Monitor Subscriber and Monitor Protocol

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	SMF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Always-on
Related Changes in this Release	Not Applicable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	Pre-2020.02.0

Feature Description

The SMF supports the Monitor Subscriber and Monitor Protocol on the Kubernetes environment. This feature allows to capture messages of subscribers and protocols.

Configuring the Monitor Subscriber and Monitor Protocol Feature

Monitoring the Subscriber

Use the following CLI command to monitor the subscriber in the SMF.

monitor subscriber supi *supi_id* **[capture-duration** *duration_sec* **| internal-messages { yes } | transaction-logs { yes }]**

NOTES:

- **supi** *supi_id*: Specifies the subscriber identifier. For example, imsi-123456789, imsi-123*
- **capture-duration** *duration_sec*: Specifies the duration in seconds during which the monitor subscriber is enabled. The default is 300 seconds (5 minutes).
- **internal-messages { yes }**: Enables internal messages. By default, internal messages are disabled.
- **transaction-logs { yes }**: Enables transaction logs. By default, transaction logs are disabled.

The **monitor subscriber** CLI command can be run simultaneously on multiple terminals. For example, run the CLI simultaneously in two SMF Ops Center terminals for two subscribers (for example, imsi-123456789012345 and imsi-456780123456789) to implement the following:

- Monitor the duration when the monitor subscriber is enabled.
- View internal messages for the specified subscriber.
- View transaction logs for the specified subscriber

Terminal 1: The following command monitors and displays subscriber messages for the specified subscriber.

```
monitor subscriber supi imsi-123456789012345 capture-duration 1000 internal-messages yes
```

Terminal 2: The following command monitors and displays transaction logs for the specified subscriber.

```
monitor subscriber supi imsi-456780123456789 capture-duration 500 internal-messages yes
transaction-logs yes
```

After the capture-duration is over or to stop the CLI, use the **Ctrl+C** keys. The captured messages are reordered and stored in a file. To retrieve the list of stored files, use the **monitor subscriber list** CLI command.

For example:

```
monitor subscriber list
RELEASE_NAMESPACE: 'smf'
'monsublogs/subscriberID_imsi-*_AT_2019-10-22T09:19:05.586237087.txt.sorted'
monsublogs/subscriberID_imsi-123456789012345_AT_2019-10-22T09:20:11.122225534.txt.sorted
```

Viewing the Sorted File on SMF Ops Center

Use the following CLI command to view the sorted file on the SMF Ops Center screen.

monitor subscriber dump filename *filename*

For example:

```
monitor subscriber dump filename
monsublogs/subscriberID_imsi-123456789012345_AT_2019-10-22T09:20:11.122225534.txt.sorted
```

Monitoring the Interface Protocol

Use the following CLI command to monitor the interface protocol on the SMF.

monitor protocol interface *endpoint_name* **capture-duration** *duration_sec*

NOTES:

- **interface** *endpoint_name*: Specifies the endpoint name on which PCAP is captured. This CLI allows the configuration of multiple endpoint names in a single CLI command.
- **capture-duration** *duration_sec*: Specifies the duration in seconds during which the monitor subscriber is enabled. The default is 300 seconds (5 minutes).
- The configured endpoint names can be retrieved using the **show endpoint** CLI command.

The **monitor protocol** CLI can be run simultaneously on multiple terminals. Also, the **interface** *endpoint_name* CLI allows the configuration of multiple endpoint names in a single CLI command. For example:

monitor protocol interface sbi,N4:10.86.73.161:8805,gtpc capture-duration 1000

Viewing Transaction History Logs

Use the following CLI command to view the transaction history on an OAM pod shell. On another terminal, use the **kubectl** command to tail the logs of the OAM pod and then run the following CLI from the Ops Center.

dump transactionhistory

NOTES:

In this release, the most recent transaction logs are stored in a circular queue of size 1024 transaction logs.

Sample Transaction Log

The following is a sample transaction log.

```
InstanceInfo: SMF.smf-service.DC.Local.0
TimeStamp: 2019-11-13 03:01:33.614095848 +0000 UTC
************* TRANSACTION: 00091 *************
TRANSACTION SUCCESS:
    Txn Type             : 24
    Priority             : 1
    Session State        : Update_Session
    Subscriber Id        : imsi-123456789012345
    Session Keys         : imsi-123456789012345:5 (primary)
LOG MESSAGES:
    2019/11/13 03:01:33.565 [INFO] [infra.application.core] Queue data 91
    2019/11/13 03:01:33.565 [DEBUG] [infra.transaction.core] Processing transaction Id: 91
 Type: 24 SubscriberID:  Keys: []
    2019/11/13 03:01:33.565 [TRACE] [infra.message_log.core] >>>>>>>
IPC message
Name: N11SmContextReleaseReq
```

```
MessageType: N11SmContextReleaseReq
Key:
--body--
{"smcontextreleasedata":{"cause":7}}
    2019/11/13 03:01:33.566 [DEBUG] [infra.transaction.core] Trying to load session
    2019/11/13 03:01:33.566 [DEBUG] [infra.session_cache.core] Get session by pk
imsi-123456789012345:5
    2019/11/13 03:01:33.566 [DEBUG] [infra.session_cache.core] Record found in local cache
 by key imsi-123456789012345:5
    2019/11/13 03:01:33.566 [DEBUG] [infra.transaction.core] Queuing new transaction for
processing
    2019/11/13 03:01:33.566 [DEBUG] [smf-service0.smf-app.messageprocessor] GetLockPriority
 for txn id: 91, Type: 24
    2019/11/13 03:01:33.566 [DEBUG] [infra.transaction.core] Session lock priority 0 txn
lock priority 10
    2019/11/13 03:01:33.567 [DEBUG] [infra.transaction.core] Session locked with priority
10
    2019/11/13 03:01:33.567 [DEBUG] [smf-service0.smf-app.gen] Handle Idle Events
    2019/11/13 03:01:33.567 [DEBUG] [smf-service0.smf-app.amf] Handling SM Context Release
 Event
    2019/11/13 03:01:33.567 [DEBUG] [smf-service0.smf-app.upf] Send N4 Release Request
    2019/11/13 03:01:33.567 [DEBUG] [infra.transaction.core] Requested host Setname:
smf-protocol Name:  Version:  ApiRoot:
    2019/11/13 03:01:33.567 [DEBUG] [infra.transaction.core] Selected remote host by set
name is Id 5 Name: smf-protocol4 Setname: smf-protocol Host: smf-protocol Port: 9003 Url:
 of available 10 hosts
    2019/11/13 03:01:33.567 [INFO] [infra.transaction.core] Calling RPC smf-protocol on
host smf-protocol4 proc-name smf-protocol proc-method: Sync
    2019/11/13 03:01:33.567 [DEBUG] [infra.ipc_action.core] Calling IPC RPC with Retry,
Retry Counter is 3
    2019/11/13 03:01:33.597 [DEBUG] [infra.ipc_action.core] Time taken to execute IPC is
0.03
    2019/11/13 03:01:33.597 [DEBUG] [infra.ipc_action.core] Destination Host 192.168.1.163
 serviced the IPC Message N4SessionReleaseReq
    2019/11/13 03:01:33.597 [DEBUG] [smf-service0.smf-app.upf] UPF N4 Session Release done
    2019/11/13 03:01:33.597 [DEBUG] [smf-service0.smf-app.messageprocessor] Returning
newStage (RELEASE: Idle)
    2019/11/13 03:01:33.597 [DEBUG] [infra.transaction.core] Last stage ( init_done ) ->
Next stage ( RELEASE: Idle )
    2019/11/13 03:01:33.598 [DEBUG] [smf-service0.smf-app.upf] Processing N4 Response
awtUpfRelProcUpfReleaseResp
    2019/11/13 03:01:33.599 [DEBUG] [infra.transaction.core] Requested host Setname:  Name:
 192.168.2.150 Version:  ApiRoot:
    2019/11/13 03:01:33.599 [DEBUG] [infra.transaction.core] Exact match found. Selected
remote host is Id 11 Name: 192.168.2.150 Setname:  Host: 192.168.2.150 Port: 9003 Url:
    2019/11/13 03:01:33.599 [INFO] [infra.transaction.core] Calling RPC smf-nodemgr on host
 192.168.2.150 proc-name smf-nodemgr proc-method: Sync
    2019/11/13 03:01:33.600 [DEBUG] [infra.ipc_action.core] Calling IPC RPC with Retry,
Retry Counter is 3
    2019/11/13 03:01:33.609 [DEBUG] [infra.ipc_action.core] Time taken to execute IPC is
0.01
    2019/11/13 03:01:33.609 [DEBUG] [infra.ipc_action.core] Destination Host 192.168.2.150
 serviced the IPC Message NmgrRersourceMgmtRequest
    2019/11/13 03:01:33.609 [DEBUG] [smf-service0.smf-app.resource] NodeMgr Resource(IP and
 commonID) Release Sent
    2019/11/13 03:01:33.609 [DEBUG] [smf-service0.smf-app.messageprocessor] Returning
newStage (RELEASE: Await UPF Release)
    2019/11/13 03:01:33.609 [DEBUG] [infra.transaction.core] Last stage ( RELEASE: Idle )
-> Next stage ( RELEASE: Await UPF Release )
    2019/11/13 03:01:33.609 [DEBUG] [smf-service0.smf-app.resource] Processing Rmgr Response
 awtRmgrRelProcNmgrResourceMgmtRsp
    2019/11/13 03:01:33.609 [DEBUG] [smf-service0.smf-app.resource] NodeMgr Resource(IP and
 commonID) Release done
    2019/11/13 03:01:33.610 [DEBUG] [smf-service0.smf-app.amf] Sending Sm context Release
```

```
Response
    2019/11/13 03:01:33.610 [DEBUG] [smf-service0.smf-app.messageprocessor] Returning
newStage (RELEASE: Await Resource Release)
    2019/11/13 03:01:33.610 [DEBUG] [infra.transaction.core] Last stage ( RELEASE: Await
UPF Release ) -> Next stage ( RELEASE: Await Resource Release )
    2019/11/13 03:01:33.610 [DEBUG] [smf-service0.smf-app.messageprocessor] Returning
newStage (finished)
    2019/11/13 03:01:33.610 [DEBUG] [infra.transaction.core] Last stage ( RELEASE: Await
Resource Release ) -> Next stage ( finished )
    2019/11/13 03:01:33.610 [DEBUG] [infra.transaction.core] Updating session
    2019/11/13 03:01:33.610 [DEBUG] [infra.session_cache.core] Save session with key
imsi-123456789012345:5 in cache
    2019/11/13 03:01:33.612 [DEBUG] [infra.session_cache.core] Queued datastore write for
key imsi-123456789012345:5
    2019/11/13 03:01:33.613 [TRACE] [infra.message_log.core] <<<<<<<<
    2019/11/13 03:01:33.613 [DEBUG] [infra.transaction.core] sent response message for 91
```