User Configuration

- Basic Configuration, on page 1
- Routing Techniques, on page 6
- Advanced Features, on page 11

Basic Configuration

Before you begin using CPS vDRA, perform the following basic configurations in CPS DRA:

- Configure Systems
- Configure Diameter Application
- Configure Routing AVP Definitions

Configure Systems

In CPS DRA, navigate to the **System and Plugin Configuration**.

Configure the stack in **DRA Configuration** plugin.

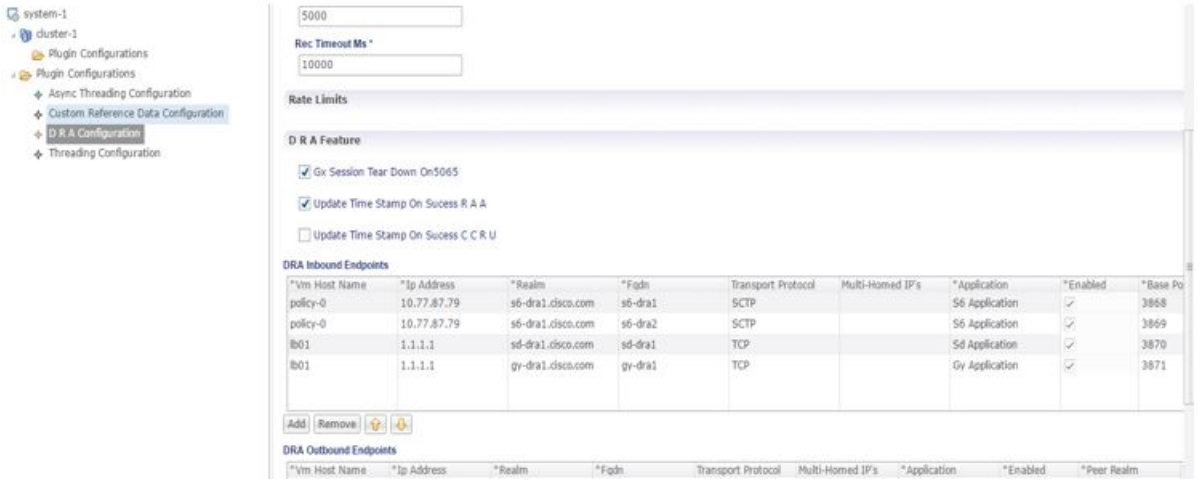
Configure the **DRA Inbound Endpoints** for incoming peer connections and **DRA Outbound Endpoints** for outgoing peer connection.

You can choose the Transport Protocol as TCP and SCTP depending on your requirement.

You can also specify the IPv4 or IPv6 address configuration for the stack connection.

The following image shows a sample configuration.

Figure 1: Sample Systems Configuration



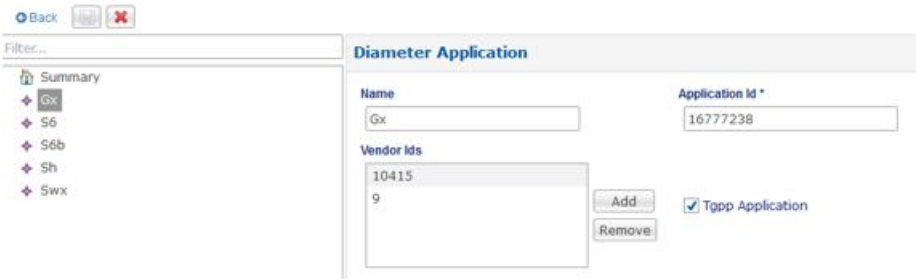
For more information, see DRA Configuration.

Configure Diameter Application

Configure the Diameter applications that are required to be connected over various interfaces with CPS vDRA.

The following image is a sample of a Gx application configuration:

Figure 2: Sample Diameter Application Configuration



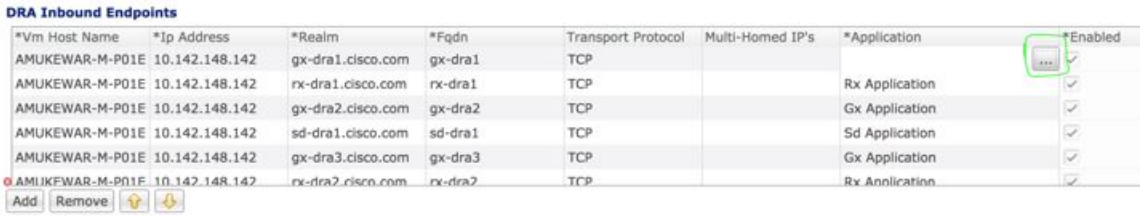
For more information, see Diameter Application.

Configure Multiple Diameter Applications for a Peer Connection

Previously, vDRA supported a single application on a peer connection. In this release, vDRA supports multiple applications on a peer connection.

To configure multiple applications for a peer connection, go to vDRA Inbound Endpoints in DRA Plugin configuration. In the Applications field, select the button as shown:

Figure 3: DRA Inbound Endpoints



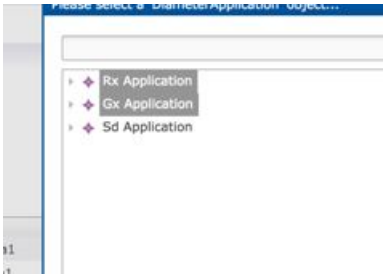
DRA Inbound Endpoints

*Vm Host Name	*Ip Address	*Realm	*Fqdn	Transport Protocol	Multi-Homed IP's	*Application	*Enabled
AMUKEWAR-M-P01E	10.142.148.142	gx-dra1.cisco.com	gx-dra1	TCP			
AMUKEWAR-M-P01E	10.142.148.142	rx-dra1.cisco.com	rx-dra1	TCP		Rx Application	
AMUKEWAR-M-P01E	10.142.148.142	gx-dra2.cisco.com	gx-dra2	TCP		Gx Application	
AMUKEWAR-M-P01E	10.142.148.142	sd-dra1.cisco.com	sd-dra1	TCP		Sd Application	
AMUKEWAR-M-P01E	10.142.148.142	gx-dra3.cisco.com	gx-dra3	TCP		Gx Application	
AMUKEWAR-M-P01E	10.142.148.142	rx-dra2.cisco.com	rx-dra2	TCP		Rx Application	

Add Remove

Select all the applications you require.

Figure 4: Application Selection



The following example shows multiple Diameter applications for a peer connection:

Figure 5: Multiple Applications

DRA Inbound Endpoints

*Vm Host Name	*Ip Address	*Realm	*Fqdn	Transport Protocol	Multi-Homed IP's	*Application
AMUKEWAR-M-P01E	10.142.148.142	gx-dra1.cisco.com	gx-dra1	TCP		Rx Application, Gx Application
AMUKEWAR-M-P01E	10.142.148.142	rx-dra1.cisco.com	rx-dra1	TCP		Rx Application
AMUKEWAR-M-P01E	10.142.148.142	gx-dra2.cisco.com	gx-dra2	TCP		Gx Application
AMUKEWAR-M-P01E	10.142.148.142	sd-dra1.cisco.com	sd-dra1	TCP		Sd Application

Configure Routing AVP Definitions

Configure the Routing AVP definitions to route calls on the basis of the AVPs that are present in diameter message.

In the **Routing AVP Definition** page, you specify the Application name and the table for table-driven routing.

In the **Diameter Application** page, configure the Application Route for table-driven routing.

The following screenshots show a sample configuration:

Figure 6: Routing Avp Definition

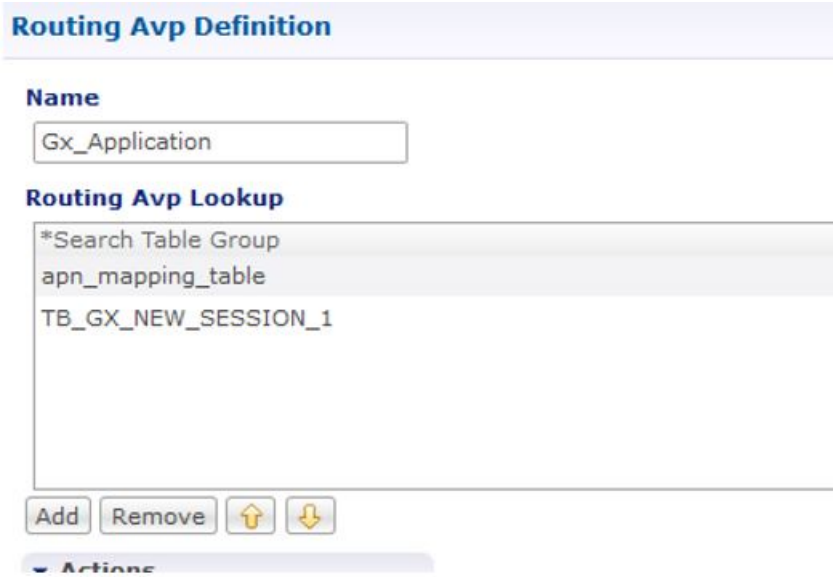


Figure 7: Diameter Application

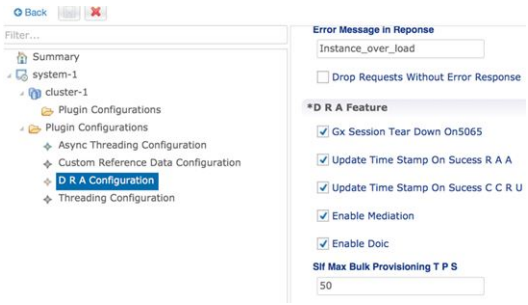
Enable Mediation

By default, mediation feature is disabled.

To enable the mediation, log in to Policy builder, select the “Enable Mediation” checkbox in **Systems** > **DRA Configuration** > **DRA Feature**.

Finally, publish the configuration changes.

Figure 8: Enable Mediation



Enable DOIC

By default, DOIC feature is disabled.

To enable the DOIC, log in to Policy Builder, select the “Enable DOIC” checkbox in **Systems** > **DRA Configuration** > **DRA Feature**.

You must also enable DOIC for the group in Peer Group SRK Mapping table as described in Peer Group SRK Mapping.

Configure throttling using DOIC. For more information, see Configure Throttling of Diameter Messages Using DOIC, on page 22.

Publish the configuration changes.

Configure Interfaces for SCTP Multi-homing

As a pre-requisite for SCTP multi-homing, you must first move the physical interfaces of Director VM inside the diameter-endpoint container.

To move the IPv4 interfaces, perform the following commands in vDRA Master CLI mode:

```
config diameter host dra-director-0-dra-director-52wumnq5l2yl
 interface ens3 ipv4 address 10.77.87.79 broadcast 10.77.87.255 prefix-length 24
diameter host dra-director-0-dra-director-52wumnq5l2yl interface ens3
route default gateway 10.77.87.1
diameter host dra-director-0-dra-director-52wumnq5l2yl interface ens5
 ipv4 address 10.225.115.199 broadcast 10.225.115.255 prefix-length 24
diameter host dra-director-0-dra-director-52wumnq5l2yl interface ens5
 route default gateway 10.225.115.1
```

To move the IPv6 interfaces, perform the following commands in vDRA Master CLI mode:

```
config diameter host dra-director-0-dra-director-52wumnq5l2yl
interface ens6 ipv6 address 2003:3051::114 prefix-length 64
diameter host dra-director-0-dra-director-52wumnq5l2yl interface
ens6 route default gateway 2003:3051::1
diameter host dra-director-0-dra-director-52wumnq5l2yl interface
ens7 ipv6 address 2003:3052::114 prefix-length 64
diameter host dra-director-0-dra-director-52wumnq5l2yl interface
ens7 route default gateway 2003:3052::1
```

Commit the configuration.

Routing Techniques

You can define the routing of calls based on destination host, SRK, or a table.

Configure Destination Host Routing

Destination host based routing is the basic and default routing technique used by CPS vDRA.

When the incoming diameter request contains the destination-host AVP that has the direct connection with the CPS vDRA, vDRA routes the message directly to that connected host.

Before you begin

Stack must be up and running.

For more information, see Basic Configuration, on page 1.

After configuring the stacks, Diameter endpoints are ready to initiate/accept Diameter connections for the defined IP address, Port, and Application-ID.

Policy DRA

For Policy DRA, you must configure the binding keys for Gx sessions.

Binding helps Policy DRA route the related Gx/Rx sessions to the same PCRF or destined PCRF.

A binding database is needed to map search keys to PCRF binding information. Each binding has a search key and binding data associated with it. The supported search keys are:

- IMSI + APN
- IPv6
- IPv4
- MSISDN + APN

Figure 9: Policy DRA Sample Configuration

binding_key_profile

Filter CRD Tables

Profile Name * (key)	IMSI APN Key Enabled	MSISDN APN Key Enabled	Framed IPv6 Enabled	Framed IPv4 Enabled	Actions
DefaultProfile	true	false	false	false	
Rx_Profile	false	false	true	false	

app_id_key_profile_mapping

Filter CRD Tables

Application Id * (key)	Profile Name	Actions
16777238	DefaultProfile	
16777236	Rx_Profile	

If the Binding Key Profile and mapping to Application ID is not configured properly, the following errors may occur:

- Gx Calls – Session binding failure in database resulting in call failures.
- Rx Calls – Binding Retrieval failure resulting in call failures.

Note IMS DRA does not require bindings. Hence, Binding Key Profile is only valid for Policy DRA.

Configure SRK Based Routing

You can configure SRK based routing in one of the following ways:

Configure Secondary Peer Fallback

Note This configuration is valid for both Policy and IMS DRA.

In SRK based routing, you can configure routing to a set of peers. This can be used for alternate routing (secondary and tertiary routes) when the Destination Host routing fails, and for binding data to select a peer for related diameter sessions. The Session Routing Table is configured within a “Peer Group SRK Mapping” Table. If a routing with dest-host fails, CPS vDRA will try to find out secondary routes on the basis of SRK.

Once the SRK of failed peer is determined by CPS vDRA, it will try to find an UP peer that is a member of:

- A peer group matching the entire SRK label. If it finds one, it will route the message to that peer.

- If it cannot find one and it is a two-label SRK, then it will try to find an UP peer in a peer group whose label2 part of its SRK matches the label2 part of the lookup SRK (where the label 1 part may be different). If it finds one, then it will route the message to that peer.

The following screenshot shows an example of SRK configuration:

Figure 10: SRK Configuration

peer_group_srk_mapping

Filter CRD Tables

Peer Group * (key)	Session Routing Key	Destination Host Routing Rule *	Destination Host Replace	Destination Realm Replace	Actions
GX_DC_1		preferred			
SD_DC_2		preferred			
GX_DC_2		preferred			
GX_DC_3	clusterb.dc1	preferred			
SD_DC_3	clusterb.dc2	preferred			
RX_DC_2	clusterb.dc1	preferred			

Configure Binding Retriever for Rx Calls

Note This configuration is valid for Policy only.

Rx (AAR) Message processing: Policy DRA receives the AAR request from Application Function (AF). AAR messages does not have destination host and the destination PCRF has to be found out by vDRA using the keys such as:

1. Framed Ipv6 Address
2. Framed IP address
3. IMSI APN key
4. MSISDN APN key

Binding is created by vDRA when Gx-CCRI is received at DRA. DRA creates the bindings on the basis of CRD configurations and the availability of AVPs in Gx message. If the configured keys are present in Gx message, vDRA creates and stores the binding in Binding Database. On receiving AAR request, vDRA searches for the session stored in bindings on the basis of Rx profile, and will determine the SRK of Gx-PCRF peer. DRA will then forward the Rx request to the Rx peer having the same SRK. [SRK will be configured as mentioned in following section].

Configure SLF Based Routing

Note This configuration is valid for IMS DRA only.

SLF Routing works with two major configurations and tables within CRD:

1. SLF Trigger Profile: For the incoming Diameter requests where Destination-Host is not present (or Destination-Host is present with same of DRA-Host Name) this SLF Trigger table is triggered. In this Table, there are three inputs that you need to configure:
   - Application-Id: Diameter Application ID for which the SLF Query is to happen.
   - Command-Code: Diameter Command Code for which the SLF query is to happen.

Note If this field is configured with a '*', it indicates that SLF query is expected to happen for all the command codes for the specific application.

   - Destination-Realm: Destination-Realm of the Diameter Endpoint (that is, HSS/AAA) or the Destination Realm of vDRA.

     Based on the Input keys (Application-Id, Command-Code and Destination-Realm) configured, if all the entries (as mentioned above) matches with the incoming message then vDRA(SLF) picks the "SLF Lookup Type" and "SLF-Destination-Type" as configured in the SLF Trigger Table.
   - SLF LookupType- Currently the SLF LookupType can be configured as IMSI or MSISDN. Based on the configured value of IMSI or, MSISDN, vDRA (SLF) further makes a query towards the SLF-DB.
   - SLF Destination-Type-Based on the configured value in the 'SLF Destination Type', vDRA (SLF) makes a further query towards the SLF-DB.

Figure 11: SLF Trigger Profile

slf_trigger_profile

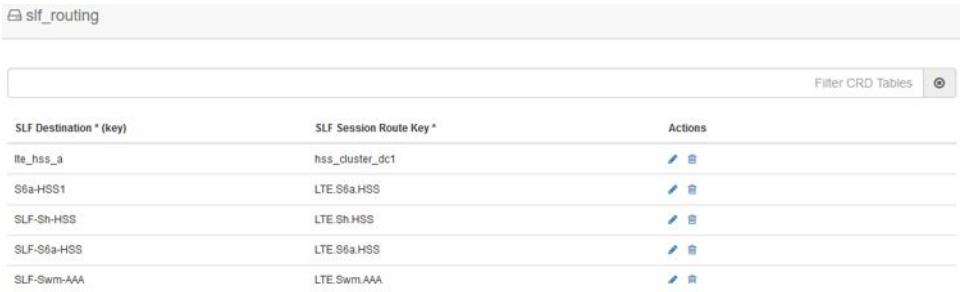
Filter CRD Tables

Application ID * (key)	Command Code * (key)	Destination Realm * (key)	Primary Lookup Type *	Secondary Lookup Type	SLF Destination Type *	Actions
16777291	*	slh-hss.com			LTE-HSS	
16777217	*	sh-hss.com			LTE-HSS	
16777251	*	slh-hss.com			LTE-HSS	
16777217	*	sh-hs.com	MSISDN		Sh-HSS	
16777251	*	mnc286.mcc311.dranetwork.org	IMSI		S6a-HSS	

2. SLF Routing: After vDRA(SLF) makes the query in SLF-DB based on SLF-LookupType and SLF-Destination-Type, an SLF-Destination is obtained.

   SLF Mapping table consists a mapping of 'SLF-Destination' which is obtained from the SLF database and a SLF-Session-Route-Key(SLF-SRK). In the SLF Mapping Table, based on the SLF-Destination an SLF-Session-Route-Key (SLF-SRK) is derived and further Peer group is derived for routing from the Peer-Group-Peer table as the next step for Routing.

Figure 12: SLF Routing



Configure Table Driven Routing

CPS vDRA has the ability to use AVPs within the Diameter messages to help determine how to route the traffic.

The AVP being evaluated is customer configurable, through the CPS DRA GUI. The addition, subtraction, or modification of the evaluation AVP is dynamic and affected real time.

Trigger Condition for Table-driven Routing:

- After Destination-Host based routing (first priority) and SRK routing (second priority) conditions are not met, vDRA goes for Table-driven routing as third priority.
- Typically, for Table-driven Routing to trigger, a Diameter message contains a Destination-Realm AVP, but no Destination-Host AVP. So, If the Dest-Host AVP is absent, empty, or equal to the DRA FQDN, then we skip Dest-Host routing altogether and proceed directly to Table-Driven routing.

Note For IMS-DRA only, the router will try to do the SLF routing (if all conditions are met), before moving to table-driven routing.

vDRA can parse and has the ability to route based on the following AVPs:

- Destination-Host
- Destination-Realm
- Origin-Host
- Origin-Realm
- APN (from Called-Station-ID)
- IMSI (from Subscription-ID)
- MSISDN (from Subscription-ID)

Regular-expression matching and combinations of AVPs are also supported. The following configuration is required in Policy Builder:

1. Application Route: For more information, see Basic Configuration, on page 1.

2. Routing AVP definitions: For more information, see Basic Configuration, on page 1.
3. Search table group configurations: For more information, see Search Table Groups.
4. CRD configuration: For more information, see Custom Reference Data Tables.

Advanced Features

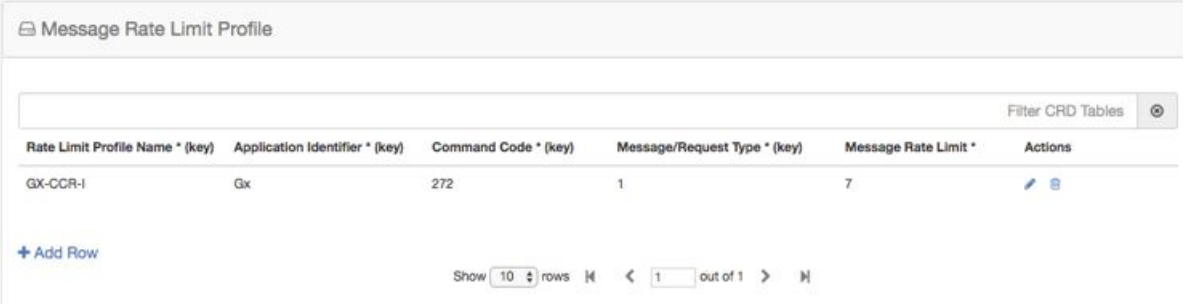
Configure Rate Limiting

You can use CPS vDRA to set rate limiting of traffic coming from and going towards a particular peer. You can configure this for both Ingress and Egress traffic. Rate limit is currently supported at peer level and message level.

1. Configure the Message Rate Limit Profile CRD table.

   Create the rate limit profile with the rate limit profile name, application ID, command code, message type, and message rate limit. For more information, see Search Table Groups – Message Rate Limit Profile table.

   Figure 13: Message Rate Limit Profile

2. Configure the Peer Rate Limiting CRD table.

   Define the peer group, peer fqdn, message direction (ingress or egress), rate limit profile (created in step 1), peer rate limit, discard behavior (whether to silently drop or send error message). For more information, see Search Table Groups – Peer Rate Limit Profile table.

   If you want the discard behavior sent in the error answer, also configure the Result Code, Error String to be sent in the answer.

Figure 14: Peer Rate Limit Profile

Peer Rate Limit Profile

Filter CRD Tables

Peer Group * (key)	Peer FQDN * (key)	Message Direction * (key)	Rate Limit Profile	Peer Rate Limit	Discard Behavior *	Result Code	Error String	Actions
match=GX_DC.*	*	Ingress	GX-CCR-I	3	Send Error Answer	3002	OVERLOAD_GX	

+ Add Row

Show 10 rows 1 out of 1

Configure Error Result Code Profile

CPS vDRA generates several internal errors that are not generic, for example, errors such as Timeout triggered, Peer not found, DB error, and so on.

You can configure error codes and error messages for internally generated errors in CPS vDRA.

To configure this error code, add entry in the Error Result Code Profile table as shown in the following image:

Figure 15: Error Result Code Profile

Error Result Code Profile

Filter CRD Tables

Application Id * (key)	Error * (key)	Result Code	Exp Result Code	Vendor Id	Err Msg	Actions
*	No Available Peer	4004	5004	10415	Peer not available	
*	No Peer Group	4005		10415	No Peer Group Available	
*	No Binding Found	3024		10415	No Binding found for request	
*	Message Loop Detected	4007		10415	Loop Detected in Message	
*	No Binding Key For Lookup		4008	10415	No Binding Key for Lookup	

+ Add Row

Show 10 rows 1 out of 1

For a particular Application ID, if DRA generates an internal error, the error code defined in this table along with the Error Message is sent back in the answer.

If the result code is not configured and the Experimental Result Code is present, then the Experimental Result-Code is sent back in the answer along with Vendor-Id AVP. Between the Result-Code and the Experimental Result code, the Result-code is of higher priority.

Configure Peer Group Answer Timeout

You can set the different request timeout durations for different application ID and peer group. In CPS vDRA, timeout is the amount of time that vDRA waits for the answer from the destination.

To configure this feature, add an entry in the Peer Group Answer Timeout table. The timeout value is in milliseconds.

The default timeout is 1.7 seconds. You can also override the timeout for specific interface in this table as shown in the following image:

Figure 16: Peer Group Answer Timeout

Peer Group Answer Timeout

Filter CRD Tables

Application Id * (key)	Peer Group * (key)	Timeout Milliseconds	Actions
16777238	GX_DC_1	25000	
16777238	GX_DC_2	22000	
16777238	GX_DC_3	20000	
16777236	RX_DC_1	20000	
16777303	SD_DC_1	25000	
16777303	SD_DC_2	25000	
16777236	RX_DC_2	22000	

+ Add Row

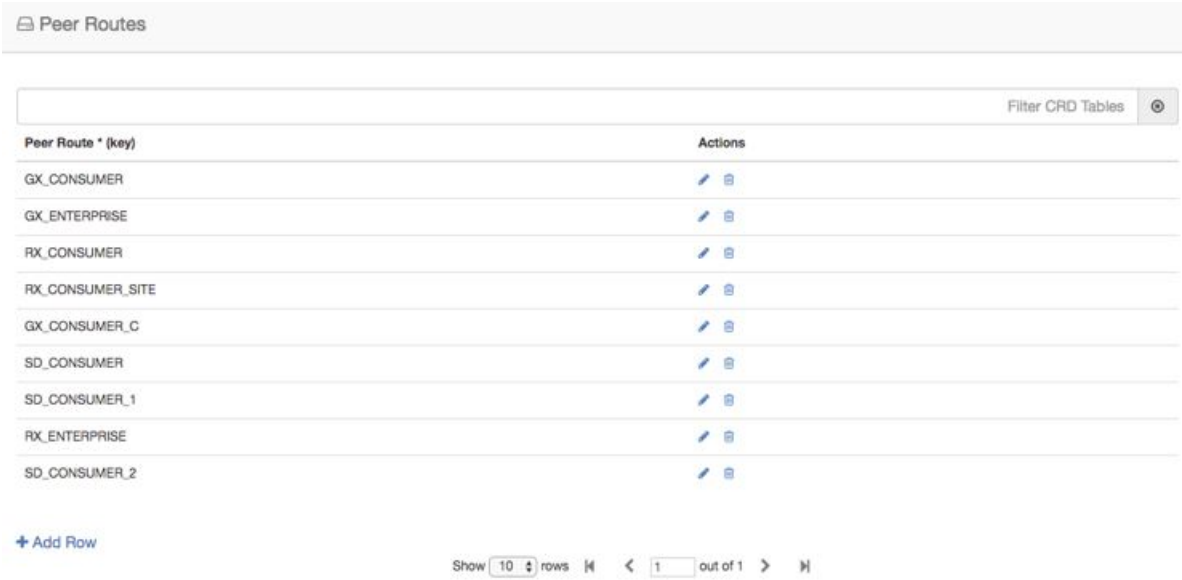
Show 10 rows 1 out of 1

Configure Peer Load Balancing

CPS vDRA has capabilities to route messages based on weight and precedence of the peer. Perform the following steps to define the peer load balancing:

1. Configure the Peer Routes table:
   - This table defines the name of the Peer Routes that are then used in the Peer Routing table.
   - This table is mainly used in Table-driven routing where the peer route is derived from the application table-driven rule tables.

Figure 17: Peer Routes



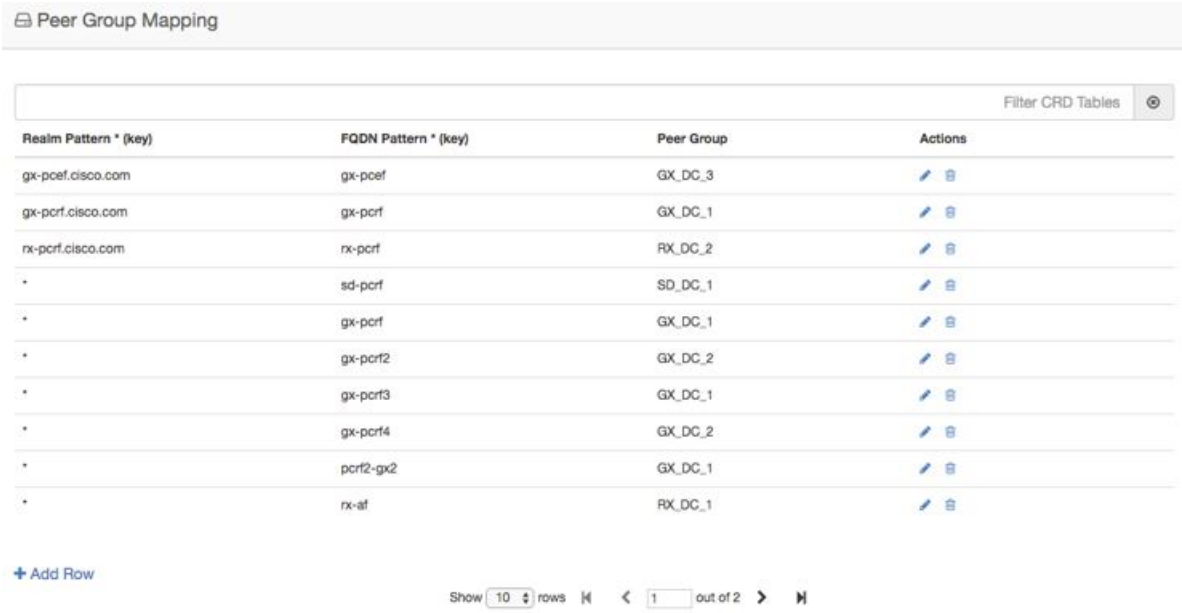
2. Configure the Peer Group Mapping table. This table defines the mapping of peers and realm with peer group.
   - Once the peer group is derived, CPS vDRA looks up this table to find the peers that belong to the derived peer group. Once DRA lists the peers in the peer group, it tries to match these peers with the Active Peer list, that is, peers which are currently connected to CPS vDRA.

     If multiple peers are up in a peer group, CPS vDRA load balances the traffic in a round-robin manner according to peer weight.

     The peer weight range is 0-1000 and the default weight is 100. If a peer weight is 0, then that peer is skipped.

     For example, if there are two peers up in a peer group with weights 100 and 200 respectively, then CPS vDRA load balances traffic between the two peers in the ratio of 33% and 67% respectively.
   - This table is applicable to SRK and Table driven routing only and is not used in Destination Host routing.

Figure 18: Peer Group Mapping



3. Configure the Peer Routing table.

   This table requires the following inputs:

   - Peer Route – derived from the Peer Route table
   - System Id – the current system ID of the CPS vDRA system
   - Peer Group – derived from the Peer Group Mapping table

   The outputs of this table are:

   - Precedence
   - Weight

   Precedence and Weight:

   Weight and Precedence are used to load balance the traffic among peers.

   If two peer groups are configured for same peer route and two peer groups are active then use the Precedence to select the row. If precedence is same then based on weight traffic will be load balanced among the two peer groups.

   In the following example, PR_1 and PR_2 have same precedence and PG_1 and PG_2 are active. Hence, traffic will be load balanced between PG_1 and PG_2 in the 1:2 ratio.

```
peer route   system ID  peer group  precedence  weight
PR_1             *       PG_1         10         10
PR_1             *       PG_2         10         20
```

   If two peer routes have different precedence and both peer groups are active then, peer group with lowest precedence value will be selected. In the following example, PG_1 will be selected.

```
peer route   system ID  peer group  precedence  weight
PR_1             *       PG_1         5         10
PR_1             *       PG_2         10        20
```

Figure 19: Peer Routing

Peer Routing

Filter CRD Tables

Peer Route * (key)	System Id * (key)	Peer Group * (key)	Precedence *	Weight *	Actions
SD_CONSUMER_2	system-1	SD_DC_1	1	1	
GX_CONSUMER	system-1	GX_DC_1	1	1	
RX_CONSUMER	system-1	GX_DC_1	1	1	
RX_CONSUMER	system-1	GX_DC_2	1	1	
SD_CONSUMER	system-1	SD_DC_2	1	1	
GX_CONSUMER_C	system-1	GX_DC_3	1	1	
SD_CONSUMER_1	system-1	SD_DC_2	1	1	

+ Add Row

Show 10 rows 1 out of 1

Configure Message Retries

CPS vDRA has the capability of retrying messages to multiple connected peers, in case the destination peer is not up. Retry mechanism works completely on the basis of SRK.

To configure message retries, you need to configure the Message Retry Profile table.

Figure 20: Message Retry Profile

Message Retry Profile

Filter CRD Tables

Peer Group * (key)	Application Id * (key)	Command Code * (key)	Result Code * (key)	Experimental RC	Number Of Retries	Actions
GX_DC_1	16777238	272	3003	false	6	
GX_DC_1	16777238	272	3004	false	6	
GX_DC_2	16777238	272	3003	false	6	
GX_DC_2	16777238	272	3004	false	6	
match=SD_DC.*	16777303	8388637	3004	false	6	
RX_DC_1	16777236	258	7000	false	6	
GX_DC_2	16777238	272	7000	false	6	

+ Add Row

Show 10 rows 1 out of 1

You can configure message retry on the basis of the following inputs:

- Peer Group

- Application Id
- Command Code
- Result-Code

The outputs of this table are:

- Number of retries
- Experimental RC

When the retry criteria matches, then:

1. First, the retry is done on any connected peer in the same peer group.
2. If no peer is found in the same peer group, the next priority is given to the peers in the peer group having the same SRK as the peer group to which the request was originally sent.
3. If the above condition also fails to find a peer, the last priority is given to the peers in the peer group that share the same second label as that of the original peer group.

Note CPS vDRA uses the Peer Group Mapping table to find the peer in the same peer group. Hence, the Peer Group Mapping table configuration is a prerequisite.

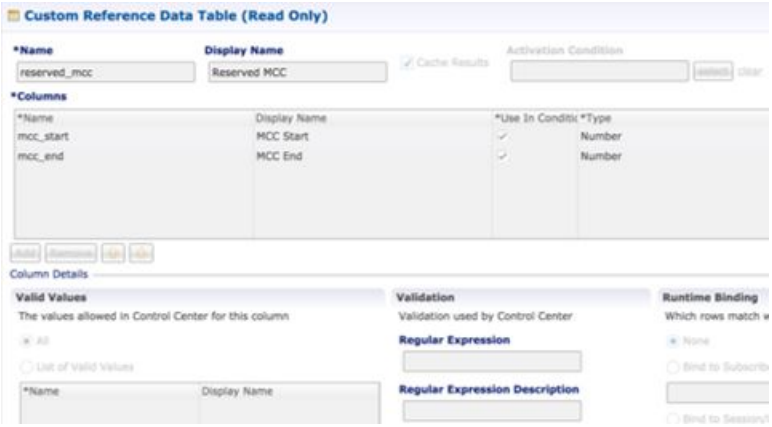
At the end, if no peer is found, retries stop. The retry also stops when the number of retries is exhausted and no response is received.

Configure Reserved IMSIs

You can now specify a reserved MCC range so that vDRA validates a parsed IMSI for SLF routing against a configured list of reserved MCC. If the IMSI matches a reserved IMSI, the value is ignored for SLF routing.

Configure the Reserved IMSI CRD table with columns for MCC Start Range and MCC End Range.

Figure 21: Reserved IMSI CRD Table



For more information, see Reserved IMSI.

In DRA, configure the MCC Start Range and MCC End Range.

Figure 22: MCC Range

Any calls within Reserved IMSI range are either routed by alternate means such as table-driven routing or result with an error.

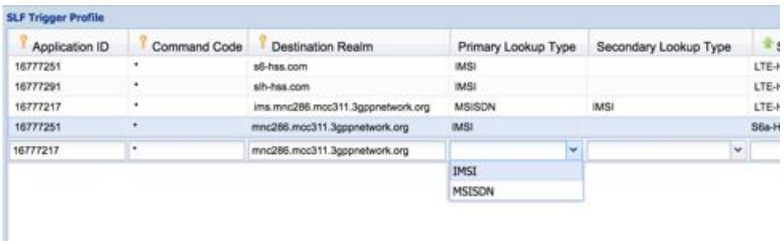
Configure Multiple Lookup in SLF Trigger Profile

Previously, in vDRA, the SLF lookup Type in the SLF trigger table had options only to support two types of lookup, that is, IMSI, MSISDN.

You can now specify Primary and Secondary Lookup Keys in the SLF Trigger Profile Table.

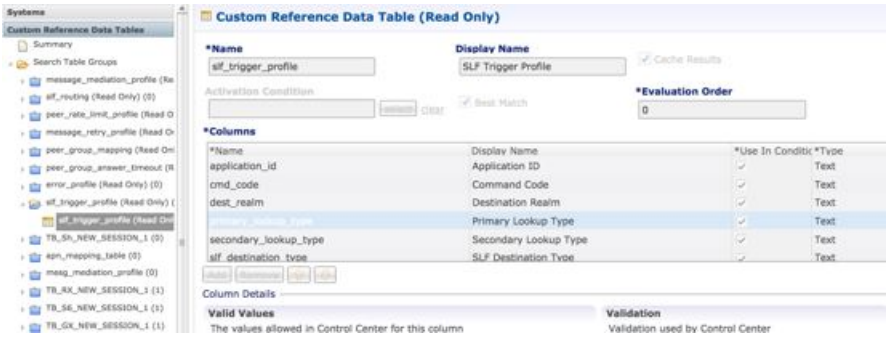
First, configure the columns in the CRD table as shown:

Figure 23: CRD Table Configuration



In the SLF Trigger Profile, you can select the Primary Lookup key from the drop down list. Similarly, you can select the Secondary Lookup key.

Figure 24: SLF Trigger Profile



Modify Result Code for AVPs Using Mediation Rules

You can configure CRDs to modify the result code of AVPs and overwrite them with a specified value.

1. Define the AVPs in the Diameter Avp Dictionary as shown in the following example for ResultCode:

   Figure 25: ResultCode AVP in Diameter Avp Dictionary

   Diameter Avp Dictionary

   Filter CRD Tables

Name * (key)	Avp Code	Vendor Id	Avp Type	Actions
ResultCode	268	10415	Unsigned32	

   + Add Row

   Show 10 rows 1 out of 1

2. Add the condition that is used to match AVPs with a particular result code. In this example, ResultCodeIs3xxx matches the result code AVP with regular expression 3.* and checks for any 3xxx result code.

   Figure 26: Add Result Code Condition in Avp Condition Profile

   Avp Condition Profile

   Filter CRD Tables

Profile Name * (key)	Avp * (key)	Avp Value *	Actions
ResultCodeIs3xxx	ResultCode	match=3.*	

   + Add Row

3. Define the action to be performed in the Avp Action Profile. In this case, ChangeResultcode is used to overwrite AVP values with the value 3002.

   Figure 27: Define Action in Avp Action Profile

   Avp Action Profile

   Filter CRD Tables

Profile Name * (key)	Avp * (key)	Avp Action	Avp Value	Actions
ChangeResultCode	ResultCode	Overwrite with value	3002	

4. Add the mediation rule in the Message Mediation Profile.

   In the following example, the mediation rule is considered on answer in ingress direction, when application is 16777251, command is 318, and condition profile ResultCode3xxx is met. When all criteria are satisfied, the “ChangeResultCode” action profile is applied.

   Figure 28: Mediation Rule

   Message Mediation Profile

   Filter CRD Tables

Application Id * (key)	Command Code * (key)	Message Type * (key)	Peer Group * (key)	Message Direction * (key)	Condition Profile * (key)	Action Profile * (key)	Actions
16777251	318	Answer	*	Ingress	ResultCodeIs3xxx	ChangeResultCode	

Hide Topology Using Mediation Rules

Define mediation rules to hide topology.

1. Define the AVPs in the Diameter Avp Dictionary as shown in the following example for OriginHost and OriginRealm:

   Figure 29: Diameter Avp Dictionary

   Diameter Avp Dictionary

   Filter CRD Tables

Name * (key)	Avp Code	Vendor Id	Avp Type	Actions
OriginHost	264	10415	OctetString	
OriginRealm	296	10415	OctetString	

2. Define the action to be performed in the Avp Action Profile. In this case, Hiding is used to overwrite the Origin Host AVP with the value “dra” and to overwrite the Origin Realm AVP with the value “dra.cisco.com”.

   Figure 30: Define Action in Avp Action Profile

   Avp Action Profile

   Filter CRD Tables

Profile Name * (key)	Avp * (key)	Avp Action	Avp Value	Actions
Hiding	OriginHost	Overwrite with value	dra	
Hiding	OriginRealm	Overwrite with value	dra.cisco.com	

3. Add the mediation rule in the Message Mediation Profile.

   In the following example, mediation rule is considered on request and answer in egress direction. It is applicable to all application, all commands, all peer groups and all message types (request/answer). When criteria satisfied, the “Hiding” action profile is applied.

   Figure 31: Mediation Rule

   Message Mediation Profile

   Filter CRD Tables

Application Id * (key)	Command Code * (key)	Message Type * (key)	Peer Group * (key)	Message Direction * (key)	Condition Profile * (key)	Action Profile * (key)	Actions
*	*	*	*	Egress	-	Hiding	

Configure Mediation Rule for Proxy-Unaware Endpoints

When endpoints are not proxy-aware, PCRF may send an RAR to the vDRA without Destination Host AVP because it is not proxy friendly.

In such cases, you can configure vDRA to modify the Destination-Host of the RAR based on a regex match of the Diameter Session-ID.

1. Define the AVPs in the Diameter Avp Dictionary as shown in the following example for DestinationHost and SessionId:

Figure 32: Diameter Avp Dictionary

Diameter Avp Dictionary

Filter CRD Tables

Name * (key)	Avp Code	Vendor Id	Avp Type	Actions
DestinationHost	293	10415	OctetString	
SessionId	263	0	OctetString	

2. Define the action to be performed in the Avp Action Profile. In this case, ExtractHostFromSessionId is used to overwrite the DestinationHost AVP with the value dynamically derived from the regex string:

```
SubString("${SessionId}", "([^;]+);.*" , 1)
```

The session ID is taken from the message, then regex ([^;]+);.* is applied on the session ID and host is extracted.

For example, if the session ID is “pcef;1450914337;172.30.96.2;0”.

Regex ([^;]+);.* is applied on session ID and it will get 2 groups:

- group1= “pcef”
- group2 = “1450914337;172.30.96.2;0”

Hence, the Destination Host is set with value pcef.

Figure 33: Define Action in Avp Action Profile

Avp Action Profile

Filter CRD Tables

Profile Name * (key)	Avp * (key)	Avp Action	Avp Value	Actions
ExtractHostFromSessionId	DestinationHost	Overwrite with value	SubString("${SessionId}", "([^;]+);.*" , 1)	

3. Add the mediation rule in the Message Mediation Profile.

In the following example, mediation rule is considered on Gx RAR in ingress direction. When all the criteria are met, the “ExtractHostFromSessionId” action profile is applied. Action will set the DestinationHost AVP just after receiving message.

Figure 34: Mediation Rule

Message Mediation Profile

Filter CRD Tables

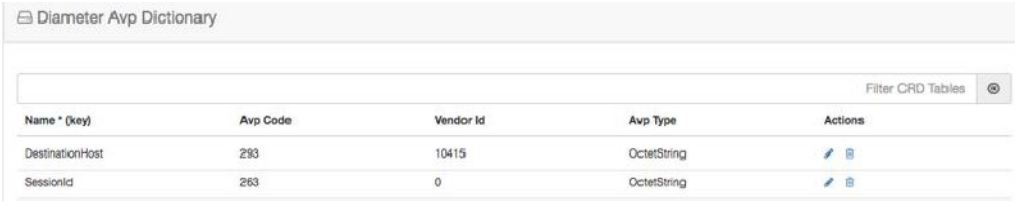
Application Id * (key)	Command Code * (key)	Message Type * (key)	Peer Group * (key)	Message Direction * (key)	Condition Profile * (key)	Action Profile * (key)	Actions
16777238	258	Request	*	Ingress	*	ExtractHostFromSessionId	

Add Prefix to an AVP Using Mediation Rules

You can define a prefix for an AVP.

1. First, define the AVPs in the Diameter Avp Dictionary as shown in the following example for the AVP, DestinationHost:

Figure 35: Defining DestinationHost in Diameter Avp Dictionary



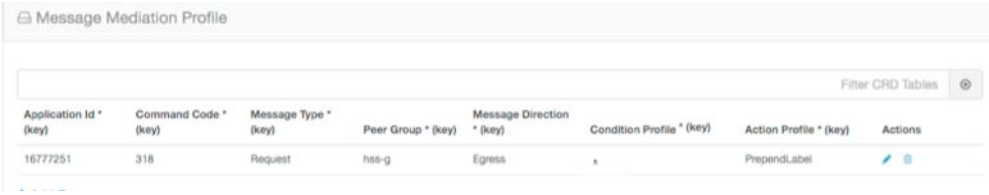
2. Define the PrependLabel profile in the Avp Action Profile that adds the prefix "core-" to the DestinationHost AVP.

Figure 36: Define PrependLabel Profile in Avp Action Profile

3. Add the mediation rule in Message Mediation Profile as shown in the following example.

   In this example, the mediation rule is applied on s6 AIR request message in egress direction. When the peer group also matches hss-g, it will apply the “PrependLabel” action profile. Action will add the prefix “core-“ to destination host.

Figure 37: Mediation Rule in Message Mediation Profile



Configure Throttling of Diameter Messages Using DOIC

About DOIC

vDRA can throttle or divert Diameter requests towards PCRF, HSS, AAA, and OCS servers based on reporting of overloaded conditions using the architecture described in RFC 7683 Diameter Overload Indication Conveyance (DOIC).

The DOIC feature in vDRA involves the following high-level processes:

DOIC Enablement
: DOIC feature in enabled/disabled in vDRA and also for the peer group.

DOIC Capability Exchange
: If the DOIC feature is enabled, then for every request received, vDRA adds the OC-Supported-Features AVP to the request message to announce the DOIC support for loss of algorithm.

OCS Map
The OCS Map is the local vDRA cache, which contains the information of each reporting node with its latest overload control state information. On receipt of OC-OLR AVP from the peer, vDRA looks for the entry in OCS Map and updates the overload control state object.

Peer Overload Detection
vDRA checks the OCS entry from the OCS Map to determine whether the overload treatment is required or not.

Table Lookup
vDRA queries the Message Class Profile table to retrieve the Message classification. The message classification is used to query the DOIC Profile table to determine the Abatement Action that must be applied.

Overload Treatment
There are three types of overload treatment supported: Forward, Divert, and Drop. For more information, see DOIC Profile.

Configuring DOIC in vDRA

To configure vDRA for throttling, perform the following steps:

1. Enable DOIC feature in vDRA in the Policy Builder as described in Enable DOIC, on page 5.
2. Enable DOIC for the peer group using the Peer Group SRK mapping table as described in Peer Group SRK Mapping.
3. Define the AVP condition in the Diameter AVP Dictionary table. For more information, see Diameter Avp Dictionary.
4. Configure the Message Class Profile table to get the message classification as output. For more information, see Message Class Profile.

   If AVP conditions are evaluated to be true, then get Message Class. Message Class can be one of P0, P1, P2, P3, P4.
5. Use the Message class to define the abatement action in the DOIC Profile table. For more information, see DOIC Profile.

Configure Throttling of Diameter Messages Using DRMP

vDRA can throttle Diameter requests based on the message priority sent in the Diameter request using the architecture described in RFC 7944 Diameter Routing Message Priority (DRMP).

To configure vDRA to use DRMP, perform the following steps as described in Configure Throttling of Diameter Messages Using DOIC, on page 22. Ensure you use the DRMP AVP in the Diameter Avp Dictionary table.

With this configuration, messages with Message Class P0 are not throttled in either rate limiter or by DOIC. For the rest of the messages, the throttling is applied as configured in DOIC table or in rate limiter.

Rate limiter currently throttles message regardless of the message class/ DRMP value. However, it will not throttle the message with message class P0.

Configure vDRA for eMPS

You can configure vDRA to recognize Diameter messages as Enhanced Multimedia Priority Services (eMPS) based on Diameter Application-Id, Command-Code, and AVP values.

An eMPS request is marked as a Priority 0 (or P0 in message classification) request and is not throttled or dropped.

1. Specify the AVP in the Diameter Avp Dictionary as shown in the following example:

   Figure 38: Diameter Avp Dictionary

   Diameter Avp Dictionary

   Filter CRD Tables

Name * (key)	Avp Code	Vendor Id	Avp Type	Actions
*				
Called-Station-Id	30	10415	OctetString	
MPS-Identifier	528	10415	OctetString	

2. Specify the AVP condition in the Avp Condition Profile table.

   Figure 39: Avp Condition Profile

   Avp Condition Profile

   Filter CRD Tables

Profile Name * (key)	Avp * (key)	Avp Value *	Actions
*	*		
IsEmergency	Called-Station-Id	911	

3. In Message Class Profile, configure the Message/Request Type as ‘None’, Message Class as P0, and Ingress Peer Group as ‘*’, so that the message is treated as eMPS irrespective of the type of message (request or response) and the peer group it is sent to.

   Figure 40: Message Class Profile

   Message Class Profile

   Filter CRD Tables

Ingress Peer Group *	Application Id *	Command Code *	Message/Request Type *	Condition Profile *	Message Class	Actions
*	1677238	272	None	IsEmergency	P0	

   + Add Row

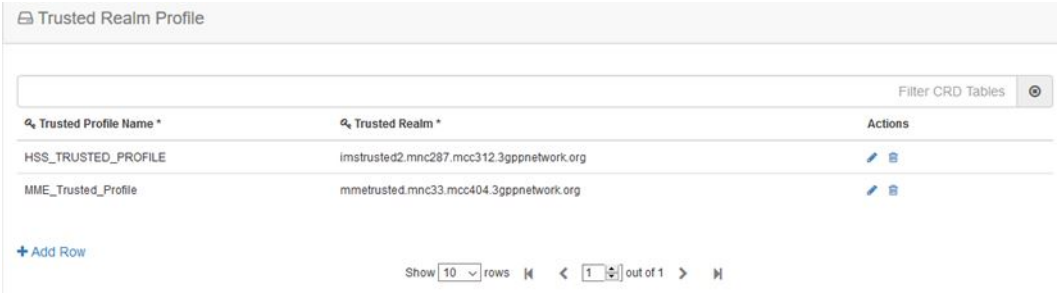
   Show 10 rows 1 out of 1

Configure Topology Hiding for S6a/d Diameter Application

Configure topology hiding for the S6a/d Diameter application (Diameter interface between the MME/SGSN and the HSS).

1. Configure a trusted realm profile: Define the name of a Trusted Profile and associate a realm which can be trusted with it.

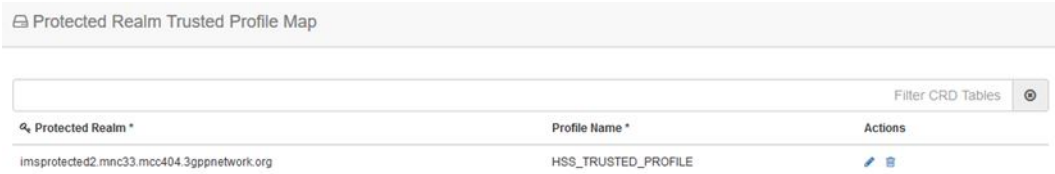
   Figure 41: Trusted Realm Profile

2. Configure the Protected realm (where vDRA is to provide the topology hiding) and link it to the trusted profile.

   Depending on the requests, if the destination/origin realm is trusted, vDRA skips the topology hiding. If the realm is not trusted, vDRA provides topology hiding.

   Figure 42: Protected Realm Trusted Profile

3. In the MME Alias Map, define the MME FQDN, which needs to be protected, and bind it to the Aliases that are used for topology hiding.

   Figure 43: MME Alias Map

4. Define all the HSS Aliases that are used for HSS topology hiding by vDRAin the HSS Aliases table. Specify whether the defined alias is shared or not by multiple protected HSSs.

Figure 44: HSS Aliases

5. In the HSS Alias Map, map the protected HSS FQDN with the HSS ALIAS defined in previous HSS Aliases table. These aliases are used for the topology hiding of the protected HSSs.

Figure 45: HSS Alias Map

Manual Peer Disconnection using REST API

CPS vDRA provides a REST API that you can use to manually disconnect peer connections from vDRA. To disconnect a single peer connection, you need to provide the peer connection key information in the API method.

Disconnect a single peer connection

To disconnect a single peer, provide the peer-connection-key (that you can find in the peer endpoint details) in the PUT method as shown:

```
PUT localActivePeerEndpoints/disconnect/key/<peer-connection-key>
```

Responses:

```
200 OK
{
    "success": {
        "code": 0,
        "message": "Request completed successfully"
    }
}
```

Note that success means the request is accepted, and vDRA attempts to disconnect the peer connection. The success response does not indicate that the peer connection has been disconnected.

Failure response:

```
404 Not Found

{
    "error": {
        "code": 2014,
        "message": "Data for key <peer-connection-key> is not found"
    }
}
```

Policy DRA Relay Configuration

Policy DRA requires a full-mesh topology to be able to send traffic between different sites. In a multi-site DRA network, relay connections are required between every pair of relay endpoints.

Relay Endpoint Configuration

Relay endpoint is used to indicate the IP address/port combination at which vDRA listens for diameter connections. These connections are used to send/receive diameter traffic between sites.

Relay endpoints for vDRA can be configured using the Policy Builder vDRA configuration plugin.

The following fields are required for the configuration:

Vm Host Name
VM Host Name is the host name of the VM. The value '*' can be used to match any host name.

Instance Id
The Instance ID field should be set to '1'. Currently, no other value is supported.

Ip Address
IPv4 or IPv6 address of the interface used to listen for relay connections. Currently, only physical IP addresses are supported. VIP addresses are not supported.

Realm
Supports configurable relay endpoint realm name.

Port
Port that is used to listen for relay connections. The default value is '4868'. Currently, the range of ports supported is '4868-4878'.

Fqdn
FQDN is the string that is used to send the relay endpoint information to the other sites.

Enabled
Used to enable or disable the relay endpoint. It should be enabled for the relay endpoint to start listening on the configured port and IP address.

Note Ensure that the DNS configuration is completed before publishing the PB changes. If DNS configuration changes are made later, the Diameter endpoints must be restarted for the changes to take effect.

Control Plane Configuration

The control plane configuration performs the following functions:

- If the IP address is a site-local address, a global Control Plane server is created that listens to connections.
- If the IP address not a site-local address, the connections are initiated from diameter-endpoint containers to the global Control Plane server running at specified IP/port. These connections are used to send and receive global control plane information between sites.

Note It is recommended not to use VIP address. Instead use physical IP address for control-plane relay link.

The following example illustrates control plane configuration:

```
admin@orchestrator# show running-config control-plane
control-plane relay gx-dra1-relay-v6-1
 address [2001:421:27c1:913:250:56ff:fea6:12]
 port    6379
```

Note The IPv6 address must be encapsulated with square [] brackets.

DNS Host Configuration

The following entities must be DNS resolvable and must have DNS Host configuration entries:

- Control Plane servers from all sites
- Relay Endpoint FQDNs from all sites

The following command is an example of DNS Host configuration:

```
admin@orchestrator# show running-config network dns host
network dns host gx-dra1-relay-v6 local
 address 2001:421:27c1:913:250:56ff:fea6:48
```

Virtual-IP Configuration

The following entities must have virtual-IP configuration entries:

- Control Plane Servers with VIP addresses

Relay Configuration for a 6-Site Policy DRA System

The following example describes how to configure the relay configuration for a six site Policy DRA system.

Relay Endpoints

There must be two relay endpoint entries configured in Policy Builder per site; one entry each for the two diameter-endpoint containers.

Control Plane Configuration

There must be at least two global Control Plane servers configured per site; one on each diameter-endpoint for redundancy.

The global Control Plane servers must have connections between each other. This means that every site must have control plane configuration entries for the control plane servers of every other site.

For a six site system, every site must have 12 Control Plane configuration entries (6 sites x 2 global Control Plane servers per site) configured in CLI.

DNS Host Configuration

There should be one DNS Host entry for each of the global Control Plane configuration entries. In a six site system, every site should have 12 DNS Host entries related to global Control Plane entries of all sites.

There should be one DNS Host entry for each Relay endpoints of every site. In a six site system, every site should have 12 DNS Host entries related to Relay Endpoint entries of all sites.

Virtual-IP Configuration

If any global Control Plane servers have VIP addresses, they should have Virtual-IP Configuration.

Every site should have Virtual-IP Configuration for any VIP addresses that are associated to that site.

Configuring Application based Sharding

Restriction

- Migration from hash-based sharding to zone based sharding is not supported.
- Since this feature supports only static sharding configurations, all database configurations needs to be committed with single commit operation.
- Currently, dynamic addition of new shards after initial configuration is not supported as it supports only static shard configurations. If one needs to add new shards or edit existing shards, it is mandatory to clean up the complete database and reconfigure again.

Procedure

Step 1 Login to Binding VNF CLI to configure application sharding on Binding VNF.

a) Configure sharded cluster master.

```
database cluster session sharded-cluster-master <true | false>
```

true: enables sharded cluster master. Internally sharding metadata creation gets triggered.

false: Sharding metadata creation is not enabled.

Example:

```
database cluster session sharded-cluster-master true
```

Note
Only one site among the mated pair which initially gets configured has to be enabled (true) so that sharding metadata creation happens from that site. For rest of the sites, it just adds members and uses existing sharding metadata.

b) Configure multi-database collections: This configuration enables the creation of internal logical shards.

```
database cluster session multi-db-collections <1..4>
```

Default: 1

For example, if configured with 2, each sharding database metadata creates an extra logical shard for the shard configured as a part of database creation. So total number of shards will be double the number.

```
database cluster session multi-db-collections 2
```

c) Configuring sharding database.

```
database cluster <cluster name> sharding-db <sharding-db-name> address <ip address> [port <port>]
database cluster <cluster name> sharding-db-seed <shardingdb name>
```

Example:

```
database cluster session sharding-db shdb-4 address 192.168.11.43
database cluster session sharding-db shdb-5 address 192.168.11.44
database cluster session sharding-db-seed shdb-4
```

Note
Port is optional. By default, sharding database uses port 27019.

d) Configure shards.

```
database cluster <cluster-name> shard <shard-name>
```

Non-arbiter shard member:

```
database cluster <cluster-name> shard <shard-name> shard-server <server-name> storage-engine MMAPv1
 address <ipaddress> port <port> priority <priority>
```

Arbiter shard member:

```
database cluster <cluster-name> shard <shard-name> shard-server <server-name> arbiter true
storage-engine MMAPv1 address <ipaddress> port <port>
```

Shard seed:

```
database cluster <cluster-name> shard <shard-name> shard-server-seed <seed-server>
```

Note
Seed server has to be one of the non-arbiter member from the same shard.

Example:

```
database cluster session shard shard-21
database cluster session shard shard-21 shard-server server-x storage-engine MMAPv1 address
192.168.11.43 port 27026 priority 10
database cluster session shard shard-21 shard-server server-y storage-engine MMAPv1 address
192.168.11.44 port 27026 priority 5
database cluster session shard shard-21 shard-server arbiter-21 arbiter true storage-engine MMAPv1
 address 192.168.11.42 port 27026
database cluster session shard shard-21 shard-server-seed server-x
```

e) Make sure all the databases are UP with appropriate status using `show database status` commad.

Sample output:

```
admin@orchestrator[an-dbmaster]# show database status cluster-name session
                                                          CLUSTER
ADDRESS        PORT   NAME        STATUS     TYPE         NAME     SHARD     REPLICA SET
-----------------------------------------------------------------------------------------------
192.168.11.42  27026  arbiter-21  ARBITER    replica_set  session  shard-21  rs-shard-21
192.168.11.43  27026  server-x    PRIMARY    replica_set  session  shard-21  rs-shard-21
192.168.11.44  27026  server-y    SECONDARY  replica_set  session  shard-21  rs-shard-21
192.168.11.42  27027  arbiter-22  ARBITER    replica_set  session  shard-22  rs-shard-22
192.168.11.43  27027  server-x    SECONDARY  replica_set  session  shard-22  rs-shard-22
192.168.11.44  27027  server-y    PRIMARY    replica_set  session  shard-22  rs-shard-22
192.168.11.43  27019  session     PRIMARY    shard_db     session  shdb-4    session-sharddb
192.168.11.44  27019  session     SECONDARY  shard_db     session  shdb-5    session-sharddb

admin@orchestrator[an-dbmaster]#
```

Step 2 Login to DRA VNF CLI to configure application sharding on DRA VNF.

a) Configure shard metadata database connection.

```
binding shard-metadata-db-connection <dbName> <address> <port>
```

Example:

```
binding shard-metadata-db-connection drasession 182.22.31.207 27019
```

- <*dbName*>: Database name for which connection URI needs to be set.

  Possible values are: all, drasession, imsiapn, ipv4, ipv6, msisdnapn, range.
- <*address*>: Address of the binding shard metadata database. This is either an IP address or an FQDN.
- <*port*>: Port of the binding shard metadata database.

Step 3 Login to DRA VNF CLI to configure same connection pool on imsiapn-msisdnapn database.

Note
We do not recommended configuring connection pool for small setups. It is required for setups whose database spans across 48 shards or more.

a) Configure same connection pool on imsiapn-msisdnapn database transactions.

```
binding cluster-binding-dbs imsiapn-msisdnapn
```

Example:

```
binding  cluster-binding-dbs imsiapn-msisdnapn
```

Example

The following are the examples for complete set of database cluster configuration on Binding VNF.

```
config
database cluster session sharded-cluster-master true
database cluster session multi-db-collections 2
database cluster session sharding-db shdb-4 address 192.168.11.43 port 27019
database cluster session sharding-db shdb-5 address 192.168.11.44 port 27019
database cluster session sharding-db-seed shdb-4

database cluster session shard shard-21
database cluster session shard shard-21 shard-server server-x storage-engine MMAPv1 address
 192.168.11.43 port 27026 priority 10
database cluster session shard shard-21 shard-server server-y storage-engine MMAPv1 address
 192.168.11.44 port 27026 priority 5
database cluster session shard shard-21 shard-server arbiter-21 arbiter true storage-engine
 MMAPv1 address 192.168.11.42 port 27026
database cluster session shard shard-21 shard-server-seed server-x

database cluster session shard shard-22
database cluster session shard shard-22 shard-server server-y storage-engine MMAPv1 address
 192.168.11.44 port 27027 priority 10
database cluster session shard shard-22 shard-server server-x storage-engine MMAPv1 address
 192.168.11.43 port 27027 priority 5
database cluster session shard shard-22 shard-server arbiter-22 arbiter true storage-engine
 MMAPv1 address 192.168.11.42 port 27027
database cluster session shard shard-22 shard-server-seed server-y
commit
```

The following is an example to configure shard metadata database connection for session and all bindings on DRA VNF.

```
config
binding shard-metadata-db-connection drasession  182.22.31.207  27019
binding shard-metadata-db-connection ipv6  182.22.31.207  27019
binding shard-metadata-db-connection ipv4  182.22.31.208  27019
```

```
binding shard-metadata-db-connection imsiapn  182.22.31.208  27019
binding shard-metadata-db-connection msisdnapn  182.22.31.208  27019
commit
```

Configuring Binding Database Overload

Procedure

Login to DRA VNF CLI to configure maximum record limit on session and bindings database.

```
binding db-max-record-limit <db-Name> <limit>
```

<db-Name>: Database name on which maximum record limit has to be set.

Possible values are drasession, imsiapn, ipv4, ipv6, msisdnapn.

<limit>: Numeric value indicating maximum records that could be stored in given database.

Example:

```
binding db-max-record-limit drasession 10000
```

Example

The following is an example for setting maximum record limit on session and all bindings database:

```
config
binding db-max-record-limit drasession 10000
binding db-max-record-limit ipv6 10000
binding db-max-record-limit ipv4 5000
binding db-max-record-limit imsiapn 5000
binding db-max-record-limit msisdnapn 5000
commit
```

Change Admin User Password for MongoDB Authentication

Caution The following procedure is service impacting and needs to be execute on all sites at the same time.

Procedure

Step 1 Execute the following step on all binding VNF of all sites.

```
db-authentication change-password database mongo user adminuser
```

When prompt is displayed, enter the current password and new password.

```
admin@orchestrator[binding-master]# db-authentication change-password database mongo user
adminuser
```

```
Value for 'current-password' (<string>): ******
Value for 'new-password' (<string>): ********
result SUCCESS
```

Step 2 Execute the following step on all DRA VNF of all sites.

```
db-authentication change-password database mongo user adminuser
```

When prompt is displayed, enter the current password and new password.

```
db-authentication change-password database mongo user
adminuser
Value for 'current-password' (<string>): ******
Value for 'new-password' (<string>): ********
result SUCCESS
```

Step 3 Execute `db-authentication sync-password database mongo` command on all binding VNF of all sites.

```
admin@orchestrator[binding-master]# db-authentication sync-password database mongo
result
SUCCESS : Mongo password sync successful
```

Step 4 Execute `db-authentication sync-password database mongo` command on all DRA VNF of all sites.

```
db-authentication sync-password database mongo
result
SUCCESS : Mongo password sync successful
```

Configuring MongoDB Authentication

Before you begin

- Currently, MongoDB authentication is supported only for fresh deployments of vDRA where application sharding has been implemented.
- MongoDB authentication is not supported for MongoDB sharding deployments.
- Make sure every shard of database cluster has Primary member present. Execute the following command to verify the same.

  ```
  show database status | tab | include PRIMARY
  ```
- Make sure all the VMs are in CONNECTED state.

  ```
  show docker engine
  ```
- Make sure there are no IP_NOT_REACHABLE alerts present on the system.

  ```
  show alert status | tab | include IP_NOT_REACHABLE
  ```
- Make sure the network cache is up to date with all IPs present on each VM.

  ```
  show network ips
  ```

Procedure

Step 1 Login to Binding VNF CLI and configure MongoDB authentication on single node setup:

a) Set password.

```
db-authentication set-password database mongo password *****
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication set-password database mongo password
Value for 'password' (<string>): ********
result SUCCESS
admin@orchestrator[an-dbmaster]#
```

b) Enable transition authentication.

```
db-authentication enable-transition-auth database mongo
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication enable-transition-auth database mongo
admin@orchestrator[an-dbmaster]#
```

c) Rolling restart of mongod instances.

```
db-authentication rolling-restart database mongo
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication rolling-restart database mongo
admin@orchestrator[an-dbmaster]#
```

d) Monitor rolling restart status.

```
db-authentication rolling-restart-status database mongo
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication rolling-restart-status database mongo
result
Rolling Restart: Not Scheduled/Completed
admin@orchestrator[an-dbmaster]#
```

e) Disable transition authentication.

```
db-authentication disable-transition-auth database mongo
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication disable-transition-auth database mongo
admin@orchestrator[an-dbmaster]#
```

f) Rolling restart of mongod instances.

```
db-authentication rolling-restart database mongo
```

Example:

```
admin@orchestrator[an-dbmaster]# db-authentication rolling-restart database mongo
admin@orchestrator[an-dbmaster]#
```

Note
During `db-authentication rolling-restart` command execution mongod instances are restarted.

g) Make sure all the databases are UP with correct status.

```
show database status
```

Sample output:

```
admin@orchestrator[an-dbmaster]# show database status
                                                        CLUSTER
ADDRESS        PORT   NAME        STATUS     TYPE        NAME     SHARD     REPLICA SET
-----------------------------------------------------------------------------------------------
```

```
192.168.11.42  27036  arbiter-1   ARBITER    replica_set  binding  shard-1   rs-shard-1
192.168.11.43  27036  server-a    PRIMARY    replica_set  binding  shard-1   rs-shard-1
192.168.11.44  27036  server-b    SECONDARY  replica_set  binding  shard-1   rs-shard-1
192.168.11.42  27037  arbiter-2   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.42  27038  arbiter-3   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.43  27037  server-a    SECONDARY  replica_set  binding  shard-2   rs-shard-2
192.168.11.44  27037  server-b    PRIMARY    replica_set  binding  shard-2   rs-shard-2
192.168.11.41  27030  binding     SECONDARY  shard_db     binding  shdb-1    binding-sharddb
192.168.11.42  27030  binding     PRIMARY    shard_db     binding  shdb-2    binding-sharddb
```

h) DRA VNF Site-A and Site-B.

```
db-authentication set-password database mongo password *****
```

Example:

```
admin@orchestrator[an-master]# db-authentication set-password database mongo password
Value for 'password' (<string>): ********
result SUCCESS
admin@orchestrator[an-master]#
```

Step 2 Login to Binding VNF CLI and configure MongoDB authentication in mated pair deployments.

Note
During `db-authentication rolling-restart` command execution mongod instances are restarted.

a) On Site A, configure session-AB, imsi-msisdn databases.
b) On Site B, configure session-AB, imsi-msisdn databases.
c) On Site A, configure the password and enable-transition-auth and run rolling-restart.
d) On Site B, configure the password and enable-transition-auth and run rolling-restart.
e) On Site A, disable transition-auth and run rolling-restart
f) Make sure all the databases are UP with appropriate status.

```
show database status
```

Sample output:

```
admin@orchestrator[an-dbmaster]# show database status
                                                        CLUSTER
ADDRESS        PORT   NAME        STATUS     TYPE         NAME     SHARD     REPLICA SET
-----------------------------------------------------------------------------------------------
192.168.11.42  27036  arbiter-1   ARBITER    replica_set  binding  shard-1   rs-shard-1
192.168.11.43  27036  server-a    PRIMARY    replica_set  binding  shard-1   rs-shard-1
192.168.11.44  27036  server-b    SECONDARY  replica_set  binding  shard-1   rs-shard-1
192.168.11.42  27037  arbiter-2   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.42  27038  arbiter-3   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.43  27037  server-a    SECONDARY  replica_set  binding  shard-2   rs-shard-2
192.168.11.44  27037  server-b    PRIMARY    replica_set  binding  shard-2   rs-shard-2
192.168.11.41  27030  binding     SECONDARY  shard_db     binding  shdb-1    binding-sharddb
192.168.11.42  27030  binding     PRIMARY    shard_db     binding  shdb-2    binding-sharddb
```

g) DRA VNF Site-A and Site-B.

```
db-authentication set-password database mongo password *****
```

Example:

```
admin@orchestrator[an-master]# db-authentication set-password database mongo password
Value for 'password' (<string>): ********
result SUCCESS
admin@orchestrator[an-master]#
```

Disabling MongoDB Authentication

Note This section is used to disable MongoDB authentication in mated pair deployments.

Procedure

Step 1 Login to Binding VNF CLI and perform the following steps:

Note
The steps need to be performed on all binding VNFs.

a) Enable transition authentication.

```
db-authentication enable-transition-auth database mongo
```

b) Rolling restart of mongod instances.

```
db-authentication rolling-restart database mongo
```

c) Rolling restart status.

```
db-authentication rolling-restart-status database mongo
```

Step 2 Login to DRA VNF CLI and remove the password by using `db-authentication remove-password database mongo` command.

Note
The step needs to be performed on all DRA VNFs.

```
admin@orchestrator[an-master]# db-authentication remove-password
Value for 'password' (<string>): ********
result SUCCESS
admin@orchestrator[an-master]#
```

Step 3 Login to binding VNF CLI and remove the password by using `db-authentication remove-password database mongo` command.

Note
The step needs to be performed on all binding VNFs.

Step 4 Login to binding VNF CLI and perform the following steps:

Note
The steps need to be performed on all binding VNFs.

a) Disable transition authentication.

```
db-authentication disable-transition-auth database mongo
```

b) Rolling restart of mongod instances.

```
db-authentication rolling-restart database mongo
```

c) Rolling restart status.

```
db-authentication rolling-restart-status database mongo
```

Step 5 Make sure all the databases are UP with appropriate status.

```
show database status
```

Sample output:

```
admin@orchestrator[an-dbmaster]# show database status
                                                        CLUSTER
ADDRESS        PORT   NAME        STATUS     TYPE         NAME     SHARD     REPLICA SET
-----------------------------------------------------------------------------------------------
192.168.11.42  27036  arbiter-1   ARBITER    replica_set  binding  shard-1   rs-shard-1
192.168.11.43  27036  server-a    PRIMARY    replica_set  binding  shard-1   rs-shard-1
192.168.11.44  27036  server-b    SECONDARY  replica_set  binding  shard-1   rs-shard-1
192.168.11.42  27037  arbiter-2   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.42  27038  arbiter-3   ARBITER    replica_set  binding  shard-2   rs-shard-2
192.168.11.43  27037  server-a    SECONDARY  replica_set  binding  shard-2   rs-shard-2
192.168.11.44  27037  server-b    PRIMARY    replica_set  binding  shard-2   rs-shard-2
192.168.11.41  27030  binding     SECONDARY  shard_db     binding  shdb-1    binding-sharddb
192.168.11.42  27030  binding     PRIMARY    shard_db     binding  shdb-2    binding-sharddb
```

Configuring Zone Aware Sharding

Restriction

- Migration from hash-based sharding to zone based sharding is not supported.
- Database reconfigure steps must be followed to enable zone aware sharding on existing database cluster.
- Configure the databases on Binding VNF site by site. Not recommended to apply database configurations on all sites in parallel.
- Within single commit all the database configurations needs to be committed.
- Currently, dynamic addition of new shards after initial configuration is not supported.
- You should make sure that there are no overlapping IPv6 addresses within a zone or across multiple zones within a database cluster (or) across multiple database clusters while doing the database configuration. IPv6 address mentioned as the start/end values while configuring the ranges should be unique.
- If you want to add new shards or edit existing shards, it is mandatory to clean up the complete database and reconfigure again.

Procedure

Step 1 Login to Binding VNF CLI to configure database for zone sharding.

a) Configuring sharding database.

```
database cluster <cluster name> sharding-db <shardingdb name> address <VM ip address> [port
<portNum>]
```

```
database cluster <cluster name> sharding-db-seed <shardingdb name>
```

Example:

```
admin@orchestrator[an-dbmaster]#
database cluster binding sharding-db shdb-1 address 182.22.31.10 port 27020
database cluster binding sharding-db shdb-2 address 182.22.31.11
database cluster binding sharding-db shdb-3 address 182.22.31.12
database cluster binding sharding-db-seed shdb-1
```

Note
<portNum> is optional. By default, the port number is 27019.

b) Enable IPv6 zone sharding.

```
database cluster <cluster name> ipv6-zone-sharding true
```

Example:

```
admin@orchestrator[an-dbmaster]# database cluster binding ipv6-zone-sharding true
```

c) Create zone and its range(s).

```
database cluster <cluster name> ipv6-zones-range <zone-name> zone-range <range-name> start <pool
 starting Prefix> end <pool ending Prefix>
```

Example:

```
admin@orchestrator[an-dbmaster]# database cluster binding ipv6-zones-range pune zone-range range1
 start 2003:3051:0000:0001 end 2003:3051:0000:0500
```

Note
It is possible to create multiple ranges for each zone. Configure the ranges in 64-bit Framed IPv6 Prefixes only.

d) Mapping of zone to shard.

```
database cluster <cluster name> shard <shard name> zone-name <zone-name>
```

Example:

```
admin@orchestrator[an-dbmaster]# database cluster binding shard shard-1 zone-name pune
```

e) Sample database configuration with two shards (two zones with two ranges per zone).

Step 2 Login to the master(primary/seed) mongo sharding database and check configured ranges, zones, shard/range mapping information.

```
database cluster binding sharded-cluster-master true
database cluster binding ipv6-zone-sharding true
database cluster binding ipv6-zones-range mumbai zone-range r1 start 2008:5000:0000:0100 end 2008:5000:0000:0500
database cluster binding ipv6-zones-range mumbai zone-range r2 start 2009:5000:0000:0100 end 2009:5000:0000:0500
database cluster binding ipv6-zones-range pune zone-range r1 start 2011:6000:0000:0001 end 2011:6000:0000:0500
database cluster binding ipv6-zones-range pune zone-range r2 start 2012:6000:0000:0001 end 2012:6000:0000:0500
database cluster binding sharding-db shdb-1 address 182.22.31.10
database cluster binding sharding-db shdb-2 address 182.22.31.11
database cluster binding sharding-db shdb-3 address 182.22.31.12
database cluster binding sharding-db-seed shdb-1

database cluster binding shard shard-1
database cluster binding shard shard-1 shard-server server-a storage-engine MMAPv1 address 182.22.31.13 port 27017 priority 10
database cluster binding shard shard-1 shard-server server-b storage-engine MMAPv1 address 182.22.31.14 port 27017 priority 5
database cluster binding shard shard-1 shard-server arbiter-1 arbiter true storage-engine MMAPv1 address 182.22.31.11 port 27017
database cluster binding shard shard-1 shard-server-seed server-a

database cluster binding shard shard-2
database cluster binding shard shard-2 shard-server server-b storage-engine MMAPv1 address 182.22.31.14 port 27018 priority 10
database cluster binding shard shard-2 shard-server server-a storage-engine MMAPv1 address 182.22.31.13 port 27018 priority 5
database cluster binding shard shard-2 shard-server arbiter-2 arbiter true storage-engine MMAPv1 address 182.22.31.11 port 27018
database cluster binding shard shard-2 shard-server-seed server-b

database cluster binding shard shard-1 zone-name mumbai
database cluster binding shard shard-2 zone-name pune
```

a) Check configured ranges.

Example:

```
use ipv6ShardDB
switched to db ipv6ShardDB
binding-sharddb:PRIMARY>
binding-sharddb:PRIMARY> db.shards.find()
```

```
binding-sharddb:PRIMARY> db.shards.find()
{ "_id" : 1, "name" : "shard-1", "hosts" : "182.22.31.13:27017,182.22.31.14:27017", "zone" : "mumbai" }
{ "_id" : 2, "name" : "shard-2", "hosts" : "182.22.31.13:27018,182.22.31.14:27018", "zone" : "pune" }
{ "_id" : 3, "name" : "shard-3", "hosts" : "182.22.31.13:27020,182.22.31.14:27020", "zone" : "hyd" }
{ "_id" : 4, "name" : "shard-4", "hosts" : "182.22.31.13:27021,182.22.31.14:27021", "zone" : "bglr" }
{ "_id" : 5, "name" : "shard-5", "hosts" : "182.22.31.13:27022,182.22.31.14:27022", "zone" : "chennai" }
{ "_id" : 6, "name" : "shard-6", "hosts" : "182.22.31.13:27023,182.22.31.14:27023", "zone" : "hyd" }
{ "_id" : 7, "name" : "shard-7", "hosts" : "182.22.31.13:27024,182.22.31.14:27024", "zone" : "bglr" }
{ "_id" : 8, "name" : "shard-8", "hosts" : "182.22.31.13:27025,182.22.31.14:27025", "zone" : "pune" }
binding-sharddb:PRIMARY>
```

b) Checking configured zones.

Example:

```
binding-sharddb:PRIMARY> db.zoneinfo.find()
{ "_id" : 1, "name" : "r1", "start" : "2017:6000:0000:0001", "end" : "2017:6000:0000:0500", "zone" : "bglr" }
{ "_id" : 2, "name" : "r2", "start" : "2018:6000:0000:0001", "end" : "2018:6000:0000:0500", "zone" : "bglr" }
{ "_id" : 3, "name" : "r1", "start" : "2013:6000:0000:0001", "end" : "2013:6000:0000:0500", "zone" : "chennai" }
{ "_id" : 4, "name" : "r2", "start" : "2014:6000:0000:0001", "end" : "2014:6000:0000:0500", "zone" : "chennai" }
{ "_id" : 5, "name" : "r1", "start" : "2015:6000:0000:0001", "end" : "2015:6000:0000:0500", "zone" : "hyd" }
{ "_id" : 6, "name" : "r2", "start" : "2016:6000:0000:0001", "end" : "2016:6000:0000:0500", "zone" : "hyd" }
{ "_id" : 7, "name" : "r1", "start" : "2008:5000:0000:0100", "end" : "2008:5000:0000:0500", "zone" : "mumbai" }
{ "_id" : 8, "name" : "r2", "start" : "2009:5000:0000:0100", "end" : "2009:5000:0000:0500", "zone" : "mumbai" }
{ "_id" : 9, "name" : "r1", "start" : "2011:6000:0000:0001", "end" : "2011:6000:0000:0500", "zone" : "pune" }
{ "_id" : 10, "name" : "r2", "start" : "2012:6000:0000:0001", "end" : "2012:6000:0000:0500", "zone" : "pune" }
```

c) Checking shard/range mapping.

Example:

```
binding-sharddb:PRIMARY> db.buckets.find()
{ "_id" : ObjectId("5cb9845c63ebec62ea803d42"), "bucket-id" : 1, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d43"), "bucket-id" : 2, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d44"), "bucket-id" : 3, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d45"), "bucket-id" : 4, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d46"), "bucket-id" : 5, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d47"), "bucket-id" : 6, "shard" : 4, "migration" : false, "zone" : "bglr" }
{ "_id" : ObjectId("5cb9845c63ebec62ea803d48"), "bucket-id" : 7, "shard" : 4, "migration" : false, "zone" : "bglr" }
```

Modifying Zone Aware Sharding

Note This section is applicable when you want to add/update/delete zones and ranges after performing initial database configuration.

If you have modified already configured zones and ranges in database, you need to perform the following steps to have sharding database metadata updated with new modified configuration commits.

Procedure

Step 1 Connect to session database cluster sharding primary database for which configuration changes have been committed.

a) Use `show database status` command to get the sharding database primary IP address and port number of a particular database cluster.

b) Use `docker exec -it orchestrator bash` command to connect to sharding database.

c) Connect to sharding database.

If MongoDB authentication is enabled, execute `mongodb --ipv6 mongodb://adminuser:`*`password`*`@[`*`IPAddress`*`]:`*`Port`*`/admin` command.

OR

If MongoDB authentication is disabled, execute `mongodb --ipv6 mongodb://[`*`IPAddress`*`]:`*`Port` command.

d) Verify that after completing 1.c, on page 40, the prompt is redirected to mongo shell of the same.

Step 2 Remove the zone collection information using the following commands from inside mongo shell:

```
use ipv6ShardDB
db.zoneinfo.remove({})
```

Step 3 Wait for 30 seconds and verify that zone collection information is updated with the new committed configurations.

```
db.zoneinfo.find()
exit
```