

SCTP Multihoming and Stack Parameters Support

- Feature Summary and Revision History, on page 1
- Stream Control Transmission Protocol (SCTP) Multihoming, on page 2
- SCTP Multihoming and Stack Parameters Support, on page 3

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	AMF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled - Always-on
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
The following enhancements were introduced: • Support to configure SCTP stack parameters. • Support for multiple SCTP and protocol pod pairs • Support for show SCTP peers CLI	2022.01.0
First introduced.	2020.03.0

Stream Control Transmission Protocol (SCTP) Multihoming

Stream Control Transmission Protocol (SCTP) is a message-oriented, reliable, transport protocol. SCTP directly supports multihoming transport protocol that runs on top of an IP network. SCTP used as a protocol with pods, services, and network policy.

Multihoming is the ability of an SCTP association to support multiple IP paths to its peer endpoint.

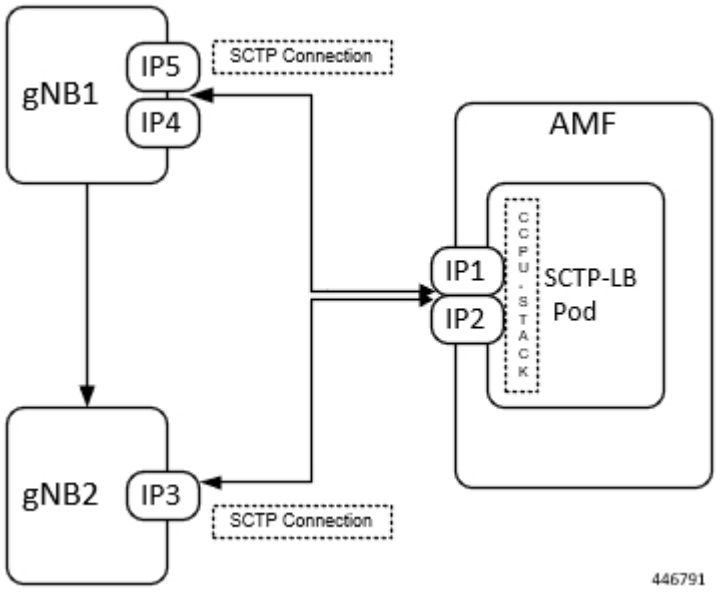
Feature Description

AMF supports a single SCTP pod (single instance) for SCTP multihoming, where the SCTP stack comes up with a list of supported host IPs. As part of the association formation, the association ID corresponds to the list of IPs, instead of a single IP.

The stack also supports multihoming for one-to-many and many-to-many connections. This support continues until any of IPs are available on either side of the SCTP end points (AMF and gNB). At the same time, traffic over multiple IPs is also possible.

The following figure represents the support structure for SCTP Multihoming:

Figure 1: SCTP Multihoming Support



Based on the represented figure, the following SCTP associations are formed:

1. Association ID – 0 [{IP1, IP2}, {IP4, IP5}]
2. Association ID – 1 [{IP1, IP2}, {IP3}]

Limitations

The SCTP multihoming feature has the following limitations:

- Currently not supported:
  - Dynamic addition or removal of IPs from the multihoming configuration without the pod restart
  - Dynamic service config delete and also dynamic IP change
- Currently observed and recommended:
  - If both member of the pair goes down, there's no redundancy.
  - Even though multiple protocol pairs are supported, there's a limitation with multiple protocol pairs. As a best practice, recommended to configure only one protocol pod pair.
  - One gNB can have only one association with AMF. Multiple associations with same gNB aren't supported.

SCTP Multihoming and Stack Parameters Support

This section describes support for the following features:

- SCTP Configurable Stack Parameters
- Multiple SCTP and Protocol Pod Pairs

Feature Description

Before implementing this feature, AMF needs separate deployment of the following five namespaces for scalability. Each AMF namespace supports the following:

- A pair of SCTP pods (active-standby)
- A pair of Protocol pods (active-standby)
- Extra pods getting deployed on Ops Center and ETCD

A single AMF namespace supports and deploys multiple SCTP pods and protocol pods. SCTP pods support multihoming and some SCTP stack-related parameters are configurable.

SCTP Configurable Stack Parameters

SCTP uses the multihomed host to provide fast failover and associated endurance during hardware failures. Using the associated parameters, the following activities are supported:

- Creating and customizing the required stack
- Configuring the resources by modifying the parameter values, which are later used in the stack template.
- No need to enter hardcoded values in multiple templates to specify different settings.

Multiple SCTP and Protocol Pod Pairs

Pods are tagged with one or more labels. The labels are later used to select and manage groups of pods in a single operation. The labels are stored in a key-value format in the metadata hash.

For more information, see the SCTP Multihoming and Stack Parameters Support, on page 1 chapter.

How it Works

This section describes how this feature works.

Multihoming Support

Supports multiple IP addresses for the SCTP stack.

Multiple SCTP and Protocol Pod Pairs

In a single AMF namespace, multiple SCTP pairs can be configured. This way on same AMF, SCTP pods can be scaled up, as per the requirement.

Configurable SCTP Endpoint Stack Parameter

Provides the option to configure multiple SCTP Endpoint stack parameters which includes the following:

- RTO
- Association
- Client-idle-timeout
- Sack
- MTU Size

Feature Configuration

Configuring this feature includes the following steps:

- Configuring Multiple SCTP and Protocol Pod Pairs
- Configuring SCTP Endpoint Parameters

Configuring Multiple SCTP and Protocol Pod Pairs

To configure multiple SCTP and Protocol pod pairs use the following configuration:

```
config
    instance instance_id instance_id
    endpoint endpoint_name
    replicas replicas_per_node
    service service_name
        interface interface_name instancetype instance_type
        internal-port internal_port_config
        vip-ip ip_address vip-port port_number
        offline vip-interface vip_interface_name
        vip-ip6 ip_address vip-ipv6-port port_number
        offline vip-interface
        end
```

NOTES:

- **instance instance_id** *instance_id*—Specify the endpoint instance ID. Must be an integer in the range of 1-4.
- **endpoint** *endpoint_name*—Specify the endpoint name.

Note In this release, it is recommended not to configure any service under NGAP. Currently, only one service can be configured under NGAP.

- **replicas** *replicas_per_node*—Specify the number of replicas per node.
- **service** *service_name*—Specify the service name.
- **interface** *interface_name*—Specify the endpoint interfaces, as the name of the SCTP in this multiple SCTP configuration. **sctp-1-sctp-primary** is an example.
- **instancetype** *instance_type*—Specify the instance type. Must be one of the following:
  - Dual
  - IPv4
  - IPv6

  The default value is Dual.
- **internal-port** *internal_port_config*—Specify the internal base-port to start the endpoint. It includes your required internal-ports and their ID from the list of available ports.
  - admin—Admin port for SCTP. The default value is 7879.
  - ipc—IPC port for SCTP. The default value is 9005.
  - keepalived—keepalived port for SCTP. The default value is 29000.
  - metrics—metrics port for SCTP. The default value is 7083.
  - pprof—pprof port for SCTP. The default value is 7850.

Note It is mandatory to configure the internal-ports CLI when more than one SCTP service is configured or more than one AMF is deployed on the same k8 cluster.

- **vip-ip** *ip_address* **vip-port** *port_number* **offline**—Specify the IPv4 address of the pod on which the VIP is enabled. Also, specify the interface and port number. This configuration marks vip-ip as offline or standby.

Note When AMF receives SCTP INIT from IPv4 address and no address is included in INIT chunk address list, AMF responds with all IPv4 and IPv6 addresses in SCTP INIT_ACK.

- **vip-interface** *vip_interface_name* —Specify the VIP interface and port number. This configuration marks **vip-ip** as offline or standby.

Note To support multi-homing, AMF listens on multiple IP address as configured.Currently, AMF uses VIP to support high availability of SCTP pods. For multi-homing, multiple VIP addresses should be configured.

It is recommended to use different physical interfaces for each different VIP so that AMF can have different routes for each VIP address.

- **vip-ip6** *ip_address* **vip-ipv6-port** *port_number*—Specify the new IPv6 address and port number.

Note When AMF receives SCTP INIT from IPv6 address and no address is included in INIT chunk address list, AMF respond with only IPv6 addresses in SCTP INIT_ACK.

After the VIP-IP and VIP-Ports are up, modify the gNBs configuration to refer to the new VIP-IP and port.

Configuration Example

The following is an example configuration.

- CLI at endpoint level: If more than one AMF is deployed in same cluster with one single SCTP service.

```
config
  instance instance-id 1
    endpoint sctp
     replicas 2
     internal-port metrics 9705 admin 9703 ipc 9701 pprof 9707 keepalived 29001
     vip-ip 209.165.201.15 vip-port 1000
     end
```

- CLI at SCTP service level: If more than one SCTP service is configured.

```
config
 instance instance-id 1
  endpoint sctp
   replicas 2
   nodes    2
   parameters mtu-size 1500
   service sctp-1 interface sctp instancetype Dual
    internal-port metrics 9705 admin 9703 ipc 9701 pprof 9707 keepalived 29001
    vip-ip 209.165.201.15 vip-port 1000
   service sctp-2 interface sctp instancetype Dual
    internal-port metrics 9715 admin 9713 ipc 9711 pprof 9717 keepalived 29011
    vip-ip 209.165.201.16 vip-port 1000
    end
```

Configuration Verification

To verify the configuration:

```
show running-config instance
```

Configuring SCTP Endpoint Parameters

Table 3: Feature History

Feature Name	Release Information	Description
AMF jumbo frame support on N2 interface	2026.03.0	You can now configure the N2 interface to support Jumbo Frames with an MTU size of 9,000 bytes. This feature allows for the adjustment of the MTU size within the SCTP stack parameters, ensuring the AMF can effectively handle larger data packets required for specific customer deployment environments and high-performance networking.

To configure the SCTP endpoint parameters, use the following configuration:

```
config
    instance instance_id instance_id
    endpoint endpoint_name
        replicas number_of_nodes
            parameters rto initial rto_initial
            parameters rto min rto_min
            parameters rto max rto_max
            parameters association valid-cookie-life valid_cookie_life
            parameters association heartbeat-interval heartbeat_interval
            parameters association path-max-retry-count path_max_retry_count
            parameters client-idle-timeout  client_idle_timeout_value
            parameters sack sack-period sack_period
            parameters sack sack-frequency sack_frequency
            parameters mtu-size mtu_size
            end
```

NOTES:

- **instance instance_id** *instance_id*—Specify the endpoint instance ID. Must be an integer in the range of 1-4.
- **endpoint** *endpoint_name*—Specify the endpoint as SCTP.
- **replicas** *number_of_nodes*—Specify the number of node replicas that must be configured for resiliency. The minimum or default value is 2.
- **parameters**—SCTP tuning parameters.
- **rto**—Retransmission timeout parameters.
- **association**—Association parameters.
- **client-idle-timeout**—Client idle timeout in seconds before SCTP shutdown. Setting it to 0 disables this timeout.

- **mtu-size** *mtu_size*—Specify the MTU size for the data packet. Must be an integer in the range of 512-9000. The default value is 1452.
- **sack**—Configures the way delayed SACKs are performed.
- **initial** *rto_initial*—Specify the initial timeout in milliseconds. Must be an integer in the range of 100-60000. The default value is 3000.
- **min** *rto_min*—Specify the minimum timeout in milliseconds. Must be an integer in the range of 100-60000. The default value is 1000.
- **max** *rto_max*—Specify the maximum timeout in milliseconds. Must be an integer in the range of 100-60000. The default value is 60000.
- **valid-cookie-life** *valid_cookie_life*—Specify the cookie life in milliseconds. Must be an integer in the range of 5000-120000. The default value is 60000.
- **heartbeat-interval** *heartbeat_interval*—Specify the heartbeat interval in milliseconds. Setting the value to zero, disables the heartbeat. Must be an integer in the range of 0-60000. The default value is 30000.
- **path-max-retry-count** *path_max_retry_count*—Specify the path maximum retry count. Must be an integer in the range of 0-20. The default value is 5.

Note When single path/IP address is available, **path-max-retry-count** parameter defines maximum number of retransmissions of the DATA packets before the address is marked unreachable by sending ABORT (when only single path).

When multiple paths/IP addresses are available, **path-max-retry-count** parameter is not applicable. The parameter checks only for DATA packet and not HEARTBEAT packet sent in SCTP.

It is recommended not to change **path-max-retry-count** parameter value and use the default value.

- **sack-period** *sack_period*—Specify the delayed sack time in milliseconds. Must be an integer in the range of 200-500. The default value is 200.

Note This parameter is effective when no data packets are flowing.

- When no data packets are flowing, separate SACK message is sent to acknowledge the data to the sender.
- If data packets are already flowing, then SACK is sent along with the data.

- **sack-frequency** *sack_frequency*—Specify the delayed SACK frequency. Must be an integer in the range of 1-5. The default value is 2.

Note Changing the frequency to 1, disables the delayed SACK algorithm.

This parameter is effective when no data packets are flowing.

- When no data packets are flowing, separate SACK message is sent with all TSNs in one SACK message to acknowledge the data to the sender.
- If data packets are already flowing, then SACK is sent along with the data.

Configuration Example

The following is an example configuration.

```
config
    instance 1
    endpoint sctp
        replicas 2
        parameters rto initial 30
        parameters rto min 10
        parameters rto max 600
        parameters association valid-cookie-life 60000
        parameters association heartbeat-interval 30000
        parameters association path-max-retry-count 5
        parameters sack sack-period 2000
        parameters sack sack-frequency 2
        parameters mtu-size 1500
        end
```

Configuration Verification

To verify the configuration:

```
show running-config instance
```