

Diameter Endpoint

- Feature Summary and Revision History, on page 1
- Feature Description, on page 2
- How it Works, on page 7
- Configuring Diameter Endpoint, on page 7
- Dynamic Configuration Change, on page 18
- Destination Host AVP and Destination Realm AVP for CCR, on page 19
- Peer Management, on page 21
- Failure Handling, on page 27
- OAM Support, on page 32
- Troubleshooting Information, on page 42

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Products or Functional Area	SMF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Always-on
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
Added the following support: • Gx and Gy custom dictionary support for RAR messages • DWR and DWA handling • Bounded peer	2023.04.0
First introduced.	2023.03.0

Feature Description

The Diameter Endpoint is an App-infra-based service that enables the Diameter functionality for the SMF service. In SMF, Diameter Endpoint implements the Gx and Gy clients. The Diameter endpoint leverages the Ops Center for configuration, operational, and action CLI commands and App-infra features, such as interpod communications, RPCs, Monitor Subscriber, Monitor Protocol, logging, and statistics.

The Diameter Endpoint provides the following support:

- Diameter peer and connection management for both the Gx and Gy interfaces.
- Failure handling templates for the configuration of failure handling and retry behavior for the Diameter messages.
- Ability to peer to multiple Diameter routing agents (DRA) and route the Diameter messages to remote Diameter endpoints through the DRAs.

Note The Diameter stack doesn't implement the ability to act as a DRA.

- Monitor Protocol and Monitor Subscriber (SUPI and IMSI subscriber key) for Diameter messages.
- Logging and statistics.

To use a Diameter endpoint, deploy a pair of Diameter pods with an Active or Standby configuration for each interface, which is Gx and Gy. Deploy these pods in the host networking configuration with a pair of internal and external VIPs.

For online charging, the Diameter Credit Control application is used. The online client, Charging Transfer Function (CTF), requests resource allocation and reports credit control information to the Online Charging System (OCS). To support Credit Control through Diameter, the CCR (Credit Control Request) and the CCA (Credit Control Answer) messages are used.

Service pod sends the gRPC messages to the Diameter endpoint for Credit Control messages. The Diameter endpoint converts the Diameter CCR messages and sends them to Gx and Gy servers, which are PCRF or

OCS. After receiving the CCA, the CCA message is converted to the gRPC message and sent to the SMF as a response.

Supported Interfaces

SMF supports the following interfaces:

- Gx interface
- Gy interface

For the features related to the Gx and Gy interfaces, see the Interfaces Support chapter.



Important It should not be assumed that the Diameter interface features available on the SMF have full feature parity and functionality with StarOS or CUPS product. Furthermore, it should not be assumed that any constructs (including, but not limited to, commands, statistics, attributes, MIB objects, alarms, logs, services) referenced in this document imply functional parity with StarOS legacy or CUPS products.

Please contact your Cisco Account or Support representative for any questions about parity between this product and any StarOS legacy or CUPS products.

Diameter Endpoint Instance

A Diameter endpoint instance is used to configure the Diameter endpoints for different interfaces, such as Gx and Gy. This configuration determines the number of Diameter endpoint pods to be deployed. Configuring a Diameter endpoint instance is mandatory.

Diameter Client Profile

A Diameter client is a Diameter node that supports Diameter client applications and the base protocol. A Diameter client is implemented in the devices that are placed at the edge of a network and provides access control services for that network.

A Diameter profile provides network access information for the Diameter application. Each Diameter client profile is associated with an existing endpoint profile and with a failure handling template by its name. SMF uses the client profile name to allow the Diameter endpoint fetch the associated endpoint profile. SMF uses this profile to evaluate the matching endpoint to send the messages.

For encoding Credit Control Requests (CCR) and decoding of the corresponding responses, Credit Control Answer (CCA) messages, you have to configure a dictionary in the Diameter client profile. A dictionary includes the list of AVPs with their descriptions that are used in the CCR and CCA messages, with the standard AVPs or vendor-specific AVPs.

Diameter Endpoint Profile

A Diameter endpoint profile provides the following information:

- Origin configuration, such as the origin realm, origin host name, origin state ID, and origin address.

For the origin state ID, the SMF provides the Diameter endpoint-originated OSI support for the Credit Control Request (CCR) or Device Watchdog Request (DWR) messages. You can enable or disable the dynamic OSI value through Ops Center using the **dynamic-origin-state-id** CLI command. The Nodemgr updates the OSI value in the cache pod. Then, the Diameter endpoint uses this value to send the Capability Exchange Request (CER), CCR, and DWR messages.

- Peer configuration, such as the peer realm, destination host name, peer IP, and peer port.
- Global parameters, such as destination host AVP, VSA support, and maximum outstanding messages.
- Timeout duration.

You can configure the Diameter endpoint profiles for each interface. These profiles are provisioned to the Diameter endpoint pods corresponding to the interface. Then, these profiles are associated with a specific Diameter client profile.



Note Configure the Diameter endpoint profile for the Diameter pods deployment.

Disconnect Peer Request Management

This section describes about the Disconnect Peer Request (DPR), Disconnect Peer Answer (DPA), and its related CLI commands to configure the DPR and DPA in the Diameter endpoint.

Feature Description

The SMF supports the DPR and it handles the received DPR from server. The DPR is sent to a peer to inform its intentions to shut down the transport connection. When the peer node sends the DPR to another node, the node validates the DPR for all required AVPs. If the validation isn't successful, the node sends the DPA with the error to the peer node.

If the validation is successful:

- The SMF node waits till the drain time before it sends any DPA to the peer node. DrainTime is the time interval between receiving a DPR by diameter endpoint and sending a DPA response. No new requests are initiated by the diameter-ep to the peer during this interval. However any messages received from the peer are processed during this interval. Drain time is a configurable parameter. It's recommended to configure the drain time value less than the timeout value for inbound messages. It's advised that no new request to be sent to that peer during this time, but in case any requests comes from the peer, it must be responded.
- Regardless of whether any messages have previously been forwarded to the client and are waiting for a response, transmit DPA successfully. The peer node sends the DPA with success.

If a server wants to disconnect itself from a peer SMF node, it sends DPR with disconnect cause AVP.

- If disconnect cause is REBOOTING, defer connecting back based on Tc Timer. Tc timer is a configurable parameter.
  - The Reboot time is the time duration after which connection is reattempted to peer from which DPR is received with disconnect cause "REBOOTING". This interval starts after sending the DPA response and connection teardown.

- If the disconnect cause is BUSY or DO_NOT_WANT_TO_TALK_TO_YOU, defer connecting back for longer duration. Duration is a configurable parameter. As it's recommended not to connect back in case of BUSY or DO_NOT_WANT_TO_TALK_TO_YOU. So, you can configure negative or zero as value to never connect back.
  - DoNotTalkTime is the time duration after which connection is reattempted to peer from which DPR is received with disconnect cause "DO_NOT_WANT_TO_TALK_TO_YOU". This interval starts after sending the DPA response and connection teardown.
  - BusyTime is the time duration after which the connection is reattempted to peer from which DPR is received with disconnect cause "BUSY". This interval starts after sending the DPA response and connection teardown.
- The time duration for reconnecting back is applicable across all pod restarts or pod failover but doesn't apply to site failover.

DWR and DWA Handling

Feature Description

Diameter endpoint handles the incoming Device Watchdog Request (DWR) from a peer and responds with a Device Watchdog Answer (DWA) with the success result code. The Diameter endpoint also supports detection of the peer having an issue or transport failure on an idle connection by sending the DWR messages.

How it Works

To send a DWR to a peer, the Diameter endpoint uses the app-infra provided host monitoring to check if DWR needs to be generated.

On each host monitoring check, the Diameter endpoint checks for any message exchanges happened between the last check and now. When there's both a request and response for a message, then Diameter endpoint considers it a message exchange. A message exchange isn't considered in the following scenarios:

- If the Diameter endpoint sends a request but receives no response from a peer.
- If the Diameter endpoint received the request but failed to send the response to the peer.

If no message exchange happens between the Diameter endpoint and peer, then Diameter endpoint generates a DWR toward peer. The Diameter endpoint considers the following scenarios as the DWR failure:

- If sending the DWR fails.
- If no DWA is received in the time configured in the **basemsg retransmission-timeout** parameter.
- If there's a response but with a failure result code.

In the case of DWR failure, a retry occurs if the **basemsg retransmissions** parameter value is configured to a nonzero value. The retry is performed in the next host monitoring check. During retries, if the Diameter endpoints received a DWA for any of the earlier sent DWR, then the Diameter endpoint ignores it and doesn't consider it toward the successful DWR.

Even after the configured retries, if the DWR isn't successful then the status is reported as a failure, and the connection is reset. After the connection is reset only during the next host monitoring check, the connection is reestablished. Then, the Capability Exchange Request (CER) and DWR are sent. However, the Diameter

endpoint sends no messages to the peer if DWR or DWA isn't successful. In the case of DWR or DWA failure, the connection is reset.

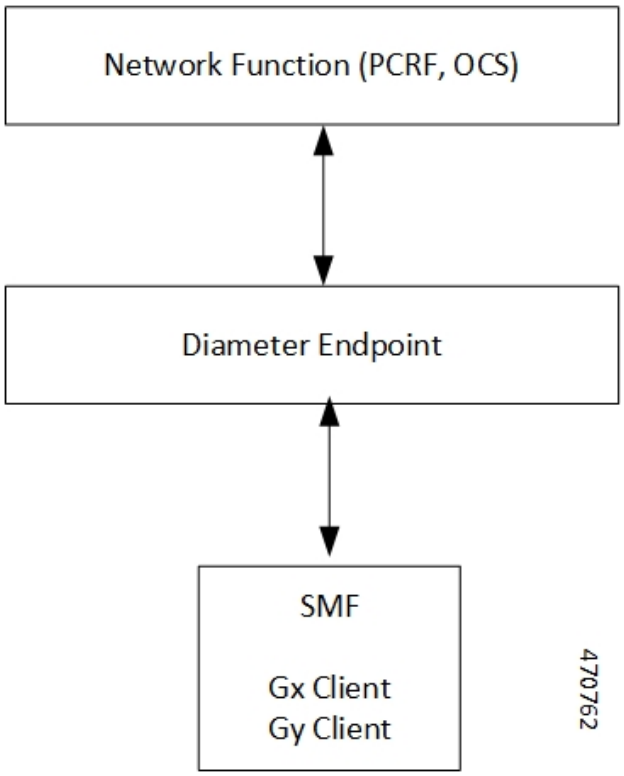
Note While the host monitoring for all hosts is performed in parallel, the monitoring completes after it's completed for all the hosts. Delay in monitoring for one host leads to delay in next monitoring for all the hosts.

Architecture

Diameter common stack, which is the Diameter endpoint, is an interface between the NFs and both client and server modes for a supported application.

The following diagram shows the architecture of the common Diameter stack.

Figure 1: Common Diameter stack



The common Diameter stack includes the following components:

- Network Functions—Policy and Charging Rules Function (PCRF) and Online Charging System (OCS)
- Diameter Endpoint
- Diameter interfaces on SMF—Gx and Gy clients

For more information on the supported Diameter interfaces and their related features, see the *Interfaces Support* chapter.

How it Works

The Diameter endpoint uses the App-infra framework for sending or receiving the Diameter messages to the Diameter peers and internal IPC messages to other pods. For example, P-GW service, cnBNG service. In SMF, the Diameter endpoint implements the Gx and Gy clients.

To support Credit Control through the Diameter, the two Diameter messages—Credit Control Request (CCR) and the Credit Control Answer (CCA) are used. A service pod sends the gRPC messages to the Diameter endpoint for conversion of the gRPC messages to the Credit Control messages. Then, the Diameter endpoint sends the CCR messages to the Gx or Gy servers, which are PCRF or OCS. After the Gx or Gy server sends the CCA message, the Diameter endpoint converts the CCA message to the gRPC message and sends it to the P-GW service as a response.

The Diameter supports the standard AVPs, as per the *3GPP 29.212 version 15.3.0* and *3GPP 32.299 version 12.9.0* specifications for the Gx and Gy interfaces.

Standards Compliance

The Diameter endpoint complies with the following standards:

- *3GPP 29.212 version 15.3.0*
- *3GPP 32.299 version 12.9.0*

Limitations

The Diameter endpoint has the following limitation:

- The Diameter endpoint supports only one pair of pods for each interface.

Configuring Diameter Endpoint

The configuration of the Diameter endpoint includes the following steps:

1. Configuring Diameter Endpoint Instances, on page 7
2. Configuring Diameter Client Profile, on page 9
3. Configuring Diameter Profile Endpoint on the Gx and Gy Interfaces, on page 10

Configuring Diameter Endpoint Instances

To configure the Diameter endpoint instances, use the following sample configuration:

```
config
  instance instance-id instance-id
    endpoint diameter
    interface interface_name
      internal-vip ip_address
```

```
    vip-ip ip_address
    mode mode_value
  exit
```

NOTES:

- **instance instance-id** *instance-id* **endpoint diameter** : Specify the number of Diameter endpoint pods to be deployed.
- **interface** *interface_name*: Specify the interface, such as Gx and Gy, for the Diameter endpoints.
- **internal-vip** *ip_address*: Specify the internal VIP IP address for the Diameter endpoint to listen for requests from the service. **internal-vip** is a mandatory parameter.
- **vip-ip** *ip_address*: Specify the VIP IPv4 address for the corresponding to the Diameter clients. **vip-ip** is a mandatory parameter.
- **mode** *mode_value*: Specify the Diameter endpoints with a specific mode. **mode** is a mandatory parameter.

Note SMF supports only the *client* as *mode_value*.

Configuration Example

The following is an example configuration of Diameter endpoint instances.

```
instance instance-id 1
endpoint diameter
  interface gy
   internal-vip 209.165.200.234
   vip-ip 209.165.200.234
   mode client
   exit
  exit
  interface gx
   internal-vip 209.165.200.234
   vip-ip 209.165.200.234
   mode client
  exit
```

Configuration Verification

To verify if the Diameter endpoint instances are configured, use the following **show running-config instance instance-id** *instance_id* **endpoint diameter** command.

```
smf# show running-config instance instance-id 1 endpoint diameter
instance instance-id 1
endpoint diameter
  interface gy
   internal-vip 209.165.200.234
   vip-ip 209.165.200.234
   mode client
   exit
  exit
  interface gx
   internal-vip 209.165.200.234
   vip-ip 209.165.200.234
```

```
 mode client
exit
```

Configuring Diameter Client Profile

Table 3: Feature History

Feature Name	Release Information	Description
Same dictionary support extended to RAR messages.	2023.04	In the previous releases, for CCR-I, CCA-U, and CCA-T messages, the dictionary specified under the client, which is a custom dictionary, was applied. For RAR message, a common dictionary was being used. With this release, RAR messages also use the same dictionary that CCR-I, CCA-U, and CCA-T messages use.

To configure the Diameter client profile, use the following sample configuration:

```
config
  profile diameter-client diameter_client_name
    endpoint endpoint_profile_name
    failure-handling-profile profile_name
    dictionary-name { dcca-custom8 | default | r8-gx-standard }
    end
```

NOTES:

- **profile diameter-client** *diameter_client_name*: Specify a Diameter client profile name.
- **endpoint** *endpoint_profile_name*: Specify an existing endpoint profile name to associate it with a Diameter client profile. You can define a maximum of 100 client profiles.
- **failure-handling-profile** *profile_name*: Specify an existing failure handling profile to associate it with a Diameter client profile.
- **dictionary-name { dcca-custom8 | default | r8-gx-standard }**: Specify one of the following dictionaries:
  - **dcca-custom8**: This dictionary is the standard Gy dictionary.
  - **default**: This dictionary is the default dictionary.
  - **r8-gx-standard**: This dictionary is the standard Gx dictionary.

Note The specified dictionary is used for Credit Control - Initial (CCR-I), Credit Control - Update (CCR-U), Credit Control - Terminate (CCR-T), and Re-Authorization Request (RAR) messages.

Configuration Example

The following is an example configuration of the Diameter client profile.

```
profile diameter-client gydc01
    endpoint gxProf1
    failure-handling-profile fh1
    exit
```

Configuration Verification

To verify if the Diameter client profile is configured, use the following **show running-config profile diameter-client** command:

```
[smf] smf# show running-config profile diameter-client
profile diameter-client gydc01 endpoint gxProf1 failure-handling-profile fh1 dictionary-name
 kt-gy
exit
profile diameter-client gydc11 endpoint gyProf1 failure-handling-profile fh2
exit
profile diameter-client gydc12 endpoint gyProf2 failure-handling-profile fh3
exit
profile diameter-client gxdc02 endpoint gxProf1 failure-handling-profile fh4
exit
```

Configuring Diameter Profile Endpoint on the Gx and Gy Interfaces

Table 4: Feature History

Feature Name	Release Information	Description
VRF support for the Diameter Gx and Gy interfaces	2023.04	SMF supports the client-side VRF for the Diameter Gx and Gy interfaces. The profile Diameter endpoint refers to the VRF name and interface name that you configure from the Diameter endpoint. The Diameter endpoint uses the VRF name to create a TCP connection from the Diameter client.

To configure the Diameter profile endpoint on the Gx and Gy interfaces, use the following sample configuration:

```
config
  profile diameter-endpoint interface_name
    name profile_name
    internal-vip ip_address
      destination-host-avp message_type
      vsa-support vendorId-source
      max-outstanding number_of_messages
      response-timeout response_timeout_value
      connection-timeout connection-timeout_value
      basemsg retransmission-timeout retransmission_timeout_value
      basemsg retransmissions max_retry_value
```

```
        basemsg watchdog-interval interval_value
        dscp [ dscp_value | af11 | af12 | af13 | af21 | af22 | af23 | af31
| af32 | af33 | af41 | af42 | af43 | be | cs1 | cs2 | cs3 | cs4 | cs5 |
cs6 | cs7 | ef ]
        origin realm realm_name
        origin host host_name address ipv4 ip_address
        origin peer origin_peer_name
        realm  realm_name
        address ipv4 ip_address
        address vrf vrf_name
        port peer_port
        destination-host-name  destination_host_name
        load-balancing-algorithm highest-weight
        route-entry host [ host-name | * ] realm [ realm-name | * ] peer
peer_name weight weight_value
        route-failure deadtime deadtime_value result-code result_code_value
threshold  number
        route-failure result-code result_codes_value
        route-failure threshold  threshold_number
        route-failure recovery-threshold percent  recovery_threshold_percentage

        dynamic-route expiry-timeout  expiry_timeout_value
        dynamic-origin-state-id  boolean_value
        exit
```

NOTES:

- **profile diameter-endpoint** *interface_name*: Specify a Diameter profile for the Gx or Gy interface.
- **name** *profile_name*: Specify the profile name of the Diameter profile interface. Each profile configures the Diameter source information and the peer information for the Diameter messages that go toward those peers.

Note You can define a maximum of 100 profiles per interface.

- **internal-vip** *ip_address*: Specify an internal VIP IP address for communication with service pods. **internal-vip** is a mandatory parameter.
- **destination-host-avp** *message_type*: Specify the type of message in which the destination host AVP is to be encoded.

Note SMF supports only the *always* value for the *message_type*.

- **vsa-support** *vendorId-source*: Specify the source of vendor IDs DIABASE to be used for negotiation of Diameter peer capabilities.

Note SMF supports only the *all-from-dictionary* value for the *vendorId-source*.

- **max-outstanding** *number_of_messages*: Specify the maximum number of Diameter messages to be sent to any peer in the profile, while awaiting the responses. The default value is 256. *number_of_messages* must be in the range of 1–4096.
- **response-timeout** *response_timeout_value*: Specify the maximum allowed response time for request messages that the Diameter applications send to the Diameter server. The default value is 60. *response_timeout_value* must be in the range of 1–300.
- **connection-timeout** *connection-timeout_value*: Specify the maximum allowed time for establishing the transport layer connectivity, such as the TCP connection, toward the Diameter server. The default value is 30. *connection-timeout_value* must be in the range of 1–300.
- **basemsg retransmission-timeout** *retransmission_timeout_value*: Specify the timeout value between retransmissions of the base messages, such as Device Watchdog Request (DWR) and Capability Exchange Request (CER), toward the Diameter server. The default value is 30. *retransmission_timeout_value* must be in the range of 1–120.
- **basemsg retransmissions** *max-retries*: Specify the maximum number of times the base messages must be retransmitted. The default value is 1. *max-retries* must be in the range of 1–10.
- **basemsg watchdog-interval** *interval_value*: Specify the minimum time interval between the two DWR messages that are sent toward the Diameter server. The default value is 5. *interval_value* must be in the range of 5–30.
- **dscp [** *dscp_value* **| af11 | af12 | af13 | af21 | af22 | af23 | af31 | af32 | af33 | af41 | af42 | af43 | be | cs1 | cs2 | cs3 | cs4 | cs5 | cs6 | cs7 | ef]** : Specify the Differential Services Code Point (DSCP) value in the IP header of the Diameter messages that are sent toward the Diameter server. The default value is **be**. *dscp_value* must be in the range of 0–63. Choose in the following other DSCP values, as required:
  - **af**xx: Specify this value for the use of an assured forwarding xx per hop behavior (PHB).
  - **be**: Specify this value for the use of the best effort forwarding PHB. **be** is the default value.
  - **cs**x: Specify this value for the use of class selector x per PHB.
  - **ef**: Specify this value for the use of expedited forwarding PHB.
- **origin realm** *realm_name*: Specify the name of the realm for the Diameter endpoint. This parameter is mandatory.
- **origin host** *host_name* **address ipv4** *ip_address* : Specify the host name, which is the FQDN of the Diameter endpoint. Specify the IPv4 address, which is the Diameter endpoint Bind IP address for the Diameter client connections.
- **origin peer** *origin_peer_name* : Specify the identifier for a Diameter peer. This parameter is mandatory.

Note You can configure a maximum of 5000 peers.

- **realm** *realm_name*: Specify the name of the realm for a peer with the name of the peer. This parameter is mandatory.
- **address ipv4** *ip_address*: Specify the IP address of the Diameter peer.
- **address vrf** *vrf_name*: Specify the VRF name that is to be referenced from the Diameter endpoint.
- **port** *peer_port*: Specify the port of the Diameter peer. This parameter is mandatory.
- **destination-host-name** *destination_host_name*: Specify the custom destination host name to be used in destination host AVP. This parameter is optional.
- **load-balancing-algorithm highest-weight** : Choose an idle server with the highest weight in failure scenarios. If multiple servers have the same high weight, then the load balancing happens among those servers.
- **route-entry host [host-name | *] realm [realm-name | *] peer** *peer_name* **weight** *weight_value* : Use this command to configure two static entries, such as a peer in the route table. If you configure an entry with the existing same flag, host, realm, then only the weight is updated with the higher of the two of them. The **host** and **realm** parameters allow wildcard character values. The **weight** is an optional parameter with the default value as 10. The **peer** is a mandatory parameter.



Note You can configure multiple route entries with the same host and realm but a different peer without being overridden.

- **route-failure deadtime** *deadtime_value* **result-code** *result_code_value* **threshold** *number*: Specify the duration in seconds for which the system keeps the route in the **FAILED** status. After the configured duration expires, the system changes the status to **AVAILABLE**. *deadtime_value* must be an integer in the range of 1–86400. The default value is 60.
- **route-failure recovery-threshold percent** *recovery_threshold_percentage*: Specify the percentage value at which the failure counter is reset when provisionally changing the status from **FAILED** to **AVAILABLE**. For example, a failure counter of 16 caused the **AVAILABLE** status to change to **FAILED** status. After the configured deadtime expires, the status changes to **AVAILABLE**. If you configured *recovery_threshold_percentage* with 75 percent, the failure counter resets to 12, which is 75 percent of 16. The default value is 90.
- **dynamic-route expiry-timeout** *expiry_timeout_value*: Specify the expiration time for dynamic routes that you created after reaching the Diameter destination host. The default value is 86400 secs, which equals one day.
- **dynamic-origin-state-id** *boolean_value* : Specify whether you want to enable or disable the dynamic origin state ID. The default value is true.



Note After the configuration changes from dynamic to static, if the peer is started with either the dynamic origin state ID or the static origin state ID, the peer sends the default static value. Similarly, for a configuration change from the dynamic to static, the peer sends the dynamic value that the Diameter endpoint fetches from the cache pod. In this case, there's a reconnection.

Configuration Example

The following is an example configuration of the Diameter endpoint on the Gy interface.

```
profile diameter-endpoint gy
 internal-vip 209.165.200.234
 name gyProf1
  destination-host-avp always
  vsa-support all-from-dictionary
  max-outstanding  128
  response-timeout 80
  connection-timeout 20
  dscp af11
  basemsg retransmission-timeout 20
  basemsg retransmissions 2
  basemsg watchdog-interval 25
  origin realm gy-cisco
  origin host gy.cisco.com
   address ipv4 209.165.200.229
  exit
  peer gyPeer1
   realm  GyServer1.com
   destination-host-name dest.host2.com
   address ipv4 209.165.200.230
   port 3900
  exit
  peer gyPeer2
   realm GyServer2.com
   address ipv4 209.165.200.231
   port 3901
  exit
exit
```

The following is an example configuration of the Diameter endpoint that is associated with the VRF.

```
profile diameter-endpoint gx
instance instance-id 1
  name gxProf1
   destination-host-avp always
   vsa-support all-from-dictionary
   max-outstanding 128
   response-timeout 20
   connection-timeout 5
   dscp be
   basemsg retransmission-timeout 2
   basemsg retransmissions 10
   basemsg watchdog-interval 6
   origin realm gx-cisco
   origin host gx.cisco.com
    address ipv4 209.165.200.230
    address vrf vrf_signaling
   exit
   peer gxPeer1
    realm dummyDestRealm.com
    address ipv4 209.165.200.231
    port  3868
   exit
   peer gxTest
    realm xyznetworks.com
    address ipv4 209.165.200.232
    port  4868
   exit
  exit
 exit
exit
```

Configuration Verification

To verify if the Diameter endpoint on the Gx or Gy interface is configured, use the following **show running-config profile diameter-endpoint** command:

```
[smf] smf# show running-config profile diameter-endpoint
 profile diameter-endpoint gx
 internal-vip 209.165.200.234
 name gxProf1
  destination-host-avp always
  vsa-support all-from-dictionary
  origin realm gx-cisco
  origin host gx.cisco.com
   address ipv4 209.165.200.235
  exit
  peer gxPeer1
   realm  GxServer1.com
   destination-host-name dest.host1.com
   address ipv4 ipv4 209.165.200.230
   port 3870
  exit
  peer gxPeer2
   realm GxServer2.com
   address ipv4 10.84.27.161
  exit
 exit

profile diameter-endpoint gy
 internal-vip 209.165.200.234
 name gyProf1
  destination-host-avp always
  vsa-support all-from-dictionary
  max-outstanding  128
  response-timeout 80
  connection-timeout 20
  dscp af11
  basemsg retransmission-timeout 20
  basemsg retransmissions 2
  basemsg watchdog-interval 25
  origin realm gy-cisco
  origin host gy.cisco.com
   address ipv4 209.165.200.229
  exit
  peer gyPeer1
   realm  GyServer1.com
   destination-host-name dest.host2.com
   address ipv4 209.165.200.230
   port 3900
  exit
  peer gyPeer2
   realm GyServer2.com
   address ipv4 209.165.200.231
   port 3901
  exit
exit
```

To verify if the Diameter endpoint is associated with a VRF, use the following **show running-config profile diameter-endpoint gx** command:

```
profile diameter-endpoint gx
instance instance-id 1
  name gxProf1
   destination-host-avp always
   vsa-support all-from-dictionary
```

```
   max-outstanding 128
   response-timeout 20
   connection-timeout 5
   dscp be
   basemsg retransmission-timeout 2
   basemsg retransmissions 10
   basemsg watchdog-interval 6
   origin realm gx-cisco
   origin host gx.cisco.com
    address ipv4 209.165.200.230
   address vrf vrf_signaling
   exit
   peer gxPeer1
    realm dummyDestRealm.com
    address ipv4 209.165.200.231
    port  3868
   exit
   peer gxTest
    realm xyznetworks.com
    address ipv4 209.165.200.232
    port  4868
   exit
  exit
 exit
exit
```

Configuring Busy Time for DPR

To configure the busy time for DPR in diameter ep, use the following configuration:

```
config
   profile diameter-client diameter_client_name
      endpoint endpoint-profile-name
         peer peer_name
         disconnect peer request
            busy time time_duration
            end
```

NOTES:

- **profile diameter-client** *diameter_client_name*: Specify a Diameter client profile name.
- **endpoint** *endpoint-profile-name*: Specify an existing endpoint profile name to associate it with a Diameter client profile.
- **peer** *peer_name*: Specify the name of the peer node.
- **busy time** *time_duration*: Specify the time duration after which the connection is reattempted to peer. The time duration value must be an integer in the range of 1–300 seconds. The default value is 3 seconds.

Configuring Do Not Talk Time for DPR

To configure the do not talk time for DPR in diameter ep, use the following configuration:

```
config
   profile diameter-client diameter_client_name
      endpoint endpoint-profile-name
         peer peer_name
```

```
disconnect peer request
   do not talk time time_interval
   end
```

NOTES:

- **profile diameter-client** *diameter_client_name*: Specify a Diameter client profile name.
- **endpoint** *endpoint-profile-name*: Specify an existing endpoint profile name to associate it with a Diameter client profile.
- **peer** *peer_name*: Specify the name of the peer node.
- **do not talk time** *time_interval*: Specify the time interval between receiving of DPR by diameter endpoint and sending of DPA response. The time interval value must be the integer in the range of 1–300 seconds. The default value is 3 seconds.

Configuring Drain Time for DPR

To configure the drain time for DPR in diameter ep, use the following configuration:

```
config
  profile diameter-client diameter_client_name
    endpoint endpoint-profile-name
      peer peer_name
      disconnect peer request
        drain time time_interval
        end
```

NOTES:

- **profile diameter-client** *diameter_client_name*: Specify a Diameter client profile name.
- **endpoint** *endpoint-profile-name*: Specify an existing endpoint profile name to associate it with a Diameter client profile.
- **peer** *peer_name*: Specify the name of the peer node.
- **drain time** *time_interval*: Specify the time interval between receiving of DPR by diameter endpoint and sending of DPA response. The time interval value must be an integer in the range of 1 to 10 seconds. The default value is 3 seconds.

Configuring Reboot Time for DPR

To configure the reboot time for DPR in diameter ep, use the following configuration:

```
config
  profile diameter-client diameter_client_name
    endpoint endpoint-profile-name
      peer peer_name
      disconnect peer request
        reboot time time_duration
        end
```

NOTES:

- **profile diameter-client** *diameter_client_name*: Specify a Diameter client profile name.
- **endpoint** *endpoint-profile-name*: Specify an existing endpoint profile name to associate it with a Diameter client profile.
- **peer** *peer_name*: Specify the name of the peer node.
- **reboot time** *time_duration*: Specify the time duration after which the connection is reattempted to peer. The time duration value must be an integer in the range of 1–300 seconds. The default value is 3 seconds.

Dynamic Configuration Change

Feature Description

The SMF allows you to change the Diameter endpoint configuration dynamically without impacting the existing calls.

How it Works

The following table lists the impact of dynamic update to the various Diameter endpoint configurations:

Table 5: Dynamic Update of Diameter Endpoint Configuration

Configuration	Dynamic Change	Impact on Existing Sessions
Connection attributes: • BaseMsg attributes, such as retransmissionTimeout, retransmissions • connectionTimeout • responseTimeout • originStateId • Origin attributes, such as Host Address, Host Name, and Realm • Peer attributes, such as address, port, and realm	Allowed	Change in connection attributes has the following impact: • RPC is deregistered and registered with new values for all peers in the endpoint profile. • All the messages initiated on the peer fail when the connection is down. • All the requests in transit on the peer are dropped. Server can't send request messages (RAR) and response messages (CCA) on these connections for existing calls.

Configuration	Dynamic Change	Impact on Existing Sessions
DisconnectPeerRequest Properties: • drainTime • rebootTime • doNotTalkTime • busyTime	Not allowed	When the DisconnectPeerRequest Properties are updated for a peer, the new values are effective only after the pod restart or after the connection is reset.
Route-entry update or addition	Allowed	An update or addition to route entry has the following impact: • Route entries are recreated after change. • After a peer is deregistered, all route entries for the peer are deleted.
Diameter-endpoint profile removal	Allowed	The removal of an endpoint profile has the following impact: • All the peers of the endpoint profile are deregistered. • Existing calls using these peers or routes fail. • No new calls are initiated on those peers.

Destination Host AVP and Destination Realm AVP for CCR

Feature Description

The destination-host AVP is used for forwarding or routing a Diameter request message. The destination host name that an application, such as Gx and Gy, uses is identified from the incoming application-level messages. The absence of the destination-host AVP causes a message to be sent to any Diameter server that supports the application within the realm specified in destination-realm AVP.

Based on the end user configuration, you can include the destination-host AVP for the CCR and the retried messages.

Destination-Host AVP is included or excluded in CCR message based on the given criteria:

- If destination-host-avp is set as **always**, the destination-host AVP in CCR-I/U/T (even in retried messages) is included.
- If destination-host-avp is set as **session-binding**, the destination-host only CCR-U/T message that are not being retried or going to secondary peer is included.

Destination-Realm AVP is set as:

- Realm of peer for CCR-I message and for CCR-U/T in non-DRA case.
- Origin-Realm received in previous CCA-I/U in DRA case.

Configuring Destination Host AVP

To configure the destination host AVP, use the following sample configuration:

```
config
  profile diameter-endpoint interface_name
    instance instance-id instance-id
      name profile_name
        destination-host-avp [ always | session-binding ]
  exit
```

NOTES:

- **profile diameter-endpoint** *interface_name*: Specify a Diameter endpoint profile, which is either Gx or Gy interface.
- **instance instance-id** *instance-id* : Specify the GR instance.
- **name** *profile_name*: Specify the Diameter endpoint profile name.

Note You can define a maximum of 100 profiles per interface.

- **destination-host-avp [always | session-binding]**: Specify one of the following types of message in which the destination host AVP is to be encoded:
  - **always**—If destination-host-avp is configured to this message type, include the destination-host-avp in both the CCR-I, CCR-U, or CCR-T messages and the retried messages. **always** is the default message type.
  - **session-binding**—If destination-host-avp is configured to this message type, encode the destination-host-avp in CCR-U and CCR-T messages, and not CCR-I and retried messages.

Configuration Example

The following is an example configuration of the destination host AVP.

```
profile diameter-endpoint gx
   instance instance-id 1
      name gxProf1
         destination-host-avp session-binding
exit
```

Peer Management

Feature Description

The SMF supports peer management in routing. It allows to choose an appropriate peer for a transaction to occur.

The peer selection logic depends on the Diameter Endpoint and Host Selection configuration.

The Diameter servers connect to the gateway in direct and indirect ways through DRA. The entities involved in the peer management are as follows:

1. Destination-Host: A Diameter entity servicing the request or the final destination for the Diameter message
2. Peer: Connects with the Diameter entity that establishes transport (TCP) connection directly. It can be either a DS or DRA.
3. Route Entry: Defines a path for reaching a Diameter entity from the gateway, essentially the next immediate peer to contact to reach a particular Diameter entity.
4. Routing Table: The diabase library maintains the table of route-entries. The construction of a routing table is through the configuration under the Diameter Endpoint (peer and route-entry CLI) and also dynamically learned from the response messages.
5. Multipath Routes: Multipath Routes are when there are more than one route entry to reach a particular Diameter entity.

Route Flags

Following are the route flags in Diameter Endpoint:

Table 6: Route Flags

Flag Name	Description
Static (S)	Static routes are the route entries based on the configuration under the Diameter Endpoint.
Path-Cache (P)	Every Diameter session has a reference to the route entry in the routing table. When you select a static route for a session, a copy of that static route or a path-cache route is created. It is as same as the static route except for the flag value.
Dynamic (D)	Dynamic routes are the route entries learned from the response messages from the peer.

Route Status

The route statuses for peers in Diameter Endpoint are as follows:

Table 7: Route Status

Route Status	Description
Pending	All the static routes start with this status and confirms the registration of an RPC host.
Active	After a successful Ping method, the Pending status moves to Active status.
Inactive	If the Ping method fails, it marks all the routes with that peer as Inactive. This status remains until the peer becomes reachable.
Failed	An Active route moves to a Failed status when it meets the route failure criteria.
Deleted	None of the routes are hard deleted but they remain soft deleted with a Deleted status.
Expired	Dynamic routes, after expiration, are set to Expired status.
Cloned	When the Path-Cache route is created from the Static route, the status of the original static route changes to Cloned. Cloned is an internal state and not borrowed from legacy implementation.

Considerations for Peer Management

Following are the considerations for peer management in Diameter Endpoint:

- The DRA creates a route table for every endpoint profile. Peers configured within the endpoint are a primary source of route entries in the route table.
- Peers have an optional CLI to set the Destination-Host-Name that is different from the peer name. If the destination host name is not set, it copies the peer name into the Destination-Host-Name.
- Each peer creates a direct and a realm-based route entry. It adds one more route entry when the Destination-Host-Name is different from the peer name.
- The default weight of the static route is 10. You cannot configure the weight of routes formed from the peer entries. However, you can override them with a higher value using a route entry with the same host and realm combination.
- The route table gets populated on the endpoint profile update notification, with each entry status as ‘Pending’.
- You can add the dynamic route in the response path only if the session is referring a realm-based route entry. For a direct route entry, you cannot add the dynamic route in the response path.
- The weight of a new dynamic route is the same as the route selected in the request path.
- Dynamic routes have the expiration time configured. The expiration time of dynamic route is updated with the new expiration time if there is a message exchange using that dynamic route or if the response

is pointing to that dynamic route where the origin-host and origin-realm in response is different from the destination-host and destination-realm of request.

Route Selection

The route selection process and its usage in retries are as follows:

1. While providing a response for a CCR-I or CCR-U message, the Diameter endpoint sends the details of the primary host, secondary host, realm, and bounded peer as part of the route data. The same details are provided by the service as part of subsequent requests.
2. In the absence of a route data field in the request from the service, the Diameter endpoint determines primary host, secondary host, and realm from the host selection profile, which is associated with the client profile, specified in the request.
3. Host selection profile has a mandatory primary and an optional secondary host-realm pair. While realm is an optional CLI, it sets to a default value.
4. The host and realm serve as an input to the route search algorithm to find possible multiple matching routes. The order depends on the status, flag and weight.
5. In the absence of host selection profile configuration, the peer management module selects and sorts all the active routes in the route table.



Note Routes selection happens through the round robin method from the ordered routes list of the primary host or secondary host. This method avoids trying all the routes of the primary host and then trying all the routes of the secondary host.

Route Sort Order

The route sort orders for entries in the Diameter peer are as follows:

- The routes with the active status take the highest precedence.
- Among the routes with the same status, if a route matches the bounded peer, then that route takes the highest precedence. Then, the next precedence is for the dynamic routes followed by the Path-Cache routes.
- The peer arranges multiple routes of the same flags among themselves based on their weight, where the higher value takes precedence.
- Weight always outweighs routes with the exact match of host and realm entries.
- Exact match entries have the priority only if there are two routes with equal weights.

Host Selection

This section describes about the host selection, its implementation, and the sample configuration for host selection on the Diameter endpoint.

Feature Description

The Diameter endpoint keeps the data of an individual profile name with algorithm and list of primary and secondary host details. The local data structure can keep a maximum of 64 host details in the indexed form. The host details include the primary host-realm, and secondary host-realm.

How it Works

The peer management module sends the request to the host selection module with the profile name, MSISDN, and IPv4 address. The host selection module checks the profile name in the local data structure that is populated from the configuration map. If the profile name matches, then it derives the host details with following algorithms.

- MSISDN modules—The MSISDN is converted into numeric value. Then, the index is calculated by the available number of MSIDN mod host entries.
- IP address modules—IP address is converted into numeric value. Then the index is calculated by the available number of numeric IP address value mod host entries.
- Round robin—The default algorithm is the round robin of available entries per profile name.



Note A host selection profile is bound with the Diameter client. You can bind one host selection profile for the multiple Diameter clients.

Configuring Host Selection through Diameter Endpoint

Diameter host selection configuration is available in the Diameter profile. You can configure multiple Diameter hosts and then bind these hosts to a different Diameter client.

To configure the host selection through the Diameter endpoint, use the following sample configuration:

```
config
  profile diameter-host-selection host_selection_name
    algorithm algorithm-name
    hosts hosts_number
      primary host host_ip_address
      primary realm realm_name
      secondary host host_ip_address
      secondary realm realm_name
    end
```

NOTES:

- **profile diameter-host-selection** *host_selection_name*: Specify the Diameter host selection profile name.
- **algorithm***algorithm-name*: Choose the algorithm to select the host. *algorithm-name* has the **ipaddr-modulus**, **msisdn-modulus**, and **round-robin** values.
- **hosts** *hosts_number*: Specify the precedence of the host in the form of index from 1-64.
- **primary host** *host_ip_address*: Specify the primary host name.
- **primary realm** *realm_name*: Specify the primary host realm.

- **secondary host** *host_ip_address*: Specify the secondary host name.
- **secondary realm** *realm_name*: Specify the secondary host realm.

Configuration Example

The following is an example configuration of the host selection.

```
profile diameter-host-selection hs1
 algorithm ipaddr-modulus
 hosts 1
  primary host 209.165.200.239
  primary realm cisco.com
  secondary host g192.168.2.1
  secondary realm google.com
 exit
 hosts 2
  primary host 209.165.200.240
  primary realm facebook.com
  secondary host 209.165.200.241
  secondary realm conflunt.com
 exit
exit

profile diameter-client dc1
 endpoint       gxProf1
 host-selection hs1
exit
profile diameter-client dc2
 endpoint       gyProf1
 host-selection hs1
exit
```

Route Failure

Feature Description

A route failure counter starts at zero. This counter is incremented in the following scenarios:

- When the transaction is attempted using that route times out.
- When a response is received with the result code matching the configured failure result codes.

The count is decremented each time that a successful response is received. When the failure count exceeds the threshold, the route is marked as failed. Periodically, the route failure count is reduced by a recovery threshold value to allow the failed route to become available.

Configuring Route Failure

To configure the route failure, use the following sample configuration:

```
config
  profile diameter-endpoint interface_name
     name profile_name
     internal-vip ip_address
        route-failure deadtime deadtime_valueresult-code result_code_value
threshold  number
        route-failure result-code result_codes_value
```

```
        route-failure threshold  threshold_number
        route-failure recovery-threshold percent  recovery_threshold_percentage
exit
```

NOTES:

- **route-failure deadtime** *deadtime_value***result-code** *result_code_value* **threshold** *number*: Specify the duration in seconds for which the system keeps the route in the **FAILED** status. After the configured duration expires, the system changes the status to **AVAILABLE**. *deadtime_value* must be an integer in the range of 1–86400. The default value is 60.
- **route-failure result-code** *result_codes_value*: Specify the answer messages that are to be considered as failures, in addition to the requests that time out.

Note You can specify up to 16 result codes.

- **route-failure threshold** *threshold_number*: Specify the number of errors that cause the **FAILED** status. The default value is 16.

Note The error counter begins at zero. In a case of a good response, the error counter decrements or increments. This counter does not decrement below zero or increment above the configured threshold number.

- **route-failure recovery-threshold percent** *recovery_threshold_percentage*: Specify the percentage value at which the failure counter is reset when provisionally changing the status from **FAILED** to **AVAILABLE**. For example, a failure counter of 16 caused the **AVAILABLE** status to change to **FAILED** status. After the configured deadtime expires, the status changes to **AVAILABLE**. If *recovery_threshold_percentage* is configured with 75 percent, the failure counter resets to 12, which is 75 percent of 16. The default value is 90.

Configuration Example

The following is an example configuration of route failure.

```
profile diameter-endpoint gydc01
   name gyProf1
   internal-vip ip_address
      route-failure deadtime  600
      recovery-threshold 16
      percent 90
      result-code rc1
exit
```

Configuration Verification

To verify if the Diameter endpoint on the Gx or Gy interface is configured, use the following **show running-config profile diameter-endpoint** command:

```
[smf] smf# show running-config profile diameter-endpoint
 profile diameter-endpoint gx
 internal-vip 209.165.200.234
 name gxProf1
```

```
   internal-vip ip_address
      route-failure deadtime 600
      recovery-threshold 16
      percent 90
      result-code rc1
exit
```

Bounded Peer

Feature Description

A peer with which the last Credit Control Request - Initial (CCR-I) or Credit Control Request - Update (CCR-U) interaction happened is considered as the bounded peer. Based on current request processing, the Diameter endpoint communicates to the service for the primary host or secondary host and the bounded peer. Then, the service communicates to the Diameter endpoint for the primary host or secondary host and the bounded peer that is to be used for the request processing.

Matching of route for the bounded peer takes the highest precedence for sending the subsequent CCR-U or Credit Control Request - Terminate (CCR-T) message. If the route list of a host has an active entry and the peer matches with the bounded peer, then the route entry takes the highest precedence for the request processing.

How it Works

This section describes how the bounded peer works.

- The details on the primary and secondary host and the bounded peer are included in the request from the service. Similarly, these details are included in the response to the service.
- If the route list of host has only dynamic route entry, then static or patchcache route entry of the peer associated with route is added to list.
- If the route list of a host has one or more active entries and the peer matches with the bounded peer, then the route entries take priority for request processing.
- In case of RAR, the diameter-ep fills host, realm, and bounded peer for primary. The service merges the details with existing Routing Data so that the secondary host/realm/bounded peer information is not lost.

Failure Handling

Feature Description

Failure handling is managed by configuring the failure handling profiles for the Diameter messages and failures. With this configuration, the behavior of a Diameter pod is determined in failure scenarios. You can configure any number of failure handling profiles.

After configuring the failure handling profiles, associate them with a specific Diameter client profile. If you don't associate a configured failure handling profile with a Diameter client profile, then a default failure profile is considered.

How it Works

Failure handling works in the following way:

- If there is a failure, the Diameter endpoint fetches the applicable failure handling profile.
- If the failure handling profiles are not defined or if a profile is defined with no message and failure type that should match the failed request, then the default failure handling is applied. The default failure handling is no retry with the **action terminate** and subaction as *with-term-req*, which implies with no termination request.
- Diameter endpoint reattempts request with the alternate peers or routes until the request is successful or the reattempt the reaches the configured retry count as specified by applicable failure handling profile.
- Routes are selected in the round robin method from the ordered routes list of each host. For example, on failure for route of host1, the next route in the list of routes of host2 is used for retrying. During retries, if the route list of one of the hosts exhausts, then the remaining route of other hosts is used for retries.
- When interacting with DRA for CCR-U/T, if the route list of primary does not have any dynamic route then the secondary is considered first for sending a message.
- During retries if an already tried route is found, then the route is skipped.
- If the applicable peer or route count is lower than the retry count, then remaining retries are discarded after all the applicable peers or routes are attempted.
- Retry count, which is determined during the first failure after the request is sent out, is used. If there is a failure during retries and that failure corresponds to a different failure handling profile with a different retry count, then also Diameter endpoint continues with the retry count that is determined during the first failure.
- Retry counting starts only after a minimum of one request is sent. Failure before the retry count is not considered.
- Even after all the reattempts or all peers or routes reattempts, if the request is not successful, then the Diameter endpoint send the action and subaction as part of response to the service.
- Action and subaction applicable for the most recent failure is used in the response.
- No additional handling is done on the Diameter endpoint for any action or subaction because action for those are taken on the service side as follows:

  Action

  - **continue**: Continue with the session
  - **terminate**: Terminate the session. This action is the default action.

  Subaction

  - **discard-traffic**: Block or discard the data traffic. This subaction is associated with action with the **continue** . action and is applicable only for the Gy interface.
  - **local-fallback**: Apply local policies. This subaction is associated with the **continue** action.
  - **none**: No action. This subaction is associated with the Associated with **continue** action and is applicable only for the Gy interface. **none** is the default option for the Gy calls for the **continue** action.

- **retry-server-on-event**: Reconnect to server on credit control update requests. This subaction is associated with the **continue** action and is applicable only for the Gx interface.
- **send-ccrt-on-call-termination**: Send CCR-T request to PCRF for credit control update request failure. This subaction is associated with the **continue** action and is applicable only for the Gx interface. **send-ccrt-on-call-termination** is the default option for Gx calls for **continue** action.
- **with-term-req**: Terminate the session by sending the termination request (CCR-T). This subaction is associated with the **terminate** action and is applicable only for the Gx interface. **with-term-req** is the default option for the Gx calls for **terminate** action.

Configuring Failure Handling

Configuration of failure handling includes the following steps:

1. Configuring Failure Handling Profile, on page 29
2. Associating the failure handling profile to the Diameter client profile. To configure the Diameter client profile, see the Configuring Diameter Client Profile, on page 9 procedure.

Configuring Failure Handling Profile

To configure a failure handling profile, use the following sample configuration:

```
config
  profile failure-handling failure_handling_profile_name
     interface diameter
        message [ any | credit-control-initial | credit-control-terminate
 | credit-control-update ]
           failure-type [ any | local-error | result-code {result-code-value
| result-code-range-start-value | result-code-range-end-value |
comma-separated-result-code-value-or-range } | experimental-result-code
experimental-result-code-value | experimental-result-code-start-value |
experimental-result-code-end-value |
comma-separated-experimental-result-code-value-or-range
}experimental-result-code-value | experimental-result-code-start-value |
experimental-result-code-end-value | comma-separated-experimental-result-code-value-or-range
 ]
             retry count
            action [ continue discard-traffic | local-fallback | retry-server-on-event
 | send-ccrt-on-call-termination | terminate with-term-request | without-term-request ]

  exit
```

NOTES:

- **profile failure-handling** *failure_handling_profile_name*: Specify a name for the failure handling profile.
- **interface diameter**: Specify the failure handling profile for the Diameter interface.
- **message [any | credit-control-initial | credit-control-terminate | credit-control-update]**: Choose a message value from the available options. This is a mandatory parameter.

- **failure-type [any | local-error | result-code {***result-code-value* | *result-code-range-start-value* | *result-code-range-end-value* | *comma-separated-result-code-value-or-range* **| experimental-result-code** **experimental-result-code-value | experimental-result-code-start-value | experimental-result-code-end-value | comma-separated-experimental-result-code-value-or-range** **}***experimental-result-code-value* | *experimental-result-code-start-value* | *experimental-result-code-end-value* | *comma-separated-experimental-result-code-value-or-range* **]**: Specify a Diameter failure type for which an action must be taken. Choose a failure type value from the available options. This is a mandatory parameter.

Note You can configure multiple failure types.

- **retry** *count*: Specify the number of alternate peers or routes to retry on receiving a failure response. The default value is zero.
- **action [continue** *discard-traffic* | *local-fallback* | *retry-server-on-event* | *send-ccrt-on-call-termination* **| terminate** *with-term-request* | *without-term-request* **]**: Choose an action value as **continue** or **terminate** . Choose subactions of the selected action from the available options, as required.

Note To clear a subaction or reset a subaction of **continue** or **terminate** action to default, perform the following steps before saving the configuration:

1. **no profile failure-handling** *failure-handling-profile-name*

   interface diameter message*diameter-message*
   failure-type*failure_name* **result-code** *result_code*
   action*action_name sub_action*
   exit

2. **profile failure-handling** *failure-handling-profile-name*
   interface diameter message*diameter_message*
   failure-type*failure_name* **result-code** *result_code*
   action*action_name*
   exit

Configuration Example

The following is an example configuration of the failure handling.

```
profile failure-handling fh1
  interface diameter
    message credit-control-initial
      failure-type any
        retry 2
        action terminate with-term-req
      exit
      failure-type local-error
        retry 3
        action continue local-fallback
      exit
      failure-type result-code 3000
```

```
        retry 2
        action continue discard-traffic
       exit
      exit
      failure-type result-code 4000
        action continue retry-server-on-event
       exit
      exit
      failure-type result-code 4001-4010
        action continue send-ccrt-on-call-termination
       exit
      exit
      failure-type result-code 4011-4020,4025,4030-4040
        action terminate without-term-req
       exit
      exit
      failure-type exp-result-code 5000
        action terminate with-term-req
       exit
      exit
      failure-type exp-result-code 5010-5015
        action terminate
       exit
      exit
    exit

    message credit-control-terminate
      failure-type any
        retry 2
        action terminate
      failure-type local-error
        action continue retry-server-on-event
      exit
    exit

    message credit-control-update
      failure-type any
        action continue send-ccrt-on-call-termination
      exit
      failure-type local-error
        action continue discard-traffic
      exit
    exit

    message any
      failure-type any
        action continue discard-traffic
      exit
      failure-type local-error
        action continue
      exit
    exit
  exit
exit

profile failure-handling fh2
  interface diameter
    message credit-control-update
      failure-type any
        retry 2
        action continue
      exit
      failure-type local-error
        action terminate without-term-req
```

```
      exit
    exit
  exit
exit
```

Configuration Verification

To verify if the failure handling is configured, use the following **show running-config profile diameter-client** command:

```
show running-config profile diameter-client
   profile diameter-client dc2
    endpoint gyProf1
       failure-handling-profile FH1
exit
```

OAM Support

Bulk Statistics Support

The SMF maintains the following metrics as part of the Diameter endpoint.

diameter_request_message_total

Description: Indicate the count of diameter requests processed by diameter endpoint.

Note

- The peer_address counter is empty for inbound requests and depending on the point of failure, this counter can be empty for the outbound requests.
- The retry counter is always logged as zero for IPC timeout.
- The transaction_type counter is hard-coded to "origin".
- The endpoint_name counter is empty for inbound requests and depending on the point of failure, this counter can empty for outbound requests.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **interface**: gx, gy
- **message_name**: ccri, ccru, ccrt, rar, asr peer_address
- **status**: attempted, peer_down, err_cfg, err_maxout, timeout_ipc, err_ipc, err_unmarshal
- **retry**: retry count gr_instance transcation_type
- **endpoint_name**: name of endpoint profile used during processing
- **message_direction**: inbound, outbound

Label Production:

- interface
- message_name
- peer_address status

Label Debug:

- retry
- gr_instance
- transaction_type

Label Trace:

- endpoint_name
- message_direction

diameter_response_message_total

Description: Indicate the count of diameter responses that the Diameter endpoint processes.

Note

- The peer_address counter is empty for inbound requests and depending on the point of failure, this counter can be empty for the outbound requests.
- The result_code counter is logged as " " for outbound responses, local failure, response timeout, and IPC timeout.
- The action or subaction counter is empty for outbound responses and "success" responses.
- The endpoint_name counter is empty for inbound requests and depending on the point of failure, this counter can empty for outbound requests.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **interface**: gx, gy
- **message_name**: ccai, ccau, ccat, raa, asa
- **peer_address**
- **status**: success, err_cfg, err_maxout, err_send, timeout_res, timeout_ipc, err_ipc, err_unmarshal, err_rc, err_exp_rc
- **result_code**: result-code or experimental-result-code that the Diameter node encountered during the response processing
- **action**: continue, terminate

- **subaction**: discard-traffic, local-fallback, retry-server-on-event, send-ccrt-on-call-termination, with-term-req, without-term-req
- **endpoint_name**: name of endpoint profile used during processing
- **gr_instance**
- **message_direction**: inbound, outbound

Label Production:

- interface
- message_name
- peer_address status
- status
- result_code

Label Debug:

- action
- subaction
- gr_instance

Label Trace:

- endpoint_name
- message_direction

diameter_response_message_seconds_total

Description: Indicate the cumulative response time in seconds of diameter requests that the Diameter endpoint processes.

Note

- The peer_address counter is empty for inbound requests and depending on the point of failure, this counter can be empty for the outbound requests.
- The result_code counter is logged as " for outbound responses, local failure, response timeout, and IPC timeout.
- The action or subaction counter is empty for outbound responses and "success" responses.
- The endpoint_name counter is empty for inbound requests and depending on the point of failure, this counter can empty for outbound requests.
- This counter is not incremented for local failure or IPC timeout.

Default Level: MetricsVerboseLevelProduction

Label Details:

- **interface**: gx, gy
- **message_name**: ccai, ccau, ccat, raa, asa
- **peer_address**
- **status**: success, err_cfg, err_maxout, err_send, timeout_res, timeout_ipc, err_ipc, err_unmarshal, err_rc, err_exp_rc
- **result_code**: result-code or experimental-result-code that the Diameter node encountered during the response processing
- **action**: continue, terminate
- **subaction**: discard-traffic, local-fallback, retry-server-on-event, send-ccrt-on-call-termination, with-term-req, without-term-req
- **gr_instance**
- **endpoint_name**: name of endpoint profile used during processing
- **message_direction**: inbound, outbound

Label Production:

- message_name
- origin_host
- origin_realm
- disconnect_cause
- result_code

Label Debug:

- action
- subaction
- gr_instance

Label Trace:

- endpoint_name
- message_direction

diam_base_msg_total

Description: Indicate the count of diameter base message requests that the Diameter endpoint processes.

Note

- The DPR counter is supported only for the DPR that is received from a peer.
- The result_code counter is logged as 5012 if the DPR received has the mandatory AVP missing or if origin-host and origin-realm in DPR does not match with destination-host and destination-realm in connection details.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **message_name**: DPR
- **origin_host**
- **origin_realm**
- **disconnect_cause**: REBOOTING, BUSY, DO_NOT_WANT_TO_TALK_TO_YOU
- **result_code**: 2001, 5012
- **gr_instance**

Label Production:

- message_name
- origin_host
- origin_realm
- disconnect_cause
- result_code

Label Debug:

- gr_instance

Label Trace:

- endpoint_name
- message_direction

diam_base_msg_seconds_total

Description: Indicate the cumulative response time in seconds of diameter base message requests that the Diameter endpoint processes.

Note

- The DPR counter is supported only for the DPR that is received from a peer.
- The result_code counter is logged as 5012 if the DPR received has the mandatory AVP missing or if origin-host and origin-realm in DPR does not match with destination-host and destination-realm in connection details.

Default Level: MetricsVerboseLevelProduction

Label Details:

- **message_name**: DPR
- **origin_host**
- **origin_realm**
- **disconnect_cause**: REBOOTING, BUSY, DO_NOT_WANT_TO_TALK_TO_YOU
- **result_code**: 2001, 5012
- **gr_instance**

Label Production:

- interface
- message_name
- diameter_dictionary_type
- status
- unknown_avp

Label Debug:

- gr_instance

diameter_encode_message_total

Description: Indicate the count of diameter base message requests that the Diameter endpoint processes.

Note

- The DPR counter is supported only for the DPR that is received from a peer.
- The result_code counter is logged as 5012 if the DPR received has the mandatory AVP missing or if origin-host and origin-realm in DPR does not match with destination-host and destination-realm in connection details.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **interface**: gx, gy, ""
- **message_name**: ccri, ccru, ccrt, raa, asa
- **endpoint_name**: name of endpoint profile used during processing
- **dict_name**
- **status**: success, failure, partial
- **unknown_avp**: 0, 1
- **gr_instance**

Label Production:

- interface
- message_name
- diameter_dictionary_type
- status
- unknown_avp

Label Debug:

- status_code
- gr_instance

diameter_decode_message_total

Description: Indicate the count of decoding that the Diameter endpoint performed.



Note

- The interface is empty for unsupported interface or command code.
- The message_name counter is empty for the unsupported interface or command code
- The dict_name counter is empty when default dictionary is used. Else, this counter indicates the name of dictionary that is used for an operation.
- The status counter is logged as failure only for the unsupported interface or command code. This counter is logged as partial if an issue that is seen during the encoding is ignored. For example, an unknown AVP.
- The unknown_avp counter indicates if any unknown AVPs are found during encoding, where zero indicates not found and 1 indicates found.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **interface**: gx, gy, ""
- **message_name**: ccai, ccau, ccat, rar, asr
- **endpoint_name**: name of endpoint profile used during processing
- **dict_name**
- **status**: success, failure, partial
- **unknown_avp**: 0, 1
- **gr_instance**

Label Production:

- interface
- message_name
- diameter_dictionary_type
- status
- unknown_avp

Label Debug:

- status_code
- result_code
- gr_instance

Label Trace:

- endpoint_name

diameter_pod_status

Description: Indicate the pod status as active or standby.



Note

- Value 1 indicates the status as active and zero indicates the status as standby.
- The VIP counter is not used to determine the pod status when a Diameter pod is running in server mode. In this case, VIP is empty in the server mode.

Metrics Type: Gauge

Default Level: MetricsVerboseLevelProduction

Label Details:

- **vip**: VIP used for determining pod status

Label Production:

- vip

Label Debug:

- gr_instance

Label Trace:

- endpoint_name

dispatch_error_total

Description: Indicate the count of inbound requests that had error or timeout during dispatch to service.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **application**
- **command_code**
- **error_code**
- **gr_instance**

Label Production:

- application
- command_code
- error_code

Label Debug:

- gr_instance

dispatch_error_seconds_total

Description: Indicate the cumulative time in seconds spent during dispatching of inbound requests to service that had error or timeout.

Default Level: MetricsVerboseLevelProduction

Label Details:

- **application**
- **command_code**
- **error_code**
- **gr_instance**

Label Production:

- application

- command_code
- error_code

Label Debug:

- gr_instance

policy_engine_message_total

Description: Indicate the count of messages that are sent to service for which response is received.

Metrics Type: Counter

Default Level: MetricsVerboseLevelProduction

Label Details:

- **application**
- **command_code**
- **gr_instance**

Label Production:

- application
- command_code

Label Debug:

- gr_instance

policy_engine_message_seconds_total

Description: Indicate the cumulative time in seconds spent during processing of a message sent to service.

Default Level: MetricsVerboseLevelProduction

Label Details:

- **application**
- **command_code**
- **gr_instance**

Label Production:

- application
- command_code

Label Debug:

- gr_instance

Troubleshooting Information

Monitor Protocol and Monitor Subscriber

Feature Description

SMF supports the SUPI and IMSI subscriber key for the Monitor Protocol and Monitor Subscriber functionality for the Diameter messages. Use the Monitor Protocol to add the interface type as Diameter.

Monitor Subscriber

Using the CLI commands, you can view the messages for a specific subscriber or a set of subscribers. For interworking, use the NF service as SMF.

Monitor Protocol

For the Diameter interface, SMF uses the Monitor Protocol for packets.



Note For the Diameter client, the monitor protocol doesn't work until you connect a minimum of one peer to the leader pod.

How it Works

You can configure the Monitor Subscriber and Monitor Protocol feature through the CLI commands in the Ops Center.

Configuring the Monitor Subscriber and Monitor Protocol Support

This section describes how to configure the Monitor Subscriber and Monitor Protocol support.

Monitoring the Subscriber Session

To monitor the subscriber in the SMF, use the following CLI command:

monitor subscriber [capture-duration *duration* **| dump filename** *file_name* **| gr-instance** *gr_instance_id* **| imei** *imei_id* **| imsi** *imsi_value* **| internal-messages [yes] | list** *file_list* **| nf-service [sgw | smf] | supi** *supi_id* **| transaction-logs [yes | no]]**

NOTES:

- **capture-duration** *duration*: Specify the duration in seconds during which a monitor subscriber is enabled. The default value is 300 seconds (5 minutes). This is an optional parameter.
- **dump filename** *file_name*: Specify the file name to view the sorted file on the SMF Ops Center.
- **gr-instance** *gr_instance_id*: Specify the GR instance ID. The instance ID 1 denotes the local instance ID.
- **imei** *imei_id*: Specify the subscriber IMEI. For example: 123456789012345, *

- **imsi** *imsi_value*: Specify the subscriber IMSI. For example: 123456789, *
- **internal-messages [yes | no]**: Enable or disable viewing the internal messages. By default, **yes** option is disabled. This is an optional parameter.
- **nf-service [sgw | smf]** : Enable the specified NF service. By default, **nf-service** is set to none.



Important The **nf-service** keyword replaces the **namespace** keyword in Release 2021.02 and beyond.

- **supi** *supi_id*: Specify the subscriber identifier. For example: imsi-123456789, imsi-123*
- **transaction-logs [yes | no]**: Enable transaction logs when set to **yes**. By default, this parameter is disabled. This is an optional parameter.

  To view the transaction history logs, use the **dump transactionhistory** command.



Note The most recent transaction logs are stored in a circular queue of size 1024 transaction logs.

Monitoring the Diameter Interface Protocol

To monitor the Diameter interface protocol on the SMF, use the following CLI command:

monitor protocol interface diameter [capture-duration *duration* **| gr-instance** *gr_instance_id* **| pcap yes gr-instance** *gr_instance_id* **]**

NOTES:

- **interface diameter** : Specify the interface name as Diameter.
- **capture-duration** *duration*: Specify the duration in seconds during which pcap is captured. The default is 300 seconds (5 minutes).
- **gr-instance** *gr_instance_id*: Specify the GR instance ID. The instance ID 1 denotes the local instance ID.
- **pcap yes gr-instance** *gr_instance_id*: Configure this option to enable PCAP file generation. By default, this option is disabled.

Debugging Data Collection

While debugging issues with the Diameter, use the output of the following commands through the Ops Center:

- **kubectl get pods -n** *namespace*
- **kubectl get statefulsets -n** *namespace*
- **kubectl get cm -n** *namespace*
- **kubectl get service -n** *namespace*

- **helm list**
- **show full-configuration** or **show running-configuration**