
    {
        avpCode = 446
        name = "NE-Used-Service-Unit"
        type = "Grouped"
        mandatoryContents = [
```
"CC-Service-Specific-Units-Used"
]
{
    avpCode = 417
    name = "CC-Service-Specific-Units-Used"
    type = "Unsigned64"
    ccsConcept = "numUnitsUsed"
    vendorId = 0
    flags = 0
    repeating = false
    maxOccurrences = 10
    conversion = [
        {  
            esg = 1
            vendor = 2
            serviceProvider = 1
        },
        {  
            esg = 3
            vendor = 2
            serviceProvider = 2
        }
    ]
    conversionScale = 1
    conversionRounding = "floor"
    interpretBase = 10
    octetLength = 0
}
{
    avpCode = 900
    vendorId = 16247
    name = "Custom-Scp-Action"
    type = "Enumerated"
    ccsConcept = "scpAction"
    repeating = true
}
{
    name = "DirectNamedEventRequest-Money Refund"
    mandatoryContents = [  
        "Subscription-Id"
        "SOS-TopUp-RequestedAction"
        "Charging-Start-Timestamp"
    ]
}
{
    name = "SOS-TopUp-RequestedAction"
    type = "Enumerated"
    value = "5"
    avpCode = 436
}
{
    name = "WalletRechargeRequest"
    mandatoryContents = [  
        "Subscription-Id"
    ]
"TopUp-RequestedAction"
"Charging-Start-Timestamp"
"Topup-Amount"
"Topup-Voucher-Number"
"Topup-Voucher-Type"
"Topup-Voucher-Id"
"Source-System-Id"
]
optionalContents = [
  "Voucher-Recharge_Failed-Flag"
  "Voucher-Recharge_Failed_Date_Time"
]

{name = "Voucher-Recharge_Failed-Flag"
type = "Integer32"
vendorId = 581
avpCode = 50998
ccsConcept = "voucherRechargeFailureFlag"
includeIf = true}

{name = "Voucher-Recharge_Failed-Date_Time"
type = "Integer32"
vendorId = 581
avpCode = 50999
ccsConcept = "voucherRechargeFailureDateTime"
excludeWhenIn = "0"}

{name = "TopUp-RequestedAction"
type = "Enumerated"
value = "4"
avpCode = 436}

{name = "Charging-Start-Timestamp"
type = "Integer32"
profileBlock = 18
profileTag = 327999
avpCode = 12000}

{name = "Topup-Voucher-Number"
avpCode = 12001
ccsConcept = "voucherInfoVoucher"
type = "Integer32"}

{name = "Topup-Amount"
avpCode = 12002
type = "Grouped"
mandatoryContents = [
  "Value-Digits"
]
optionalContents = [
  "Exponent"
]
Chapter 5

{ name = "Value-Digits"
  avpCode = 12003
  ccsConcept = "voucherInfoValue"
  type = "Integer32"
}

{ name = "Topup-Voucher-Type"
  avpCode = 12005
  ccsConcept = "voucherTypeName"
  type = "$UTF8String"
}

{ name = "Topup-Voucher-Id"
  vendorId = 581
  avpCode = 50026
  ccsConcept = "voucherInfoVoucherId"
  type = "Integer32"
}

{ name = "WalletRechargeResponse"
  mandatoryContents = [
    "Topup-Balance-Information",
    "Topup-Receipt-Number"
  ]
}

{ name = "Topup-Balance-Information"
  vendorId = 581 # Intec
  avpCode = 50030
  type = "Grouped"
  mandatoryContents = [
    "Topup-Balance-Type-ID"
  ]
  optionalContents = [
    "Topup-Balance-Expire-Date",
    "Topup-Unit-Value"
  ]
}

{ name = "Topup-Balance-Type-ID"
  avpCode = 50020
  vendorId = 581 # Intec
  ccsConcept = "walletInfoBalanceType"
  type = "Integer32"
}

{ name = "Topup-Balance-Expire-Date"
  vendorId = 581 # Intec
  avpCode = 50032
  ccsConcept = "walletInfoBalanceExpiry"
  type = "Integer32"
}

{ name = "Topup-Unit-Value"
vendorId = 581 # Intec
avpCode = 50099

type = "Grouped"

mandatoryContents = [
    "Topup-Value-Digits"
]

optionalContents = [
    "Topup-Exponent"
]

{ name = "Topup-Value-Digits"
  avpCode = 50020
  vendorId = 581 # Intec
  ccsConcept = "walletInfoBalanceSystemValue"
  type = "Integer32"
}

{ name = "Topup-Exponent"
  avpCode = 50199 # the ICD doesn't define this but it's not an issue for testing
  vendorId = 581 # Intec
  type = "Integer32"
}

{ name = "Topup-Receipt-Number"
  avpCode = 50024
  type = "UTF8String"
}

{ name = "CreateEDRRequest"
}

{ name = "CreateEDRResponse"
}

{ name = "BadPINRequest"
}

{ name = "BadPINResponse"
}

{ name = "Termination-Cause"
  avpCode = 295
  type = "Enumerated"
  ccsConcept = "terminationCause"
  conversion = [
    {
      esg = 8 # releasedNoFunds
      vendor = 6 # DIAMETER_AUTH_EXPIRED
    }
    {
      esg = 9 # disconnectedLegBNoFunds
      vendor = 6 # DIAMETER_AUTH_EXPIRED
    }
  ]
### avpCode

**Syntax:**
```
avpCode = code
```

**Description:**
The numeric tag code that is to be set whenever an AVP of this type is created (for example, added to a request message). It can also be used to ascertain the type of AVP unpacked from a response message.

**Type:** Integer

**Optionality:** Mandatory

**Example:**
```
avpCode = 888005
```

### ccsConcept

**Syntax:**
```
ccsConcept = "concept"
```

**Description:**
The “CCS concept” to which the AVP directly relates.

**Type:** String

**Optionality:** Optional. Used by some, though not all AVPs.

**Allowed:**
The value is a string value, associated in the code with an enumeration. See `ccsConcepts` (on page 4).

**Notes:**
The AVPs are ultimately filled out from available “ccsConcepts”. These represent variables available to the DCD actions library at the time of sending the message.

**Example:**
```
ccsConcept = "acsProfile"
```

### cdrTag

**Syntax:**
```
cdrTag = "tag_name"
```

**Description:**
The EDR tag name to amend the EDR record with, depending on the `cdrOperation` parameter value.

**Type:** String

**Optionality:** Optional

**Allowed:**
Alphanumeric characters only, plus underscore.

**Default:** None

**Notes:**
- `cdrTag` is optional, but if `cdrOperation` is specified then `cdrTag` must also be specified.
- For an AVP you can just specify `cdrTag`, then `cdrOperation` will default to "replace".
- The tag name should refer to a DCD tag, not one of the pre-defined ACS tags. Otherwise the operation will have no effect.

**Example:**
```
cdrTag = "CMX_EN"
```
cdrOperation

Syntax: 

cdrOperation = "operation"

Description: The operation to perform on the cdrTag value in the EDR record.

Type: String

Optionality: Optional - only referred to if cdrTag is non-blank.

Allowed: 
- replace
- leave
- append

Not case sensitive, for example Replace = REPLACE = replace

Default: replace

Notes: 
- If cdrOperation is specified then cdrTag must also be specified.
- replace - if this tag is present, all instances are removed from the EDR and then append this instance to the EDR.
- leave - if this tag already exists, do nothing, otherwise append this instance to the EDR.
- append - regardless of existence or not of this tag, append this instance to the EDR.

Example: 
cdrOperation = "replace"

conditionProfileBlock

Syntax: 

conditionProfileBlock = block_number

Description: The profile block to use for conditional AVP filling.

Type: Integer

Optionality: Optional (when omitted no condition checking is performed).

Allowed: Any valid profile block number.

Default: None

Notes: 
- If both the block and tag are specified, and there is no data in the location, then condition checking will fail and the AVP will not be populated.

Example: 
conditionProfileBlock = 17

conditionTag

Syntax: 

conditionTag = tag_number

Description: The profile block field to use for conditional AVP filling.

Type: Integer

Optionality: Optional (missing then no condition checking is performed).

Allowed: Any valid profile field number.

Default: None

Notes: 
- If both the block and tag are specified, and there is no data in the location, then condition checking will fail and the AVP will not be populated.

Example: 
conditionTag = 2009

conditionValue

Syntax: 

conditionValue = condition

Description: The value to use to determine if AVP filling is to be performed.

Type: Integer

Optionality: Optional (default used if omitted).
Chapter 5

Allowed: Any positive integer.
Default: 1
Notes: This example populates the AVP if and only if profile block/tag contains value 14.
Example: conditionValue = 14

conversion

Syntax: conversion = [mapping]
Description: An array of integer values, defining a mapping from the Oracle enumeration to the billing vendor’s equivalent enumeration value.

serviceProvider is optional, and allows a limiting of a specific conversion to a single service provider.
Type: Integer
Optionality: Mandatory
Notes: It is used if the AVP type is any of the integer types (including enumeration). This mapping is performed regardless of the ccsConcept, and is the responsibility of the AVP traverser, not the ccsConcept helper functions.
Example: conversion = [
    {
        esg = 880
        vendor = 880880
        serviceProvider = 1
    }
    {
        esg = 890
        vendor = 890890
        serviceProvider = 2
    }
]

esg

Syntax: esg = int
Description: The integer value in the conversion array that indicates which equivalent value should be used by the billing vendor.
Type: Integer
Optionality: Mandatory
Allowed: 
Default: 
Notes: This value differs based on the AVP that uses it.

For example, see ACS Action handler (on page 5) for esg values defined for the Termination-Cause AVP in the configuration.
Example: esg = 3

vendor

Syntax: vendor = int
Description: The billing vendor's integer value in the conversion array that maps to the equivalent esg value.
Type: Integer
Optionality: Mandatory
This value differs based on the AVP that uses it. For example, the vendor values defined for the Termination-Cause AVP in the configuration are as follows:

<table>
<thead>
<tr>
<th>Integer</th>
<th>Reason</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DIAMETER_LOGOUT</td>
<td>The user initiated a disconnect</td>
</tr>
<tr>
<td>2</td>
<td>DIAMETER_SERVICE _NOT_PROVIDED</td>
<td>This value is used when the user disconnected prior to the receipt of the authorization answer message.</td>
</tr>
<tr>
<td>3</td>
<td>DIAMETER_BAD_ANSWER</td>
<td>This value indicates that the authorization answer received by the access device was not processed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>DIAMETER_ADMINISTRATIVE</td>
<td>The user was not granted access, or was disconnected, due to administrative reasons, such as the receipt of a Abort-Session-Request message.</td>
</tr>
<tr>
<td>5</td>
<td>DIAMETER_LINK_BROKEN</td>
<td>The communication to the user was abruptly disconnected.</td>
</tr>
<tr>
<td>6</td>
<td>DIAMETER_AUTH_EXPIRED</td>
<td>The user’s access was terminated since its authorized session time has expired.</td>
</tr>
<tr>
<td>7</td>
<td>DIAMETER_USER_MOVED</td>
<td>The user is receiving services from another access device.</td>
</tr>
<tr>
<td>8</td>
<td>DIAMETER_SESSION_TIMEOUT</td>
<td>The user’s session has timed out, and service has been terminated.</td>
</tr>
</tbody>
</table>

Example: vendor = 6

serviceProvider

Syntax: serviceProvider = int

Description: If mentioned, the conversion is limited to the specific service provider.

Type: Integer

Optionality: Optional

Allowed: Optional

Default: None

Notes: None

Example: serviceProvider = 2
conversionScale
Syntax: conversionScale = scale
Description: Defines a scaling of esg integer values to get server integer values.
Type: Integer
Optionality: Optional
Default: 1
Notes: Positive means multiply, negative means divide.
Any scaling occurs AFTER conversion rules defined above.
Example: conversionScale = -10
Convert incoming Diameter seconds into deci-seconds.

conversionRounding
Syntax: conversionRounding = "rounding_type"
Description: The conversion method to use between internal and server numeric values. If an AVP has the conversionScale parameter set then conversionRounding may also be configured, otherwise the conversionRounding parameter will be ignored.
Type: String
Optionality: Optional (default used if not set).
Allowed: floor - any fractions are dropped
        ceiling - round up fractional parts,
        round - round to nearest whole number. that is, x.5 or higher will be rounded up and others will be rounded down.
Default: floor - existing behavior is preserved.
Notes:
Example: conversionRounding = "round"

excludeIf
Syntax: excludeIf = true|false
Description: DCD outgoing messages will exclude AVP values that match this rule.
Type: Boolean
Optionality: Optional.
Allowed: true, false
Default: N/A
Notes: The AVP type should be set to "Integer32". If includeIf is also defined then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either excludeIf or includeIf, but not both.
Example: excludeIf = false

excludeIfMatches
Syntax: excludeIfMatches = "search_string"
Description: DCD outgoing messages will exclude AVP values that match this rule. The AVP type should be set to "UTF8String".
**Type:** String  
**Optionality:** Optional  
**Allowed:**  
- "search_string" Check for the specified search string anywhere in the string  
- "^search_string$" Check the specified search string matches the whole string  
- "^search_string" Check for the specified search string at the beginning of the string  
- "search_string$" Check for the specified search string at the end of the string  

**Default:** N/A  
**Notes:** If `includeIfMatches` is also defined then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either `excludeIfMatches` or `includeIfMatches`, but not both.  

**Example:**  
```
excludeIfMatches = "f006$"
```

**excludeWhenIn**  
**Syntax:** `excludeWhenIn = "range|list"`  
**Description:** DCD outgoing messages will exclude AVP values that match this rule.  
**Type:** Integer  
**Optionality:** Optional  
**Allowed:**  
- delimited range "x..y" (numbers from x to y)  
- greater than or equal to x ">=x"  
- less than or equal to x "<=x"  
- any of the listed numbers x, y or z "x,y,z"  
- (list may be any length) a single number x "x"  

**Default:** N/A  
**Notes:** The AVP type should be set to "Integer32". If `includeWhenIn` is also defined then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either `excludeWhenIn` or `includeWhenIn`, but not both.  

**Example:**  
```
excludeWhenIn = "1..99"
```

**flags**  
**Syntax:** `flags = number`  
**Description:** What flags to override in the AVP header (as octet value). For example, for M, V bits: flags = 192  
**Type:** Integer  
**Optionality:** Optional (default used if not set).  
**Allowed:**  
**Default:** 0
Notes: If the vendorID parameter value is greater than zero, then V-bit will be set regardless.

Example: flags = 192

**includeIf**

Syntax: includeIf = true|false

Description: DCD outgoing messages will include AVP values that match this rule.

Type: Boolean

Optionality: Optional.

Allowed: true, false

Default: N/A

Notes: The AVP type should be set to "Integer32".

If excludeIf is also defined then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either excludeIf or includeIf, but not both.

Example: includeIf = true

**includeIfMatches**

Syntax: includeIfMatches = "search_string"

Description: DCD outgoing messages will include AVP values that match this rule.

Type: Integer

Optionality: Optional.

Allowed: search_string Check for the specified search string anywhere in the string

^search_string$ Check the specified search string matches the whole string

^search_string Check for the specified search string at the beginning of the string

search_string$ Check for the specified search string at the end of the string

Default: N/A

Notes: The AVP type should be set to "UTF8String".

If excludeIfMatches is also defined then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either excludeIfMatches or includeIfMatches, but not both.

Example: includeIfMatches = "^f003"

**includeWhenIn**

Syntax: includeWhenIn = "range|list"

Description: DCD outgoing messages will include AVP values that match this rule.

Type: Integer

Optionality: Optional.
Chapter 5

 Allowed: delimited range "x..y"
 (numbers form x to y)
greater than or equal to x 
">=x"
less than or equal to x 
"<=x"
any of the listed numbers x, y or z 
"x,y,z"
 (list may be any length)
a single number x "x"
 Default: N/A
 Notes: If includeWhenIn is also defined, then DCD logs a warning that the configuration is inconsistent. Depending on which parameter appears first in the configuration file DCD will load either excludeWhenIn or includeWhenIn, but not both.
 Example: includeWhenIn = "12,14,16,-18,20,22"

interpretBase

 Syntax: interpretBase = number
 Description: Defines the base to use when interpreting numbers that are stored as strings.
 Type: Integer
 Optionality: Optional (default used if not set).
 Allowed:
 - 0 - see notes
 - 8 - octal
 - 10 - decimal
 - 16 - hexadecimal
 Default: 10 (decimal)
 Notes: If the configured value is 0 then strings will be interpreted in the order of decimal constant, octal constant or hexadecimal constant. Any of these may be preceded by a + or a - sign. A:
 - Decimal constant - begins with a non-zero digit and consists of a sequence of decimal digits
 - Octal constant - has a prefix of 0 (zero) followed by a sequence of the digits 0 to 7, or
 - Hexadecimal constant - has a prefix of 0x or 0X followed by a sequence of the decimal digits and letters a (or A) to f (or F).
 Example: interpretBase = 10

literal

 Syntax: literal = "value"
 Description: A literal value.
 Type: String
 Optionality: Optional
 Default: empty
 Notes:
 - In some cases where we use the AVP definition to create a request message, we may wish to simply specify a literal value rather than obtain the information from a ccsConcept field. In these cases we specify the value here, in string form, and it will be converted to the required type when requested from the configuration AVP object.
 - This parameter was previously "value".

62  NCC Diameter Charging Driver Technical Guide
Example: \[
\text{literal} = "1"
\]

**mandatoryContents**

**Syntax:** \[\text{mandatoryContents} = "avps"\]

**Description:** A simple string array, applicable to AVPs of type “Grouped” and those with no type, specifying the AVPs (by name) that must be included in this AVP.

**Type:** String array

**Optionality:** Optional

**Default:** empty

**Example:**
\[
\text{mandatoryContents} = \[
\quad "\text{Subscription-Id-Type}\"
\quad "\text{Subscription-Id-Data}\"
\]
\]

**maxOccurrences**

**Syntax:** \[\text{maxOccurrences} = \text{value}\]

**Description:** The number of repeating AVPs (whether a group or an individual field).

It does not actually limit how many AVPs can be processed; it limits how the AVP is physically ‘unloaded’ into a contiguous range of profile variables.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

<table>
<thead>
<tr>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The default value means that all AVP values are unloaded into the same configured profile block/tag (admittedly not very useful, but this ensures backwards compatibility).</td>
</tr>
</tbody>
</table>

**Notes:** To unload an array of AVPs into a *series* of profile tag variables

maxOccurrences is set to the number of different values to be captured.

The first will be unloaded into the profile tag configured for the AVP.

The second will be unloaded into that tag+1, and so on.

**Note:** When unloading a repeating group the specially-defined variable ‘index’ can be obtained by coding an AVP with variable="index". This takes the value of the current loop index.

**Example:**
\[
\text{maxOccurrences} = 10
\]

**name**

**Syntax:** \[\text{name} = "name"\]

**Description:** The name of the AVP, which can be used by the code for direct retrieval, as well as logging and debug messages.

**Type:** String

**Optionality:** Mandatory

**Example:**
\[
\text{name} = "CC-Money"
\]

**octetLength**

**Syntax:** \[\text{octetLength} = \text{value}\]

**Description:** How many bytes to use for integer quantities, if the type is “OctetString”.

**Type:** Integer

**Optionality:** Optional
Allowed: Should be a number 0 through 4.
Notes: 0 is a special case meaning encode the value as an ASCII string, and determine the number of bytes from the string size. For responses the values 1-4 simply mean treat as integer encoded, and use the number of bytes given.
Example: octetLength = 0

optionalContents
Syntax: optionalContents = "avps"
Description: A simple string array, applicable to AVPs of type “Grouped” and those with no type, specifying the AVPs (by name) that may be included in this AVP
Type: String
Optionality: Optional
Default: empty
Example: optionalContents = [
    "Service-Identifier",
    "Requested-Service-Unit",
    "Subscription-Id"
]

profileBlock
Syntax: profileBlock = num
Description: The profile block
Type: Integer
Optionality: Used only if the ccsConcept field is a profile variable.
Allowed: The value given for this is a number, and must be valid for a profile block (that is, in the range 0 through 18).
Notes: This parameter is used to identify the profile block it will be stored into/retrieved from. See also the related parameters, profileTag and profileFormat.
Example: profileBlock = 9

profileFormat
Syntax: profileFormat = "format"
Description: The format of the profile.
Type: String
Optionality: Used only if the ccsConcept field is a profile variable.
Allowed: The value given for this must be one of the valid storage formats for ACS profile fields. The allowable values for this parameter are:
- STRING
- NSTRING
- LNSTRING
- INTEGER
- UNSIGNED64
- RAW
- TIME
- BOOLEAN
- ARRAY
Default: INTEGER
Notes: This parameter is used to identify its storage format. See also the related
parameters, profileBlock and profileTag.

Example:
profileFormat = "LNSTRING"

profileTag
Syntax: profileTag = num
Description: The profile tag.
Type: Integer
Optionality: Used only if the ccsConcept field is a profile variable.
Notes: This parameter is used to identify the profile tag it will be stored into/retrieved from. See also the related parameters, profileBlock and profileFormat.
Example: profileTag = 999

repeating
Syntax: repeating = true|false
Description: Allows the configuration-driven code to recognize that there may be a number of repeating instances of this AVP in its containing group, not just one.
Type: Boolean
Optionality: Mandatory
Allowed: true, false
Notes: In the building of a request, repeating AVPs are added until the getConcept call indicates no more concept data is available. In the unpacking of a response, repeating AVPs are extracted (and setConcept calls made) until the Diameter stack indicates there are no more to retrieve.
Example: repeating = true

type
Syntax: type = "type"
Description: The type of AVP
Type: String
Optionality: There is a scenario where type may be omitted altogether, strange as it may seem. When defining the AVPs that make up a request message, you can list them as mandatory or optional contents in an AVP that has no other information. This means the contents should be directly placed into the request message without (for example) an intervening group.
Allowed: This is a string value that, if specified, must be:
  • Either one of the base types described in the Diameter RFC 3588 specification:
    ▪ OctetString
    ▪ Integer32
    ▪ Integer64
    ▪ Unsigned32
    ▪ Unsigned64
    ▪ Grouped
    ▪ Address
    ▪ Time
    ▪ UTF8String
    ▪ DiameterIdentity
DiameterURI

Enumerated

Or the name of any previously defined AVP in the configuration. The latter facility allows us to relate two CCS concept fields to the same base type without having to repeat the full definition of that base type. When an AVP object is defined to be of the same type as a preceding AVP object in this way, it will inherit all the attributes of that base type except its name, repeating attribute and its ccsConcept.

Notes: Note that Float32 and Float64, although defined in RFC 3588, are not supported, as CCS does not use floating point values.

Example: 

vendorId

Syntax: vendorId = ID

Description: A number that identifies the vendor ID of the corresponding Diameter AVP.

Type: Integer

Optionality: Optional

Default: 0

Notes: It will be used whenever we have to insert this AVP into a request message.

Example: vendorId = 0

Example cdrTag/Operation configuration

The cdrTag and cdrOperation parameters can be used by any AVP/ccsConcepts pairing.

The following is just an example of how to configure these parameters.

```json
{
  # This 'AVP' simply defines the list of AVPs for a direct named event reservation.

  # Give this AVP a position in the configuration 'tree'. This is just
  # a label to allow representation of the tree to the actions library.
  name = "DirectNamedEventRequest"

  # Define the list of AVPs (by name) for this node. Note that the AVPs are at the
  # sibling level for this node if there are no type or avpCode parameters for this
  # node.
  # An error will occur if mandatory contents are not available, but non-present
  # contents are silently ignored.
  mandatoryContents = [
    "Service-Identifier"
    "ChargingMaxEventClassAndEventName"
  ]

  optionalContents = []
}
```

```json
{
  avpCode = 13000
  name = "ChargingMaxEventClassAndEventName"
  type = "Grouped"
  mandatoryContents = [
    "CMX-eventClass"
    "CMX-eventName"
    "DIA-Service-Identifier"
  ]
}
```
null

```python
optionalContents = []

{
    avpCode = 13001
    name = "CMX-eventClass"
    ccsConcept = "eventClass"
    type = "UTF8String"
    cdrTag = "CMX_EC"
    cdrOperation = "append"
}

{
    avpCode = 13002
    name = "CMX-eventName"
    ccsConcept = "eventName"
    type = "UTF8String"
    cdrTag = "CMX_EN"
    cdrOperation = "leave"
}

# This 'AVP' represents the RFC 4006 Service-Identifier.
    avpCode = 439
    name = "DIA-Service-Identifier"
    type = "Unsigned32"
    ccsConcept = "eventType"
    cdrTag = "DIA_SI"
    cdrOperation = "replace"
... 
```

**ContextCopy parameters**

The ContextCopy section defines the profiles in which DCD call context data may be copied to at call time. This is normally for the purposes of control plan branching.

**Example ContextCopy section**

Here is an example of the ContextCopy section.

```
ContextCopy = [
    {
        contextItem = "scpActionSupervise"
        profileBlock = 17
        profileTag = 6356992 # Hex 0x00610000
        profileFormat = "INTEGER"
    }
    {
        contextItem = "scpActionDoNotSupervise"
        profileBlock = 17
        profileTag = 6356993 # Hex 0x00610001
        profileFormat = "INTEGER"
    }
    {
        contextItem = "scpActionRelease"
        profileBlock = 17
        profileTag = 6356994 # Hex 0x00610002
        profileFormat = "INTEGER"
    }
    {
        contextItem = "scpActionSendMessage"
        profileBlock = 17
        profileTag = 6356995 # Hex 0x00610003
    }
```
profileFormat = "INTEGER"
}
{
  contextItem = "scpActionPlayAnnouncement"
  profileBlock = 17
  profileTag = 6356996 # Hex 0x0061004
  profileFormat = "INTEGER"
}
{
  contextItem = "scpActionSuperviseWithoutControlling"
  profileBlock = 17
  profileTag = 6356997 # Hex 0x0061005
  profileFormat = "INTEGER"
}
{
  contextItem = "callState"
  profileBlock = 17
  profileTag = 6356998 # Hex 0x0061006
  profileFormat = "INTEGER"
}
{
  contextItem = "sendCount"
  profileBlock = 17
  profileTag = 6356999 # Hex 0x0061007
  profileFormat = "INTEGER"
}
{
  contextItem = "preCallAnnouncementId"
  profileBlock = 17
  profileTag = 6357000 # Hex 0x0061008
  profileFormat = "INTEGER"
}
{
  contextItem = "preCallLowBalance"
  profileBlock = 17
  profileTag = 6357001 # Hex 0x0061009
  profileFormat = "INTEGER"
}

contextItem

Syntax: contextItem = "name"

Description: The name of the DCD context item from the allowed list.

Type: String

Optionality: Optional

Allowed:
- "scpActionSupervise"
- "scpActionDoNotSupervise"
- "scpActionRelease"
- "scpActionSendMessage"
- "scpActionPlayAnnouncement"
- "scpActionSuperviseWithoutControlling"
- "callState"
- "sendCount"
- "preCallAnnouncementId"
- "preCallLowBalance"

Default:

Notes: All the supported context items are listed in the example.
Example: \texttt{contextItem = "scpActionSupervise"}

**profileBlock**

**Syntax:** \texttt{profileBlock = val}

**Description:** The profile block to use.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:**

**Notes:** Examples all use 17 (temporary storage)

**Example:** \texttt{profileBlock = 17}

**profileFormat**

**Syntax:** \texttt{profileFormat = "format"}

**Description:** The profile tag format

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:**

"INTEGER"

"STRING"

"TIME"

**Default:**

**Notes:**

**Example:** \texttt{profileFormat = "INTEGER"}

**profileTag**

**Syntax:** \texttt{profileTag = decival}

**Description:** The profile tag in which to store data.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:**

**Notes:** Example tag values are all in the DCD range (0x0061nnnn), but will need to be configured in ACS and Prepaid Charging before they are available for use. In the example, 6356992 is value of Hex 0x00610000

**Example:** \texttt{profileTag = 6356992}

**TimeIn and TimeOut**

TimeIn and TimeOut are used to copy a timestamp at the beginning of a call (TimeIn), and recording elapsed time during a call (TimeOut). If present, the timestamps will be stored in the configured profile fields.

- **TimeIn** - the time the CCA was received by the SLC from the VWS
- **TimeOut** - the time the CCR was sent from the SLC to the VWS

These parameters are optional - if omitted, no timestamps will be recorded.
Example tag values are all in the DCD range (0x0061nnnn), but will need to be configured in ACS and Prepaid Charging before they are available for use.

The parameters are defined in the *ContextCopy parameters* (on page 67) section.

**Example TimIn and TimeOut**

Here are the example parameters.

```plaintext
TimeIn = {
  profileBlock = 17
  profileTag = 6357002 # Hex 0x006100a, continuing from example ContextCopy values
  profileFormat = "TIME"
}
TimeOut = {
  profileBlock = 17
  profileTag = 6357003 # Hex 0x006100b
  profileFormat = "TIME"
}
```

**PeerSchemes Configuration Section**

**Example PeerSchemes**

Here is a high level structure of the configuration of a scheme in the *PeerSchemes section.*

*Note: The PeerSchemes or Peers section is mandatory.*

```plaintext
PeerSchemes = [
  {
    schemeName = "SchemeA"
    Peers = [
      {
        name = "host1"
        scheme = [ "scheme1", "scheme2" ]
        permittedOriginHosts = [ "host1.realm1.oracle.com"
        peer_group = "host1"
        transport = "tcp"
        initiation = "connect"
        RemoteAddresses = [ "192.168.1.10"
        remote_port = 3868
        netmaskBits = 32
        permittedInstances = 0
        reqSctpInboundStreams = 8
        reqSctpOutboundStreams = 8
        sctp_hbinterval = 1000
      }
    }
  }
]```
watchdogPeriod = 30
connectionTimeout = 30
inBufferSize = 0
outBufferSize = 0
}

# end of Peer host1
{
  Peer_Host2_Parameters
}

# End of Scheme A
{
schemeName = "SchemeB"
Peers = [
  Parameters_for_SchemeB_peers
]

# End of Scheme B
} # End of PeerSchemes section

schemeName

Syntax: schemeName = "name"

Description: The name identifying the scheme.

Type: String

Optionality: Mandatory

Example: schemeName = "SchemeA"

Peer host parameters

The following parameters are used for a peer host. They are found within the Peers section.

Note: The PeerSchemes or Peers section is mandatory.

The available parameters are:

[
  name = "host1"
  scheme = [ "scheme1", "scheme2" ]
  permittedOriginHosts = [
    "host1.realm1.oracle.com"
  ]
  peer_group = "host1"
  transport = "tcp"
  initiation = "connect"
  RemoteAddresses = [
    "192.168.1.10"
  ]
}
remote_port = 3868
netmaskBits = 32
permittedInstances = 0
reqSctpInboundStreams = 8
reqSctpOutboundStreams = 8
sctp_hbinterval = 1000
watchdogPeriod = 30
connectionTimeout = 30
inBufferSize = 0
outBufferSize = 0
}

Note: All the peer configuration items from "transport" on down can also have global defaults set in the DIAMETER.DCD section. See DCD (on page 36).

connectionTimeout
Syntax: connectionTimeout = timeout
Description: The timeout for re-establishing connections (RFC 3588 Tc).
Type: Integer
Optionality: Optional
Allowed: in seconds
Default: 30
Example: connectionTimeout = 30

inBufferSize
Syntax: inBufferSize = size
Description: The size of the socket receive buffer.
Type: Integer
Optionality: Mandatory
Allowed: in bytes
Default: 0 (means to use the OS default)
Example: inBufferSize = 0

initiation
Syntax: initiation = "action"
Description: How to start the connection.
Type: String
Optionality: Mandatory
Allowed: 
  - listen = listen for incoming connections
  - connect = connect an outgoing connection.
Example: initiation = "connect"
name
  Syntax: \texttt{name = "name"}
  Description: The name identifying either peer or group of peers.
  Type: String
  Optionality: Mandatory
  Example: \texttt{name = "host1"}

\texttt{netmaskBits}
  Syntax: \texttt{netmaskBits = bits}
  Description: The number of bits for netmask.
  Type: Integer
  Optionality: Mandatory
  Default: 32 (bits for netmask, that is, a single machine (/32))
  Example: \texttt{netmaskBits = 32}

\texttt{outBufferSize}
  Syntax: \texttt{outBufferSize = size}
  Description: The size of the socket send buffer.
  Type: Integer
  Optionality: Mandatory
  Allowed: in bytes
  Default: 0 (means to use the OS default)
  Example: \texttt{outBufferSize = 0}

\texttt{peer_group}
  Syntax: \texttt{peer_group = "name"}
  Description: The peer group we are a part of.
  Type: String
  Optionality: Optional
  Default: Defaults to the name.
  Example: \texttt{peer_group = "host1"}

\texttt{permittedInstances}
  Syntax: \texttt{permittedInstances = number}
  Description: The number of permitted instances.
  Type: Integer
  Optionality: Mandatory
  Notes: If set to 0 then allow all.
  Example: \texttt{permittedInstances = 0}

\texttt{permittedOriginHosts}
  Syntax: \texttt{permittedOriginHosts = ["host"]}
  Description: The list of peer names that will be checked against the OriginHost AVP during the capabilities exchange.
  Type: String
Chapter 5

Optionality: Mandatory
Notes: This parameter accepts at least one host and will accept any one of them when it gets the Capabilities Exchange Answer. These are the hosts allowed to talk to this client.
Example:

```
permittedOriginHosts = [
    "host1.realm1.oracle.com",
    "host2.realm1.oracle.com"
]
```

remote_port
Syntax: `remote_port = number`
Description: The remote post number.
Type: Integer
Optionality: Optional
Default: Defaults to the RFC specified 3868
Example: `remote_port = 3868`

RemoteAddresses
Syntax: `remoteAddresses = ["ipaddress"]`
Description: The list of Remote IP addresses.
Type: Array of string parameters
Optionality: Mandatory
Notes: If an address becomes unavailable the list will be cycled through.
Example:

```
remoteAddresses = [
    "192.168.1.10"
]
```

reqSctpInboundStreams
Syntax: `reqSctpInboundStreams = number`
Description: The number of requested inbound sctp streams.
Type: Integer
Optionality: Mandatory
Notes: There is no guarantee you will actually get these.
Example: `reqSctpInboundStreams = 8`

reqSctpOutboundStreams
Syntax: `reqSctpOutboundStreams = number`
Description: The number of requested outbound sctp streams.
Type: Integer
Optionality: Mandatory
Notes: There is no guarantee you will actually get these.
Example: `reqSctpOutboundStreams = 8`

scheme
Syntax: `scheme = ["scheme", "scheme"]`
Description: The list of schemes to which you have limited the peer.
Type: Array of strings
Optionality: If using PeerSchemes this is mandatory.
Allowed: Names of configured schemes.
Example: `scheme = [ "scheme1", "scheme2" ]`

`sctp_hbinterval`
Syntax: `sctp_hbinterval = interval`
Description: The interval for sctp heartbeats.
Type: Integer
Optionality: Optional
Allowed: in milliseconds
Default: 1000
Example: `sctp_hbinterval = 1000`

`transport`
Syntax: `transport = "type"`
Description: The protocol for this host peer.
Type: String
Optionality: Optional
Allowed:
- `sctp`
- `tcp`
Default: If not specified, then it uses the one from the global DCD section.
Example: `transport = "tcp"`

`watchdogPeriod`
Syntax: `watchdogPeriod = period`
Description: The quiet period before sending a DWR. (RFC 3588 Tw).
Type: Integer
Optionality: Mandatory
Allowed: in seconds
Default: 30
Example: `watchdogPeriod = 30`

## Routes

### Introduction

The Routes section is used to specify the routing configuration for the BeClient, that is, how to select a peer of the realm.

### Routes parameters

Here is an example routes section of the DIAMETER section of the eserv.config file.
```
routes = [
    {
        realm = "FirstRealm"
        host = "host1.realm1.oracle.com"
    }
]"
direct
Syntax: direct = true|false
Description: Whether this is a direct server connection, or if a proxy/agent is used.
Type: Boolean
Optionality: Optional
Allowed: true, false
Default: true
Notes: May be specified for each host. See HostSpecificData parameters (on page 78).
Example: direct = true

host
Syntax: host = "permitted_origin_host"
Description: The host name of the next-hop as configured in DCD.Peers.permittedOriginHosts for the relevant peer.
Type: String
Optionality: Mandatory
Allowed: This is the permittedOriginHosts value of the peer.
May be specified for each host. See HostSpecificData parameters (on page 78).
Example: host = "host1.realm1.oracle.com"

priority
Syntax: priority = priority
Description: The priority of the route.
Type: Integer
Optionality: Optional
Notes: Only those routes with the lowest priority are used.
May be specified for each host. See HostSpecificData parameters (on page 78).
Example: priority = 1

realm
Syntax: realm = = name
Description: The Realm identity.
Type: String
Optionality: Mandatory
May be specified for each host. See HostSpecificData parameters (on page 78).
Example: realm = "FirstRealm"
round_robin
Syntax: \( \text{round\_robin} = \text{weight} \)
Description: The weight for round\_robin selection.
Type: Integer
Optionality: Optional
Default: 0
Notes: Zero indicates a failover type selection. See routing (on page 47) for details.
May be specified for each host. See HostSpecificData parameters (on page 78).
Example: \( \text{round\_robin} = 0 \)

NamedEventTypes

Introduction

The NamedEventTypes section is used to define a mapping from the Prepaid Charging descriptor of a named event (the eventClass and eventName) to a single integer (eventType) for availability to the Service-Identifier AVP.

NamedEventTypes parameters

Here is an example of the NamedEventTypes section.

```plaintext
NamedEventTypes = [
    { 
        eventClass = "abc"
        eventName = "def"
        eventType = 123
        isDebit = true
    },
    { 
        eventClass = "ghi"
        eventName = "jkl"
        eventType = 456
    }
]
```

eventClass
Syntax: \( \text{eventClass} = \text{"class"} \)
Description: The event class.
Type: String
Optionality: Optional
Example: \( \text{eventClass} = \text{"abc"} \)

eventName
Syntax: \( \text{eventName} = \text{"name"} \)
Description: The event name.
Type: String
Optionality: Optional
Chapter 5

Example: eventClass = "def"

eventType
Syntax: eventType = type
Description: The event type for availability to the Service-Identifier AVP.
Type: integer
Optionality: Optional
Allowed: Mapping to agreed event type with server vendor.
Example: eventType = 123

isDebit
Syntax: isDebit = true|false
Description: Whether this named event represents a debit or credit for the subscriber.
Type: Boolean
Optionality: Optional
Allowed: true, false
Default: true
Example: isDebit = true

HostSpecificData

Introduction
The following is an optional section, with members defined according to hostname. The purpose is to represent data specific to particular hosts. This permits a common configuration file to be deployed to multiple machines.

Settings here should override global settings, for the specified host only. See DCD parameters (on page 36) and Routes parameters (on page 75) for global settings of these parameters.

HostSpecificData parameters
Here is an example of the HostSpecificData parameters.

```plaintext
HostSpecificData = [
    {  
        name = "ocpc.oracle.com"
    }

    DCD = {  
        Origin-Host = "ocpc.oracle.com"
        Origin-Realm = "ocpc.oracle.com"
    }

    routes = [  
        {  
            realm = "myDomainA"
            host = "host1.realm1.oracle.com"
            priority = 1  
            round_robin = 0
```
direct = true

host

name

Origin-Host
Chapter 5

Origin-Realm
Syntax: Origin-Realm = "diameterId"
Description: The Diameter AVP Origin-Realm.
Type: String
Optionality: Optional (default used if not set).
Allowed: 
Default: 
Notes: See DCD parameters (on page 36) for the global setting.
Example: Origin-Realm = "ocpc.oracle.com"

priority
Syntax: priority = priority
Description: This is the priority of the route.
Type: Integer
Optionality: Optional
Notes: Only those routes with the lowest priority are used.
See Routes parameters (on page 75) for the global setting.
Example: priority = 1

realm
Syntax: realm = "realmname"
Description: The realm name, as configured in the Prepaid Charging screens.
Type: String
Optionality: Mandatory
Allowed: 
Default: 
Notes: See Routes parameters (on page 75) for the global setting.
Example: realm = "myDomainA"

round_robin
Syntax: round_robin = weight
Description: This is the weight for round_robin selection.
Type: Integer
Optionality: Optional
Default: 0
Notes: Zero indicates a failover type selection. See routing (on page 47) for details.
See Routes parameters (on page 75) for the global setting.
Example: round_robin = 0

Statistics logged by diameterBeClient

Introduction
Diameter statistics are generated by each SLC, and then transferred at periodic intervals to the Service Management System (SMS) for permanent storage and analysis.
An existing statistics system (smsStats) provides functions for the collection of basic statistical events. This is provided in the NCC SMS application. Refer to *SMS Technical Guide* for details.

**Enabling statistics**

Follow these steps to enable statistics on an SCP after installing the database entries on the SMF.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the Table Replication tab of the SMS Node Management screen, select the DCD replication entry:  
      SMS->SMF_STATISTICS_DEFN->SMF_STDEF_DCD  
      and drag it over to the allocated Replication Groups. Refer to *Configuring Table Replication* in the *SMS User Guide* for details. |
| 2    | After creating the config file, you need to send a HUP to the smsStatsDaemon to force a reread of the database entries:  
      # fuser -s 1 /IN/service_packages/SMS/bin/smsStatsDaemon |

**DCD statistics**

SMS statistics are logged with APPLICATION_ID = 'DCD' (application number 97)

The following statistics are defined:

- `DIAMETER_MIN_LATENCY` - Minimum Latency
- `DIAMETER_MAX_LATENCY` - Maximum Latency
- `DIAMETER_AVERAGE_LATENCY` - Average Latency
- `DIAMETER_CC_TYPE_INITIAL` - Initial Requests sent
- `DIAMETER_CC_TYPE_UPDATE` - Update Requests sent
- `DIAMETER_CC_TYPE_TERMINATION` - Termination Requests sent
- `DIAMETER_CC_TYPE_EVENT` - Event Requests sent
- `DIAMETER_TIMEOUT` - CCRs timed out
- `DIAMETER_SUCCESS` - CCAs received with success result code
- `DIAMETER_FAILURE_3xxx` - CCAs received, error code in range 3000 to 3999
- `DIAMETER_FAILURE_4xxx` - CCAs received, error code in range 4000 to 4999
- `DIAMETER_FAILURE_5xxx` - CCAs received, error code in range 5000 to 5999
- `DIAMETER_FAILURE_1xxx` - CCAs received, error code in range 1000 to 1999
- `DIAMETER_FAILURE_UNKNOWN` - CCAs received, error code in undefined range

For all statistics, the Destination-Realm or Host ID involved is put into SMF_STATISTICS.DETAIL.

**Reports**

The following reports are available:

- DCD System Stats
- DCD System Stats by Realm/Host

Reports are generated using the SMS Report Functions screen. Refer to *SMS User’s Guide* for details.

**Example report**

Here is an example DCD System Stats report.

**DCD Statistics Listing**
Chapter 5

Start Date: 15 October 2007
Finish Date: 30 October 2007
Report Type: Totals

24 October 2007, 21:20:12

<table>
<thead>
<tr>
<th>Node Name</th>
<th>Statistics ID</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_FAILURE_UNKNOWN</td>
<td>3</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_SUCCESS</td>
<td>319</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_FAILURE_5xxx</td>
<td>14</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_CC_TYPE_INITIAL</td>
<td>214</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_FAILURE_1xxx</td>
<td>2</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_TIMEOUT</td>
<td>63</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_CC_TYPE_UPDATE</td>
<td>185</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_FAILURE_3xxx</td>
<td>8</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_CC_TYPE_TERMINATION</td>
<td>86</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_CC_TYPE_EVENT</td>
<td>14</td>
</tr>
<tr>
<td>mtv-tst-scp10</td>
<td>DIAMETER_FAILURE_4xxx</td>
<td>39</td>
</tr>
</tbody>
</table>

Completed

**DCD EDRs**

**EDR generation**

EDRs are generated and processed by the slec_acs on the SLC and uploaded at regular intervals to the SMS using the cmnPushFiles process.

Diameter Charging Driver (DCD) tags are appended to the EDRs generated by the Advanced Control Services application. Refer to *NCC Event Detail Record Reference Guide*, topic ACS EDR tags for the full list and descriptions.

**DCD EDR tags**

Here are the EDR tags produced by DCD.

**DIA_RC (result code)**

*Description:* Number indicating diameter result-code received in CCA message.

*Format:* Integer

*Concept:* Result-Code

*Notes:* For Diameter event based messages, this will always be 0, and hence not recorded.

*Example:* DIA_RC=2001

**DIA_REQ (current session message number)**

*Description:* Sequential number, indicating message within the current session.

*Format:* Integer

*Concept:* CC-Request-Number

*Notes:* For Diameter event based messages, this will always be 0, and hence not recorded.

*Example:* DIA_REQ=1
DIA_SID (session id)
Description: This is a unique value identifying the Diameter session.
Format: Of the form:
\[\text{DiameterIdentity};\text{time};\text{SLEE CallID}\]
Where:
- \(\text{DiameterIdentity}\) is that of the SLC (that is, the Origin-Host used in the CCR message)
- \(\text{time}\) is the time of the first request (expressed as the number of seconds since the Unix epoch time)

Concept: Session-ID
Notes:
Example: \(\text{DIA_SID=scp1.oracle.com;1201809603;1394074}\)

DIA_TIME (time ccr sent)
Description: The time the CCR was sent, in hundredths of second
Format: Date - “YYYY-MM-DD-HH-MM-SSSS”
Concept: Session-ID
Notes:
Example: \(\text{DIA_TIME=2008-03-27-20-41-3831}\)

Custom tag names

The \text{cdrTag} configuration parameter allows for an EDR to have tag names customer defined.

Example EDRs

Here are some example EDRs generated by DCD.
Refer to \text{ACS EDR Tags} for the non-DCD tags.

Example 1
Whole EDR for an InitialTimeReservation and ConfirmTimeReservation:

\text{EDR:} 'VOICE_MO|CID=285222|OA=0|OTI=0|CUST=1|SN=07776666444|TN=07776666444|CGN=8888887|CLI=8888887|SK=1|TCS=20080327204138|TCE=20080327204241|LPN=|LAC=|CS=4|CPC=10|CC=|CPNI=0|PCNA=|PTNA=|CGNA=|TFN=ST-2,SDTN-21,uatb-3,END-14|LGID=0|CPN=uatbWsceBrch|CAET=3|CCET=60.0|CA=60777666555|RELC=17|OCPI=CPNN=3|CGNN=3|CPPI=1|NOAT=1|CBAT=0|FATS=0|CCET=20080327204138|HTS=20080327204138|AIDL=\|DIA_SID= nzwn-test03-z2;47ec0682;45a26\|DIA_REQ=0\|DIA_RC=2001\|DIA_TIME=2008-03-27-20-41-3831\|DIA_SID=nzwn-test03-z2;47ec0682;45a26\|DIA_REQ=1\|DIA_RC=2001\|DIA_TIME=2008-03-27-20-41-3847\|FCA=60777666555|WALR=86' Note the DCD part of the EDR. All four tags are present twice:

- The Initial Time Reservation
  \(\text{DIA_SID=nzwn-test03-z2;47ec0682;45a26}\|\text{DIA_REQ=0}\|\text{DIA_RC=2001}\|\text{DIA_TIME=2008-03-27-20-41-3831}\|

- The Termination Time Reservation
  \(\text{DIA_SID=nzwn-test03-z2;47ec0682;45a26}\|\text{DIA_REQ=1}\|\text{DIA_RC=2001}\|\text{DIA_TIME=2008-03-27-20-41-3847}\|

Example 2
Whole EDR for DirectNamedEvent:
EDR:

`VOICE_MO|CID=287224|OA=0|OTI=0|CUST=1|SN=0777666444|TN=|CGN=8888887|CLI=8888887|SK=1|TCS=20080327231115|TCE=0|LPN=|LAC=|CS=1|CPC=10|CC=|CPNI=0|PCNA=|TPNA=|CGNA=|TGNA=|TFN=ST-1,bevt-2,END-3|LGID=0|CPN=DirectDebit|CAET=0|CCET=0.0|CA=|REL=31|OCPI=|CPNN=3|CGNN=3|CPPI=1|NOAT=0|CBAT=0|FATS=0|CCTS=0|HTS=0|AIDL=|DIA_SID=nzwn-test03-z2;47ec2993;461f8|DIA_RC=2001|DIA_TIME=2008-03-27-23-11-1577'

Note in the DCD part of the EDR that event based EDRs only need to receive DIA_SID, DIA_RC and DIA_TIME, that is, no DIA_REQ:

```
DIA_SID=nzwn-test03-z2;47ec2993;461f8|DIA_RC=2001|DIA_TIME=2008-03-27-23-11-1577'
```

Example 3

For cdrTag for MMM_TAG and ZZZ_TAG, the following would be an example of what the resulting EDR would look like.

```
CCS_BE|CID=205383|OA=0|OTI=0|CUST=1|SN=1130|TN=|CGN=0212994768|CLI=0212994768|SK=3|TCS=20091117192600|LPN=|LAC=|CS=1|CPC=10|CC=|CPNI=0|PCNA=|TPNA=|CGNA=|TGNA=|TFN=ST-1,CCDR-8,CCDR-12,CCDR-14,bevt-2,DISC-3,END-7|LGID=0|CPN=ST-BE-END|OCPI=|CPNN=3|CGNN=3|CPPI=1|NOAT=0|CBAT=0|FATS=0|CCTS=0|HTS=0|AIDL=|AAA_TAG=11111|CMX_EC=CR96791|CMX_EN=BasicTest|DIA_SID=eng-host06-z6.usp.co.nz;4b02f8c8;32247|DIA_RC=2001|DIA_TIME=2009-11-17-19-26-0062|MMM_TAG=55555|ZZZ_TAG=777
```
Chapter 6

About Installation and Removal

Overview

Introduction

This chapter provides details of the installation and removal process for the application.

In this chapter

This chapter contains the following topics.

Installation and Removal Overview

Checking the Installation

Installation and Removal Overview

Introduction

For information about the following requirements and tasks, see *NCC Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

DCD packages

An installation of DCD includes the following packages, on the:

- SMS:
  - dcdSms
- SLC:
  - dcdScp

Checking the Installation

Introduction

Refer to these check lists to ensure the package has been installed correctly.

DCD SLC directories and files

The DCD installation on the SLC creates the following directories:

- /IN/service_packages/DCD/bin
- /IN/service_packages/DCD/etc
- /IN/service_packages/DCD/lib
- /IN/service_packages/DCD/tmp
The DCD installation installs the following binaries and interfaces:

- /IN/services_packages/DCD/bin/diameterBeClient

The DCD installation installs the following example configuration file:

- /IN/services_packages/DCD/etc/eserv.config.dcd.example

The DCD installation installs the following shared library:

- /IN/services_packages/DCD/lib/diamActions.so

**DCD SMS directories**

Check that the statistics and control plans have been installed correctly.

The DCD installation on the SMS creates the following directories:

- /IN/service_packages/DCD/db
- /IN/service_packages/DCD/lib
NCC Glossary of Terms

**AAA**

**ACS**
Advanced Control Services configuration platform.

**AVP**
Attribute Value Pair, used in Diameter to represent properties of a particular request or answer.

**BE**
Billing Engine

**CC**
Country Code. Prefix identifying the country for a numeric international address.

**CCA**
Credit-Control-Answer, used in Diameter by the credit-control server to acknowledge a Credit-Control-Request (CCR) from the credit-control client.

**CCR**
Credit-Control-Request, used in Diameter by the credit-control client to request credit authorization from the credit-control server.

**CCS**
1) Charging Control Services (or Prepaid Charging) component.
2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.

**CEA**
Peer message: Capabilities Exchange Answer

**CER**
Peer message: Capabilities Exchange Request

**cron**
Unix utility for scheduling tasks.

**Diameter**
A feature rich AAA protocol. Utilises SCTP and TCP transports.
DP
Detection Point

DPR
Peer message: Disconnect Peer Request

DTMF
Dual Tone Multi-Frequency - system used by touch tone telephones where one high and one low frequency, or tone, is assigned to each touch tone button on the phone.

DWR
Peer message: Device Watchdog Request

EDR
Event Detail Record

Note: Previously CDR. The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the NCC documentation.

GSM
Global System for Mobile communication.

It is a second generation cellular telecommunication system. Unlike first generation systems, GSM is digital and thus introduced greater enhancements such as security, capacity, quality and the ability to support integrated services.

HLR
The Home Location Register is a database within the HPLMN (Home Public Land Mobile Network). It provides routing information for MT calls and SMS. It is also responsible for the maintenance of user subscription information. This is distributed to the relevant VLR, or SGSN (Serving GPRS Support Node) through the attach process and mobility management procedures such as Location Area and Routing Area updates.

HTML
HyperText Markup Language, a small application of SGML used on the World Wide Web.

It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational and presentational items, and some hypertext and multimedia.

IDP
INAP message: Initial DP (Initial Detection Point)

IN
Intelligent Network

INAP
Intelligent Network Application Part - a protocol offering real time communication between IN elements.
**Initial DP**

Initial Detection Point - INAP Operation. This is the operation that is sent when the switch reaches a trigger detection point.

**IP**

1) Internet Protocol
2) Intelligent Peripheral - This is a node in an Intelligent Network containing a Specialized Resource Function (SRF).

**IP address**

Internet Protocol Address - network address of a card on a computer

**ISDN**

Integrated Services Digital Network - set of protocols for connecting ISDN stations.

**IVR**

Interactive Voice Response - systems that provide information in the form of recorded messages over telephone lines in response to user input in the form of spoken words or, more commonly, DTMF signalling.

**MAP**

Mobile Application Part - a protocol which enables real time communication between nodes in a mobile cellular network. A typical usage of the protocol would be for the transfer of location information from the VLR to the HLR.

**Messaging Manager**

The Messaging Manager service and the Short Message Service components of Oracle Communications Network Charging and Control product. Component acronym is MM (formerly MMX).

**MIN**

Mobile Identification Number, also known as an MSID.

**MM**

Messaging Manager. Formerly MMX, see also XMS (on page 91) and Messaging Manager (on page 89).

**MSID**

Mobile Subscriber Identification, also known as an MIN.

**MSISDN**

Mobile Station ISDN number. Uniquely defines the mobile station as an ISDN terminal. It consists of three parts; the country code (CC), the national destination code (NDC) and the subscriber number (SN).
Peer
Remote machine, which for our purposes is capable of acting as a Diameter agent.

PIN
Personal Identification Number

SCP
Service Control Point. Also known as SLC.

SCTP
Stream Control Transmission Protocol. A transport-layer protocol analogous to the TCP or User Datagram Protocol (UDP). SCTP provides some similar services as TCP (reliable, in-sequence transport of messages with congestion control) but adds high availability.

Session
Diameter exchange relating to a particular user or subscriber access to a provided service (for example, a telephone call).

SGML

SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

SMS
Depending on context, can be:
- Short Message Service
- Service Management System platform
- NCC Service Management System application

SN
Service Number
SRF
Specialized Resource Function - This is a node on an IN which can connect to both the SSP and the SLC and delivers additional special resources into the call, mostly related to voice data, for example play voice announcements or collect DTMF tones from the user. Can be present on an SSP or an Intelligent Peripheral (IP).

SSP
Service Switching Point

TCP
Transmission Control Protocol. This is a reliable octet streaming protocol used by the majority of applications on the Internet. It provides a connection-oriented, full-duplex, point to point service between hosts.

USSD
Unstructured Supplementary Service Data - a feature in the GSM MAP protocol that can be used to provide subscriber functions such as Balance Query and Friends and Family Access.

VLR
Visitor Location Register - contains all subscriber data required for call handling and mobility management for mobile subscribers currently located in the area controlled by the VLR.

VWS
Oracle Voucher and Wallet Server (formerly UBE).

XML
eXtensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.

It is called extensible because it is not a fixed format like HTML. XML is a ‘metalanguage’ — a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it’s written in SGML.

XMS
Three letter code used to designate some components and path locations used by the Oracle Communications Network Charging and Control Messaging Manager (on page 89) service and the Short Message Service. The published code is MM (on page 89) (formerly MMX).
Index

A
AAA • 89
About Installation and Removal • 87
About This Document • v
Acct-Application-Id • 38
ACS • 89
ACS Action handler • 5, 58
ACS service context • 7
acs.conf Configuration • 21, 24
Additional EDR tags • 3
Application-identifiers values • 19
Attribute Value Pairs • 2
Audience • v
Auth-Application-Id • 37, 43
AVP • 89
avpCode • 55
AVPs • 2, 50
AVPs parameters • 50

B
Background Processes • 25
Bad PIN • 16
balanceEnquiryMethod • 44
BE • 89

C
CC • 89
CCA • 89
CCR • 89
CCS • 4, 47, 49, 89
CCS eserv.config Configuration • 21, 23
CCS time reservation • 7
ccsConcept • 56
ccsConcepts • 4, 56
ccsServiceLibrary configuration • 23
cdrOperation • 56
cdrTag • 56
CEA • 89
CER • 89
Charge details • 9
checkDefinedAvpFlags • 38
Checking the Installation • 87
Checking voucher bad PIN • 17
CollectiveNotice • 40
conditionProfileBlock • 57
conditionTag • 57
conditionValue • 57
Configuration • 21
Configuration components • 21
Configuration file format • 22
Configuration Overview • 21
connectionTimeout • 74
ContextCopy parameters • 48, 69, 71
textItem • 70
conversion • 57
conversionRounding • 60
conversionScale • 60
Copyright • ii
Credit Control messages • 2
cron • 89
Custom tag names • 85

D
databaseCacheValiditySeconds • 38
DCD • 36, 74
DCD_acks.conf configuration • 24
DCD EDR tags • 46, 84
DCD EDRs • 84
DCD packages • 87
DCD parameters • 2, 36, 80, 82
DCD SLC directories and files • 87
DCD SLEE configuration • 24
DCD SMS directories • 88
DCD statistics • 83
Debug flags • 42
defaultEventFailover • 45
defaultFailureHandling • 45
defaultFixedCostDuration • 45
defaultSessionFailover • 45
Description • 16
DIA_RC (result code) • 84
DIA_REQ (current session message number) • 84
DIA_SID (session id) • 85
DIA_TIME (time ccr sent) • 85
Diagram • 4, 15, 17
Diameter • 89
Diameter Credit Control • 1
Diameter eserv.config configuration • 23
diameterBeClient • 23, 25
direct • 78, 81
Direct time charge • 9
Document Conventions • vi
Domain tab • 47, 49
Domains parameters • 48
DomainTypes • 43
DomainTypes parameters • 43
DP • 90
DPR • 90
DTMF • 90
DWR • 90
dynamicWalletReload • 48

E
Editing the file • 23
EDR • 90
EDR generation • 84
enableDraft8 • 19, 20, 38
enableScap • 38, 46
Enabling statistics • 83
eserv.config Configuration • 21, 22