

Troubleshooting

- Using CLI Data, on page 1
- Logs, on page 4
- Frequently Encountered Scenarious, on page 6

Using CLI Data

This section describes the show and clear commands that are used for troubleshooting.

show subscriber

This section describes the **show subscriber** commands for the existing subscribers sessions.

Table 1: show subscriber Command Output Description

Field	Description
|	Output modifiers.
all	Displays all the existing subscriber sessions.
supi	Displays subscriber sessions based on SUPI ID.
gnodeb-id	Displays the gnodeb-id of the session.

clear subscriber

This section describes the **clear subscriber** commands for the existing subscribers sessions.

Table 2: clear subscriber Command Output Description

Field	Description
|	Output modifiers.
all	Clears all the subscriber sessions.

Field	Description
gnodeb-id	Clears the sessions that have the specified gnodeb-id.
supi	Clears the sessions based on the SUPI value.

Monitor Subscriber

Table 3: Feature History

Feature Name	Release Information	Description
Monitor Subscriber	2023.04	Cisco AMF supports the monsub to capture the N1/N2/N8/N11/N12/N15/N20/N22/N26 interface level messages. Default Setting: Disabled – Configuration Required

Feature Description

The "Monitor Subscriber" is a debugging and troubleshooting tool which captures the N1/N2/N8/N11/N12/N15/N20/N22/N26 interface level messages. The messages are logged only if SUPI is present or can be found from AMF database.

Note The monitor subscriber captures messages exchanged between the AMF and peer nodes only after identifying the subscriber's SUPI. Once it identifies the SUPI, it captures subsequent messages.

Example: When receiving an initial registration request with an encoded SUCI, the monitor subscriber (if enabled with SUPI) captures messages post-authentication after extracting the SUPI.

Configuring the Monitor Subscriber

Following are the various CLI options available for the monitor subscriber.

Option: 1

```
[amf-ops-center] amf# monitor subscriber supi imsi-123456789012345 capture-duration 200
internal-messages yes
```

With the preceding CLI option, both internal and external messages are logged for duration of 200 seconds. To explicitly record N1N2 messages, you must configure the 'internal-messages' option with the value "yes."

Option: 2

```
monitor subscriber supi imsi-123456789012345
```

With the preceding CLI option, only N26 and rest API messages are logged for a duration of 300 secs (default capture duration).

Option: 3

```
[amf-ops-center] amf# monitor subscriber supi imsi-123456789012345 capture-duration 50000
transaction-logs yes
logging transaction message enable.
```

With the preceding CLI option, transaction level messages are logged which are used for internal debugging.

Option: 4

```
[amf] amf# monitor subscriber supi imsi-123456789012345 capture-duration 3000
internal-messages yes file-name amf
```

With the preceding CLI option, MonSub file is generated with provided file-name in CLI.

Limitations

Following are the limitations for the monsub:

- If OAM pod restarts, the previously stored MonSub logs gets deleted.
- Enabling MonSub for a large number or all subscribers in a production environment impacts the system performance. So, it is recommended to enable the Monsub for few or specific subscribers.

Note

- The CLI option for enabling Monsub with imsi-* is not recommended in loaded system with bulk calls. As mentioned in the preceding section, specific SUPI (example - imsi-1234567890) should be used to capture the message logging with available options.
- In multi-node deployments, the source IP address displayed in the monitor-subscriber output may reflect the local kubernetes node IP rather than the peer NF's original IP. This is an expected result of standard Source Network Address Translation (SNAT) when traffic is routed between nodes to reach the pod. Traffic handled within the same node will continue to display the original peer NF IP.

Not Supported

The MonSub doesn't support the following.

- Messages related to Non-UE
- Monsub CLI (Monitor Subscriber IMSI) and (Monitor Subscriber IMEI)
- Messages towards Lawful Intercept (LI) interface
- All the SBI messages towards NRF
- N2 interface messages like NGSETUP, NGAP_ERROR_INDICATION and NG_RESET

Logs

Feature Description

AMF utilizes the common logging framework to generate logs from its microservices.

The supported log levels are:

- Error
- Warn
- Info
- Debug
- Trace

Note Warn level logging takes place during production.

Error

These errors are fatal errors, which can impact service for multiple subscribers.

Examples of the error messages:

- Node discovery of SBA fails after query from NRF and local configuration
- Mandatory IE missing in an NGAP message
- Memory cache startup errors
- Endpoint not found

Sample log:

```
[ERROR] [ApplicationContext.go:1820] [infra.dpd.core] Ping Unsuccessful for client Id 4
Name: amf-protocol-ep0 Setname: amf-protocol-ep Host: amf-protocol-ep Port: 9003 Url:  for
 [246]
```

Warn

These errors impact few specific call-flows majorly, but not blockers of functionality.

Example of the warning messages:

- Node discovery of SBA fails but we have more options to retry.
- Mandatory IE missing in a NAS message
- RPC timeout
- Procedural timeout

- Validation failure (not critical)

  Example: Registration rejected as Registration request message received registration type as the Reserved registration type.

- External entity sending unexpected or negative response

  Example: Handover Cancel, Hand over Failure, or Initial Context Setup Failure

- Unexpected value of objects maintained by AMF

  Example: NIL value of transaction

- Unable to fetch a subscriber

Sample log:

```
[WARN] [amf-service.amf-app.messageprocessor] No procedure defined for message type 763
```

Info

This log level purpose is to know information for cause.

Examples of the information messages:

- Procedural outcome Example: Disabling of ICSR for Registration
- Collision abort, cleanup, suspend, or continue.

Sample log:

```
[INFO] [amf-service.amf-app.auth] Sending N12 Authentication Request to Rest EP
```

Debug

This log level purpose is to get debug messages.

Example of the debug messages:

- All external exchanged messages
- Sending Registration accept to UE
- State machine changes
- Collision detailed logging

Sample log:

```
[DEBUG] [process.go:1606] [amf-service.amf-app.reg] [supi:123456789012345]
[supi:123456789012345] [1] Preparing registration accept to UE 123456789012345
```

Trace

This log level purpose is to get content of all external tracing messages.

Example of the trace messages:

- Registration request message
- N1N2 transfer message

Sample log:

```
[TRACE] [process.go:1627] [amf-service.amf-app.reg] [supi:123456789012345]
[supi:123456789012345]
[496] Sending RegistrationAccept:&MsgNas
{N1MsgType:154,N2MsgType:0,N1Msg:&MsgNas_MsgRegistrationAccept
{MsgRegistrationAccept:&ngn_nas.PBRegistrationAccept{ExtendedProtocolDiscriminator:126,SecurityHeaderType:
&SecurityHeaderType{HeaderType:PLAIN_5G_NAS,},MessageIdentity:&MessageType{MessageType:REGISTRATION_ACCEPT,}
,VgsRegistrationResult:&VgsRegistrationResult{EmergencyRegistered:false,NssaaPerformed:false,SmsAllowed:false,
VgsRegistrationResultValue:TGPP_ACCESS,}
```

How it Works

This section describes how this feature works.

Log Tags

Use log tags to tag the logs for specific procedures which are part of a flow or an event. Enabling of AMF logging takes place at different log levels for different log tags.

Name	Purpose	Example Log tags
AMF service	To capture procedures.	• LogTagReg • LogTagPDU, and so on
Protocol Endpoint	To capture on the interface.	• LogTagNas • LogTagNgap • LogTagNonUE
Rest Endpoint	To capture on the interface.	• LogTagN11 • LogTagN14 • LogTagNRF • LogTagN11OrN14 (N1NMsgTransfer can come from N14/N11 interfaces) and so on

Frequently Encountered Scenarious

Geo-Replication Pod in Pending State

This section describes how to correct geo-replication pod conflict if shared hardware setup.

Problem

After completing Day1 configuration on AMF, when you deploy AMF and SMF on the same mode, the geo-replication pod is in pending state.

The following table lists the ports configured use by a geo-replication pod. The port numbers are for reference purpose only.

Note The default base port is 15000. You can change the default base port.

Table 4: Ports Configured for Geo-replication Pod

15000	INFRA_PROMETHEUS_PORT
15001	PPROF_EP_PORT
15002	INFRA_ADMIN_PORT
15003	IPC_EP_PORT
15004	GEO_KEEPALIVED_PORT
15005	INFRA_DIAG_PORT

Resolution

1. Change the default base port for geo-pod from 15000 to other available port range.

   instance instance-id *<instance_id>* **endpoint geo internal base-port start** *<new_port>*

Note <*instance_id*> should match the <*local_instance_id*>.

Configure the relevant keepalive port in the SMI configuration (base_port + 4) .

This configuration is required only for the GR setup.

2. To verify that the new port change configuration is reflecting, run the following command.

   ```
   kubectl describe pod georeplication-pod-0 -n cn | grep -i port
   ```

3. SSH to the server where geo-pod is running and run the following command.

   ```
   sudo netstat -plan | grep grpod | grep <port_range> | grep -v
   ```