

# vDicosDia Managed Object Model

## USER GUIDE

**Copyright**

© Ericsson AB 2015, 2016, 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	Term Definitions	1
1.1.1	Stack ID	1
1.1.2	Stack Instance	1
1.1.3	Stack Instance User	1
1.1.4	Diameter Application	1
1.1.5	Multiple Connections	2
<b>2</b>	<b>Diameter Application Modeling Support</b>	<b>3</b>
<b>3</b>	<b>Diameter Stack Configuration Management</b>	<b>5</b>
3.1	General	5
3.1.1	Common Attributes to All MOs	5
3.1.2	DIA-CFG-Application MO	5
3.1.3	DIA-CFG-StackContainer MO	5
3.2	Configuring the Diameter Stack	5
3.2.1	Initial Configuration	6
3.2.2	Delete Configuration	6
3.3	Configuring the Own Node	6
3.3.1	Managed Objects	6
3.3.1.1	DIA-CFG-OwnNodeConfig MO	6
3.3.2	Configuration	7
3.4	Configuring a Peer Node and Connections	7
3.4.1	Managed Objects	7
3.4.1.1	DIA-CFG-PeerNodeContainer	7
3.4.1.2	DIA-CFG-NeighbourNode MO	7
3.4.1.3	DIA-CFG-Conn MO	8
3.4.2	Configuration	8
3.5	Configuring the Realm Routing Table (RRT)	8
3.5.1	Managed Objects	8
3.5.1.1	DIA-CFG-RoutingContainer	8
3.5.1.2	DIA-CFG-Drt MO	8
3.5.1.3	DIA-CFG-AuthReqContainer MO	9
3.5.1.4	DIA-CFG-AccReqContainer MO	9
3.5.1.5	DIA-CFG-AppRouting MO	9
3.5.2	Configuration	9
3.6	Configuring the Dictionary	9
3.6.1	Managed Objects	9
3.6.1.1	DIA-CFG-DictionaryContainer	10
3.6.1.2	DIA-CFG-Vendor	10
3.6.1.3	DIA-CFG-AVPDef	10
3.6.2	Configuration	10



<b>4</b>	<b>COM Operations with Diameter MOs</b>	<b>11</b>
4.1	COM NETCONF Operations with Diameter MOs	11
4.1.1	NETCONF Example of "create" Operation	11
4.1.2	NETCONF Example of "get" Operation	12
4.1.3	NETCONF Example of "replace" Operation	12
4.1.4	NETCONF Example of "delete" Operation	13
4.2	COM CLI Operations with Diameter MOs	14
4.2.1	CLI Example of "create" Operation	14
4.2.2	CLI Example of "get" Operation	15
4.2.3	CLI Example of "replace" Operation	15
4.2.4	CLI Example of "delete" Operation	16
4.3	COM Schematron Validation Service Support	16
<b>5</b>	<b>Diameter over SS7 CAF Configuration Guidelines</b>	<b>19</b>
5.1	Configuring SS7 CAF CP Manager Address for Non-Scaling Case	19
5.2	Configuring SS7 CAF CP Manager Address for Scaling Case	19
5.3	Configuring SS7 CAF CP	20
5.4	Configuring SS7 CAF SCTP	21
<b>6</b>	<b>Logging</b>	<b>23</b>
	<b>Glossary</b>	<b>25</b>
	<b>Reference List</b>	<b>27</b>



# 1 Overview

This document provides an overview of configuration management possibilities for the Diameter stack and preparations for installing the Diameter Application.

Information about the most important Managed Objects (MOs) is also included in this document.

**Note:** When configuration data is changed, a latency period of up to 10 seconds exists before the change has an impact on the traffic.

Long commands, code examples, and paths are broken into separate lines to make reading easier. Line breaks are indicated with “⇒” (arrow) symbol. This is not part of the command, example, or path — do not copy multiple lines at once, and do not put space at the end of the lines when copying, just continue with the next line.

## 1.1 Term Definitions

This section describes the key concepts of the Diameter Stack.

### 1.1.1 Stack ID

Stack ID is an identifier of an instance of the Diameter base configuration. A group of applications can use the same Stack ID if they want to share the configuration. A Stack ID has a one-to-one relation with Own Nodes. The Stack ID is defined by applications and is configured in the system during installation.

### 1.1.2 Stack Instance

Stack Instance is a term used in this document and is a synonym of Stack ID.

### 1.1.3 Stack Instance User

Stack Instance User is a concept used throughout the configuration documents to identify an application or a group of applications using the same Stack Instance, as defined by a Stack ID.

### 1.1.4 Diameter Application

Diameter Application is an application that extends the base protocol provided by vDicos or CBA. A Diameter Application is associated with only one Stack Instance. Several applications can share the same Stack Instance.



### **1.1.5 Multiple Connections**

Setting up either a single connection or multiple connections is possible between two Diameter peers. By the use of multiple connections, the processor load is spread, (a new process is created for each established connection and is distributed by a round robin algorithm) and the throughput is increased.



## 2 Diameter Application Modeling Support

It is possible to mount the Diameter sub-tree to any place in the ECIM tree. The following model files are provided in order to do that:

- `dia_mim_mp.xml` – Includes Diameter COM model.
- `dia_mim.xml`, `dia_mim.schema`, `dia_mim.modelinfo` – Include Diameter LDAP model.
- `dia_mim.emx` – Includes Diameter EMX model.

The listed files can be found in the following directories:

- `'cdf-query getpath SWI/VDDiameterMimFiles'/spec`
- `'cdf-query getpath SWI/VDDiameterMimFiles'/incl`







## 3 Diameter Stack Configuration Management

This section describes the configuration management in the Diameter stack.

**Note:** Install Diameter Application together with vDicos before any configuration activities.

### 3.1 General

This section provides general information on MOs used in the Diameter stack.

#### 3.1.1 Common Attributes to All MOs

All MOs inherit from JIM-ManagedObject for backward compatibility.

None of the inherited attributes, such as `objectClass`, `ownerId`, `groupId`, `shareTree`, and `permissions`, must be modified when configuring the Diameter stack.

#### 3.1.2 DIA-CFG-Application MO

The DIA-CFG-Application MO identifies the vDicos Diameter stack. It constitutes a container to all MOs defined for the vDicos Diameter stack and is derived from JIM-Application. The only purpose of this MO is to serve as a container for the objects belonging to the relevant application.

Only one instance of this MO exists, which is created during installation by the vDicos Diameter stack.

No management activities are to be performed in this MO.

#### 3.1.3 DIA-CFG-StackContainer MO

The DIA-CFG-StackContainer MO is a container for Node and Routing data related to one Stack Instance and is automatically created during Diameter installation.

### 3.2 Configuring the Diameter Stack

This section describes configuration activities in the Diameter stack.



### 3.2.1 Initial Configuration

When the Diameter stack is successfully installed, the following main steps need to be performed to configure it:

- Modify the attributes of the Own Node.
- Add a Peer Node.
- Modify the attributes of the Peer Node
- Add a Connection (if multiple connections are used).
- Add a Realm Routing Table (RRT).
- Modify the attributes of the RRT

### 3.2.2 Delete Configuration

When the Diameter stack is not used, the following main steps need to be performed to delete its configuration:

- Delete the RRT.
- Delete the connection(s).
- Delete the Peer Node(s).

## 3.3 Configuring the Own Node

### 3.3.1 Managed Objects

#### 3.3.1.1 DIA-CFG-OwnNodeConfig MO

The DIA-CFG-OwnNodeConfig MO is used to define the configuration of the Own Diameter Node.

Each instance of this MO defines a particular Diameter stack Instance User of the vDicos Diameter stack. Only one instance of this MO is available per Diameter stack Instance User. Each instance uses the Diameter stack identity (`stackId`) of its corresponding Diameter application as the key.

Once the application has been successfully installed, the DIA-CFG-OwnNodeConfig MO, which is created for the application, has to be configured.

The deletion of a particular DIA-CFG-OwnNodeConfig MO automatically deletes all objects related to that Diameter stack Instance. All Peer Nodes and the RRT associated with the relevant Stack Instance are deleted. The corresponding Stack ID is deleted from the `stackId` lists of both the DIA-CFG-Vendor and the



DIA-CFG-AVPDef MOs if this Stack ID exists in these lists. These objects are not deleted themselves because they can be in use by other Diameter applications.

A new Own Node can only be created through reinstallation of the application.

### 3.3.2 Configuration

For the steps of configuring Own Node, see [vDicosDia, Configure Own Node](#).

## 3.4 Configuring a Peer Node and Connections

### 3.4.1 Managed Objects

This section describes the MOs of the Peer Nodes.

#### 3.4.1.1 DIA-CFG-PeerNodeContainer

The DIA-CFG-PeerNodeContainer MO is a container for Peer Nodes related to one Stack Instance. This MO is automatically created during Diameter Installation.

#### 3.4.1.2 DIA-CFG-NeighbourNode MO

The DIA-CFG-NeighbourNode MO is used to define Diameter Nodes to which the Own Node has a direct transport connection or connections.

Peer Nodes are objects used to specify Peer Nodes known to the applications using the Diameter stack. The list of all Peer Nodes is known as Peer Table.

Every stack instance user, identified by a `stackId`, has its own set of DIA-CFG-NeighbourNode MOs. Such objects have to be created for each Peer Node in the network for each Diameter Application. For example, a set of DIA-CFG-NeighbourNode MOs is available for the Home Subscriber Server (HSS) application and another one for the Call Session Control Function (CSCF) application if these applications are installed and defined as different Stack Instance Users.

Prior to the creation of a new DIA-CFG-NeighbourNode MO, the `stackId` must be defined during installation. Otherwise, an error occurs and the create operation does not take place.

All hosts in the RRT must be defined as Peer Nodes in the MO. A DIA-CFG-NeighbourNode MO to be deleted must be previously deleted from the RRT for the same Stack Instance. This means that the reference to this DIA-CFG-NeighbourNode MO is no longer kept by the DIA-CFG-AppRouting MO and that the RRT no longer offers routing toward the Peer Node.



### 3.4.1.3 DIA-CFG-Conn MO

The DIA-CFG-Conn MO is used to define a specific connection for a Diameter node. The first connection object is created by default when the Peer Node object is created. For each new desired connection, a new connection object must be created.

**Note:** Adding a second connection implicitly changes the traffic handling policy of the Diameter stack from single restricted connection to multiple connections, hence some temporary traffic disturbances can be expected while the existing single connection is dropped and two new connections are built. Adding further connections after the second does not involve change in the traffic handling policy and has no traffic impact.

Every Peer Node, identified by `nodeId`, has its own set of DIA-CFG-Conn MOs.

**Note:** The sum of all DIA-CFG-NeighbourNode and DIA-CFG-Conn MOs on the node for all stacks and peers has to be lower than 9500. If this limit is exceeded, the create operation is rejected with a `JIM_RESULT_OPERATIONS_ERROR` error.

### 3.4.2 Configuration

For the steps of configuring Peer Node and Connections, see [vDicosDia, Configure Peer Node and Peer Connection](#) and [vDicosDia, Disable or Enable Peer Node](#).

## 3.5 Configuring the Realm Routing Table (RRT)

This section provides information on the configuration of RRT.

### 3.5.1 Managed Objects

This section describes the MOs of an RRT.

#### 3.5.1.1 DIA-CFG-RoutingContainer

The DIA-CFG-RoutingContainer MO is a container for RRTs related to one Stack Instance. This MO is automatically created during Diameter installation.

#### 3.5.1.2 DIA-CFG-Drt MO

The DIA-CFG-Drt MO represents an entry in the RRT and is used to find the routing required by the Diameter base protocol, which is based on the destination realm. Every Diameter Application has its own RRT built up of DIA-CFG-Drt instances.

Prior to creation of a new DIA-CFG-Drt object, the `stackId` must be defined during installation, otherwise an error occurs and the DIA-CFG-Drt object create operation does not take place.



If an entry is configured with "default" string for the `realm` field, it is treated as Default RRT entry and it is used for the routing of messages with unknown realm. entry can be created both for incoming and outgoing requests.

Empty string can not be used for the `realm` field during the creation of a `DIA-CFG-Drt`.

### 3.5.1.3 **DIA-CFG-AuthReqContainer MO**

The `DIA-CFG-AuthReqContainer` MO is a container that has a child relationship with `DIA-CFG-Drt` objects. It keeps routing data associated with request messages defined by the Diameter Authentication application protocols. This MO is automatically created when an RRT is created.

### 3.5.1.4 **DIA-CFG-AccReqContainer MO**

The `DIA-CFG-AccReqContainer` MO is a container that has a child relationship with `DIA-CFG-Drt` objects. It keeps routing data associated with request messages defined by the Diameter Accounting application protocols. This MO is automatically created when an RRT is created.

### 3.5.1.5 **DIA-CFG-AppRouting MO**

The `DIA-CFG-AppRouting` MO contains a Route Action and possibly a list of Peer Node IDs. The Route Action is used for incoming Diameter request messages while the Peer Node IDs are used when Diameter request messages are addressed to other Diameter Nodes.

## 3.5.2 **Configuration**

For the steps of configuring the Realm Routing Table (RRT), see `vDicosDia`, [Configure Routing Table](#).

## 3.6 **Configuring the Dictionary**

Configuring vendors and Attribute Value Pairs (AVPs) are optional steps depending on the application. All vendors and AVPs can be automatically created by the application installation and are normally not changed afterwards.

### 3.6.1 **Managed Objects**

This section describes the MOs of vendors and AVPs.



### 3.6.1.1 DIA-CFG-DictionaryContainer

The DIA-CFG-DictionaryContainer MO is a container for vendors and AVPs related to a Dictionary. This MO is automatically created during Diameter installation.

### 3.6.1.2 DIA-CFG-Vendor

The DIA-CFG-Vendor MO contains attributes to define a Diameter Vendor. The vendor is a container that has a list with references of all its AVPs.

This object can be used by different Stack Instances, in the sense that they use a certain vendor. That means that they are aware of the existence of that vendor.

Prior to creation or updating of a DIA-CFG-Vendor object, all application `stackIds` that are to be defined for the vendor must be defined during installation, otherwise an error occurs and the create or update operation does not take place.

When a DIA-CFG-Vendor object is deleted, all its AVPs are automatically deleted. Prior to deletion, make sure that no AVPs of any application are used.

### 3.6.1.3 DIA-CFG-AVPDef

The DIA-CFG-AVPDef MO stores information about an AVP definition and is used to extract information from incoming Diameter messages and to build new Diameter messages.

An AVP code can be reused by different vendors, this is why the DIA-CFG-AVPDef object key is made of the `AVPCode` and the `vendorId`. This object can be used by the different Stack Instances, in the sense that they use a certain AVP.

Prior to creation or updating of a DIA-CFG-AVPDef MO, all `stackIds` that make use of that AVP specified by the `stackId` attribute, must be defined during installation, otherwise an error occurs and the create or update operation does not take place.

Similarly, an AVP grouped within another AVP of type “Grouped” cannot be deleted as long as the latter has references to those DIA-CFG-AVPDef objects that are grouped within it. Any delete request addressed to a grouped AVP object results in an error and the operation does not take place. To delete an AVP grouped within another one, first dissociate it from the list of grouped AVPs before deletion. Alternatively, if the grouped AVP is deleted, it is possible to delete those AVPs previously grouped within it.

## 3.6.2 Configuration

For steps on configuring Vendor and Attribute-Value Pairs, see [vDicosDia, Expand Diameter Dictionaries](#).



## 4 COM Operations with Diameter MOs

COM North Bound Interfaces (NBIs) NETCONF and CLI are the only means to configure Diameter stack in vDicos and CBA.

Both NETCONF and CLI utilize a Secure Shell (SSH) connection to the Operation and Maintenance (O&M) Virtual IP (VIP) interface of the vDicos node.

**Note:** In the examples below Diameter sub-tree is mounted to ManagedElement.

### 4.1 COM NETCONF Operations with Diameter MOs

To use NETCONF an operator has to pass an XML file to the shell:

```
ssh -p 830 <user_name>@<node_vip_address> -s -t =>
netconf < <xml_file>
```

where <xml\_file> refers to the XML file name.

#### 4.1.1 NETCONF Example of "create" Operation

Use this XML to create DIA-CFG-PeerNodeContainer:

**Note:** In the example below the parent MOs are assumed to have been already created.

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <DIA-CFG-Application xmlns="urn:com:ericsson:ecim:dia_mim">
          <applicationName>DIA</applicationName>
          <DIA-CFG-StackContainer>
            <stackContainerId>TST_CLI6</stackContainerId>
            <DIA-CFG-PeerNodeContainer operation="create">
              <peerNodeContainerName>peerNodeContainerName</peerNodeContainerName>
              <peerNodeContainerId>TST_CLI6</peerNodeContainerId>
            </DIA-CFG-PeerNodeContainer>
          </DIA-CFG-StackContainer>
        </DIA-CFG-Application>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

Example 1 Example XML for "create"



## 4.1.2 NETCONF Example of "get" Operation

Use this XML to get DIA-CFG-Vendor:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <ManagedElement>
        <managedElementId>1</managedElementId>
        <DIA-CFG-Application xmlns="urn:com:ericsson:ecim:dia_mim">
          <applicationName>DIA</applicationName>
          <DIA-CFG-DictionaryContainer>
            <dictionaryContainerName>dictionaryContainerName</dictionaryContainerName>
            <DIA-CFG-Vendor>
              <diaVendorId>2</diaVendorId>
            </DIA-CFG-Vendor>
          </DIA-CFG-DictionaryContainer>
        </DIA-CFG-Application>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

Example 2 Example XML for "get"

## 4.1.3 NETCONF Example of "replace" Operation

Use this XML to replace DIA-CFG-OwnNodeConfig:





```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <DIA-CFG-Application xmlns="urn:com:ericsson:ecim:dia_mim">
          <applicationName>DIA</applicationName>
          <DIA-CFG-StackContainer>
            <stackContainerId>TST_CLI4</stackContainerId>
            <DIA-CFG-OwnNodeConfig operation="replace">
              <stackId>TST_CLI4</stackId>
              <hostId>hss1.ericsson.se</hostId>
              <allowConnectFromUnknownNode>true</allowConnectFromUnknownNode>
              <watchdogTimeIdle>25</watchdogTimeIdle>
              <maxNumberOfRetries>8</maxNumberOfRetries>
              <maxRequestPendingTime>5</maxRequestPendingTime>
              <tcTimer>45</tcTimer>
              <realm>ericsson.se</realm>
              <ipAddressesList>0:10.1.137.2</ipAddressesList>
              <sctpAddressesList>0:10.1.137.4</sctpAddressesList>
              <transportLayerType>3</transportLayerType>
              <maxOutboundSctpStreams>3</maxOutboundSctpStreams>
              <maxInboundSctpStreams>5</maxInboundSctpStreams>
              <diaVendorId>7</diaVendorId>
              <productName>Ericsson Diameter Stack</productName>
              <firmwareRevision>3</firmwareRevision>
              <supportedAuthAppIds>0</supportedAuthAppIds>
              <portNr>40000</portNr>
              <enabled>false</enabled>
              <loadRegulationEnabled>true</loadRegulationEnabled>
              <sendErrorAtOverload>true</sendErrorAtOverload>
              <traceSctpHandler>TRUE</traceSctpHandler>
              <sctpHandlerLogLevel>7</sctpHandlerLogLevel>
            </DIA-CFG-OwnNodeConfig>
          </DIA-CFG-StackContainer>
        </DIA-CFG-Application>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

Example 3 Example XML for "replace"

#### 4.1.4

#### NETCONF Example of "delete" Operation

Use this XML to delete DIA-CFG-AvpDef:



```

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <DIA-CFG-Application xmlns="urn:com:ericsson:ecim:dia_mim">
          <applicationName>DIA</applicationName>
          <DIA-CFG-DictionaryContainer>
            <dictionaryContainerName>dictionaryContainerName</dictionaryContainerName>
            <DIA-CFG-Vendor>
              <diaVendorId>2</diaVendorId>
              <DIA-CFG-AvpDef operation="delete">
                <avpId>2:1003</avpId>
              </DIA-CFG-AvpDef>
            </DIA-CFG-Vendor>
          </DIA-CFG-DictionaryContainer>
        </DIA-CFG-Application>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>

```

Example 4 Example XML for "delete"

## 4.2 COM CLI Operations with Diameter MOs

To use CLI an operator has to pass commands to the shell:

```
ssh -p 830 <user_name>@<node_vip_address> -s -t cli
```

### 4.2.1 CLI Example of "create" Operation

Use these commands to create DIA-CFG-PeerNodeContainer:

**Note:** In the example below the parent MOs are assumed to have been already created.

**configure**

```
show all verbose ManagedElement=1,⇒
DIA-CFG-Application=DIA,⇒
DIA-CFG-StackContainer=TST_CLI6
```

```
ManagedElement=1,DIA-CFG-Application=DIA,⇒
DIA-CFG-StackContainer=TST_CLI6,⇒
DIA-CFG-PeerNodeContainer=TST_CLI6
```

**commit**



### 4.2.2 CLI Example of "get" Operation

Use this command to display DIA-CFG-Vendor:

```
show all verbose ManagedElement=1,⇒  
DIA-CFG-Application=DIA,⇒  
DIA-CFG-DictionaryContainer=dictionaryContainerName,⇒  
DIA-CFG-Vendor=0
```

### 4.2.3 CLI Example of "replace" Operation

Use these commands to modify DIA-CFG-OwnNodeConfig:

**configure**

```
ManagedElement=1,DIA-CFG-Application=DIA,⇒  
DIA-CFG-StackContainer=TST_CLI1,⇒  
DIA-CFG-OwnNodeConfig=TST_CLI1
```

hostId=hss1.ericsson.se

allowConnectFromUnknownNode=true

watchdogTimeIdle=25

maxNumberOfRetries=8

maxRequestPendingTime=5

tcTimer=45

realm=ericsson.se

ipAddressesList="0:10.1.137.2"

ipAddressesList="1:10.1.137.3"

sctpAddressesList="0:10.1.137.4"

sctpAddressesList="1:10.1.137.5"

transportLayerType=3

maxOutboundSctpStreams=3

maxInboundSctpStreams=5

productName="Ericsson Diameter Stack"

firmwareRevision=3

supportedAuthAppIds=0



```
portNr=43000
enabled=true
loadRegulationEnabled=true
sendErrorAtOverload=true
traceSctpHandler=TRUE
sctpHandlerLogLevel=7
commit
```

#### 4.2.4 CLI Example of "delete" Operation

Use these commands to delete DIA-CFG-AvpDef:

```
configure

no ManagedElement=1,DIA-CFG-Application=DIA,⇒
DIA-CFG-DictionaryContainer=dictionaryContainerName,⇒
DIA-CFG-Vendor=0,⇒
DIA-CFG-AvpDef=0:20016

commit
```

### 4.3 COM Schematron Validation Service Support

Diameter provides the support for COM Schematron Validation Service when configuring via COM NETCONF or COM CLI. The support is implemented as a set of Schematron-based constraints applicable to the configuration model. The constraints are inserted in MIM MP XML file and used by COM for validation before the transaction is committed. For more details, refer to MAF R2A74 Application Developer's Guide, Reference [2].



```
configure
ManagedElement=1,DIA-CFG-Application=DIA,⇒
DIA-CFG-StackContainer=TST_CLI1,⇒
DIA-CFG-NeighbourNode=server1.com\23TST_CLI1
initiateConnection=true
validate
Transaction validation failed!
Layer must be defined if peer node is enabled and act as Initiator.
transportLayerType=2
validate
Transaction validation failed!
If SCTP, then NeighbourNode.sctpAddressesList attribute must be set.
sctpAddressesList="0:2dea::10:a:2"
validate
Transaction is valid!
commit
```

Example 5 Schematron Validation





## 5 Diameter over SS7 CAF Configuration Guidelines

This section describes the configuration guidelines for Diameter over SS7 CAF.

### 5.1 Configuring SS7 CAF CP Manager Address for Non-Scaling Case

The environment variable `Ss7CpManagerAddr` must include the addresses (including port numbers) of all nodes where SS7 CAF is installed. That is needed to allow Diameter processes (acting as CP dynamic users) to register in SS7 CAF CP Manager that can be running on any of SS7 CAF nodes.

```
<SoftwareConfigData domain="Lpmsv">
...
  <parameter name="Ss7CpManagerAddr">
    <value>PL-3:6669,PL-4:6669, PL-5:6669,PL-6:6669</value>
  </parameter>
...
</SoftwareConfigData>
```

Example 6 SiteSpec.xml with 4 SS7 CAF Payload Nodes in Non-Scaling Case  
(Depends on the Number of Payload Nodes)

### 5.2 Configuring SS7 CAF CP Manager Address for Scaling Case

To participate in scale-out and scale-in operations when SCTP is running, use LDE MIP as Common Parts manager address in the `Ss7CpManagerAddr` environment variable.

```
# Define moveable IP addresses
...
mip payload ss7cafcpmaddress eth0:3 internal 169.254.100.203
...
```

Example 7 Configure Address in cluster.conf:

```
...
MSGIPA=CP_MANAGER, ss7cafcpmaddress:6669
...
```

Example 8 State in cp.cnf:



```

...
<PROPERTY NAME="CPManagerAddress" TYPE="string">
  <VALUE>ss7cafcpmaddress:6669</VALUE>
</PROPERTY>
...

```

Example 9 State in active\_om.cim:

```

<SoftwareConfigData domain="Lpmsv">
...
  <parameter name="Ss7CpManagerAddr">
    <value>ss7cafcpmaddress:6669</value>
  </parameter>
...
</SoftwareConfigData>

```

Example 10 SiteSpec.xml with Scaling, where ss7cafpmaddress is an LDEwS MIP (Independent from the Number of Payload Nodes)

**Note:** If LDE MIP is not set or configured, there is a high risk that Diameter links over SCTP will not be in UP state after scale-in, scale-out, node reboot, lock-unlock PL and so on.

## 5.3 Configuring SS7 CAF CP

User IDs used by Diameter processes must be reserved by defining SS7 CAF CP parameter DYNAMICIDS. That helps to avoid any user ID collision during dynamic registration.

```

...
DYNAMICIDS=USER15-USER20
...

```

Example 11 cp.cnf

If MSGBUFSIZE or MSGMAXENTRIES are changed, take the the following limitation for SS7CAF into account:

$$(\text{number of stackIds}) \times \text{MSGBUFSIZE} \times \text{MSGMAXENTRIES} < 250000000$$

```

...
MSGBUFSIZE=5000
MSGMAXENTRIES=3000
...

```

Example 12 cp.cnf





## 5.4 Configuring SS7 CAF SCTP

The SS7 CAF SCTP configuration options `PortRangeFrom` and `PortRangeTo` must be set to 0. That is needed to allow SCTP to assign free port numbers during SCTP end point initialization.





## 6 Logging

All vDicos logs, including the ones related to the Diameter stack, are stored by Core MW LOG service under the following directory:

```
/opt/cdclsv/storage/log/lpmsv
```

Application logs are located in the following directory:

```
/cluster/storage/no-backup/coremw/var/log/⇒  
saflog/vdicos/applog
```





## Glossary

For the glossary of this document, refer to RFC 6733—Diameter Base Protocol, Reference [3].





## Reference List

### Other Documents

- [1] Diameter Managed Object Model, 19089-LZN 708 0652
- [2] MAF R2A74 Application Developer's Guide, 5/198 17-CAA 901 2587/5

### Standards

- [3] RFC 6733– Diameter Base Protocol, <http://datatracker.ietf.org/doc/rfc6733/>