Performance Optimization Support

- Feature Summary and Revision History, on page 1
- Feature Description, on page 4
- Async BG-IPC from GTPC-EP towards SGW-Service, on page 5
- Batch ID Allocation, Release, and Reconciliation Support, on page 5
- Cache Pod Optimization, on page 8
- CDL Flush Interval and Session Expiration Tuning Configuration, on page 8
- DDN Call Flow Optimization, on page 9
- DDN Timeout Configuration, on page 14
- Domain-based User Authorization Using Ops Center, on page 15
- Edge Echo Implementation, on page 17
- ETCD Peer Optimization Support, on page 19
- Optimized GTPv2 Encoder and Decoder, on page 20
- GTPC Endpoint with GR Split, on page 22
- GTPC Endpoint Interface Split with S11 and S5 , on page 23
- GTPC IPC Cross-rack Support, on page 25
- Interservice Pod Communication, on page 33
- MBR Call Flow Optimization, on page 36
- Maintenance Mode, on page 45
- Partial CDL Update for Idle-Active Call Flow, on page 47
- PFCP Session Report with DLDR Throttling Support, on page 49
- Resiliency Handling, on page 52
- Roaming Peer Path Management Optimization, on page 55
- Flag DB Database Updates, on page 59
- UDP Proxy Functionality Merged into Protocol Microservices, on page 60

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Products or Functional Area	cnSGW-C

Applicable Platforms	SMI
Feature Default Setting	Async BG IPC from GTPC-EP towards SGW-Service: Enabled – Always-on Batch ID Allocation, Release, Reconciliation Support: Disabled – Configuration required to enable Cache Pod Optimization CDL Flush Interval and Session Expiration Tuning Configuration: Enabled – Configuration required to disable DDN Call Flow Optimization: Disabled – Configuration required to enable DDN Timeout Configuration: Disabled – Configuration required to enable Edge Echo Implementation: Enabled – Always-on ETCD Peer Optimization Support: Enabled - Always-on Flag the DB Database Updates: Enabled – Always-on GTPC Interface Split: Disabled – Configuration required to enable GTPC IPC Cross-rack Support: Disabled – Configuration required to enable Interservice Pod Communication: Disabled – Configuration required to enable Maintenance Mode: Disabled – Configuration required to enable MBR Call Flow Optimization: Disabled – Configuration required to enable Optimized GTPv2 Encoder and Decoder: Enabled – Always-on Partial CDL Update for Idle Active Call Flow: Enabled – Always-on PFCP Session Report with DLDR Throttling Support: Disabled – Configuration required to enable Resiliency Handling: Disabled – Configuration required to enable Roaming Peer Path Management Optimization: Disabled – Configuration required to enable UDP Proxy functionality merge into Protocol microservices: Enabled – Configuration required to disable

Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
Support for the following sub-features were introduced: • Added support for cache pod optimization. • PFCP Session Report with DLDR Throttling Support • Resiliency Handling	2023.01.0

Revision Details	Release
Support for the following sub-features were introduced: • Batch ID Allocation, Release, and Reconciliation • CDL Flush Interval and Session Expiration Tuning Configuration • DDN Call Flow Optimization • DDN Timeout Configuration • Edge Echo Implementation • ETCD Peer Optimization Support • Flag the DB Database Updates • GR Split • GTPC Interface Split • GTPC IPC Cross-rack Support • Interservice Pod Communication • Introduced support for IPv6. • Maintenance Mode • Optimization of Modify Bearer Request and Response (MBR) call flows • Optimized GTPv2 Encoder and Decoder is provided for additional Request and Response messages. • UDP Proxy and GTPC-EP Merge • UDP Proxy and PFCP-EP Merge	2022.04.0
First introduced.	2021.02.3

Feature Description

This chapter describes about the performance optimization features.

Some of the performance optimization features are common across cnSGW-C and SMF.

For complete information on SMF features, see the *UCC 5G SMF Configuration and Administration Guide*.

Async BG-IPC from GTPC-EP towards SGW-Service

Feature Description

cnSGW-C supports Asynchronous BG-IPC call from GTPC-EP to cnSGW-C, by consuming less resources of the GTPC-EP pod.

For Async BG-IPC call, the GTPC-EP pod uses:

- SendRequestWithCallbackAsResponseWithRequestId API of app-infra to send request messages to the SGW-service.
- GetIPCRequestCallbackResponseWithRequestId API to receive response from the SGW-service.

Batch ID Allocation, Release, and Reconciliation Support

Feature Description

The nodemgr allocates a unique ID to the subscriber that is in the attached state. When the subscriber detaches, the unique ID is released to the nodemgr. If the allocation and deallocation procedures increase, the nodemgr performance is impacted and the sgw-service continues to wait longer to complete these procedures.

The Batch ID Allocation, Release, and Reconciliation Support feature provide a mechanism to reduce the interaction between the sgw-service and nodemgr, which in turn optimizes the nodemgr's performance.

How it Works

This section describes how this feature works.

Batch ID Allocation:

Allocation of batch ID involves the following steps:

- The sgw-service manages the ID store by allocating the free IDs to the subscriber. When the IDs are unavailable in the store, the sgw-service sends the Batch ID Allocation Request to nodemgr.
- In response, nodemgr returns a batch of 128 IDs with the ID Reserve Report Interval. The sgw-service updates the ID store with the IDs received from the nodemgr and starts a timer for the ID Reserve Report Interval.
- If all the IDs are used before the duration configured in the ID Reserve Report Interval, sgw-service sends a Batch of ID Allocation Request to the nodemgr with a notification to reserve all IDs from the previous request.
- If the ID Reserve Report Interval timer expires before the sgw-service allocates all the IDs, sgw-service sends the unused IDs back to nodemgr through the Reserve Report Batch operation.

Batch ID Release:

Releasing of the batch ID involves the following steps:

- The sgw-service manages the IDs that the ID store releases for each nodemgr.
- The sgw-service returns the ID to the ID store whenever an ID is deallocated. If the ID store is full, the sgw-service sends a Batch ID Release Request and the released IDs to the respective nodemgr.
- When sgw-service starts adding IDs to the ID store, the ID release timer starts.
- If the ID release timer expires before the batch IDs are releases or the batch is full, sgw-service sends the released IDs to nodemgr.

Batch ID Reconciliation

Batch ID reconciliation occurs when the service pod and the nodemgr pod restarts.

On service pod restart:

1. When the service pod receives the batch IDs and becomes unresponsive before allocating the IDs, the nodemgr does not get the Batch ID Reserve Request causing the ID reserve procedure to time out. In such a scenario, the nodemgr reconciles the unreserved or unallocated IDs with CDL. The IDs that are not allocated to the subscribers are released to the ID store.
2. The service pod collects the IDs that are released and if it becomes unresponsive before releasing them to the nodemgr. In this scenario, the IDs are dropped.

On nodemgr pod restart:

1. The IDs existing in the in-flight Batch ID Reserve Request and Batch ID Release Request are dropped.
2. The nodemgr notifies cachemgr about the allocated IDs in a batch. If nodemgr becomes unresponsive before notifying the IDs to cachemgr, after a restart, nodemgr starts allocating the new IDs. The nodemgr allocated the IDs based on the last allocated ID and the batch size.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   sgw sgw_name
      resmgr-batch-operation [ disable | enable ]
      end
```

NOTES:

resmgr-batch-operation [disable | enable]—Configures the batch operation. By default, **resmgr-batch-operation** is disabled.

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics

The following statistics are supported for the Batch ID Allocation and Release Support feature:

- sgw_resource_mgmt_stats—Captures the total number of the cnSGW-C resource management statistics.

Sample queries:

```
sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",
id_req_type="id_batch_alloc",instance_id="0",service_name="sgw-service",status="attempted"}
 3
:sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",
id_req_type="id_batch_alloc",instance_id="0",service_name="sgw-service",status="success"}
 3
```

```
sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1"
,id_req_type="id_batch_dealloc",instance_id="0",service_name="sgw-service",status="attempted"}
 2
sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",id_req_type=
"id_batch_dealloc",instance_id="0",service_name="sgw-service",status="success"}
 2
```

```
sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",
id_req_type="id_batch_dealloc_timeout",instance_id="0",service_name="sgw-service",status="attempted"}
 1
:sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",id_req_type
="id_batch_dealloc_timeout",instance_id="0",service_name="sgw-service",status="success"}
 1
```

```
sgw_resource_mgmt_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",
id_req_type="id_batch_release_timeout",instance_id="0",service_name="sgw-service",status="attempted"}
 1
-:sgw_resource_mgmt_stats{app_name="smf",cluster="Local",d-ata_center="DC",gr_instance_id="1",id_req_type
="id_batch_release_timeout",instance_id="0",service_name="sgw-service",status="success"}
 1
```

- nodemgr_rmgr_batch_reconcile_stats—Captures the total count of batches that are sent for reconciliation.

  Sample queries:

```
nodemgr_rmgr_batch_reconcile_stats{app_name="smf",cluster="Local",data_center="DC",instance_id="0",
service_name="nodemgr",status="success"} 1
```

- nodemgr_resource_mgmt_resp_stats—Captures the total number of IDs released due to reconciliation.

  Sample queries:

```
nodemgr_resource_mgmt_resp_stats{app_name="smf",cluster="Local",data_center="DC",error="",
gr_instance_id="0",instance_id="0",ip_ver_type="IP_TYPE_NONE",req_type="ID_REQ_REL_RECONCILE",
service_name="nodemgr",status="success"} 16
```

For more information on bulk statistics support, see *UCC Serving Gateway Control Plane Function Metrics Reference*.

Cache Pod Optimization

Feature Description

The cnSGW-C supports the cache pod optimization to reduce the cache pod query at the GTPC endpoint.

The get affinity query is used to receive the affinity information in an outgoing request or response message toward the GTPC endpoint. With this optimization, the GTPC endpoint pod doesn't send the query to the cache pod for the upcoming request messages.

To receive this affinity information, an affinity query is used in an outgoing request or response message toward the GTPC endpoint. With this optimization, the GTPC endpoint pod doesn't send the query to the cache pod for the upcoming request messages.

In the previous releases, after the cnSGW-C sent out the DDN and received the MBR from the MME, the GTPC endpoint had to send the query to the cache pod to get affinity information. Later, the cnSGW-C used the affinity information so that an MBR can be forwarded to the correct service pod.

With this optimization, you can prevent the extra cache pod query.

CDL Flush Interval and Session Expiration Tuning Configuration

Feature Description

You can modify the default service-pod parameters to fine-tune the throughput performance and optimize the load performance.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   profile sgw sgw_name
      timers [ session-expiration-in-secs session_expiration |
affinity-expiration-in-secs affinity_expiration | session-dbsync-interval-in-ms
 database_sync ]
      end
```

NOTES:

- **session-expiration-in-secs** *session_expiration* —Specify the duration for which the session is cached on service pod. *session_expiration* accepts value in the range of 1-600 milliseconds. The default value is 30 milliseconds.
- **affinity-expiration-in-secs** *affinity_expiration* —Specify the duration for which the session affinity keys are valid on the service pod and other pods. *affinity_expiration* accepts value in the range of 1-1200 seconds. The default value is 80 seconds.

- **session-dbsync-interval-in-ms** *database_sync* —Specify the duration after which the session is synchronized in the database. *database_sync* accepts value in the range of 1-10000 milliseconds. The default value is 500 milliseconds.

Configuration Example

The following is an example configuration.

```
config
     profile sgw sgw1 [ timers session-expiration-in-secs 30 | affinity-expiration-in-secs
 80 | timers session-dbsync-interval-in-ms 500 ]
     end
```

DDN Call Flow Optimization

Feature Description

The Downlink Data Notification (DDN) Call Flow Optimization feature lets you suspend the invoking of the DDN procedure for a specified period. With this feature, the network-initiated service request procedure is scheduled before the DDN is received and the UE is moved to the active state. In case of the timer expiry, if the UE ID is active, or the Modify Bearer Request is in progress, the cnSGW ignores the DDN procedure.

How it Works

This section describes how this feature works.

cnSGW-C invokes the DDN procedure in the following scenarios:

- A session rapidly moves to the idle state causing the Uplink and Downlink data to trigger simultaneously.
- When a subscriber is in the idle state and the Downlink data is received, cnSGW-C sends a DDN to MME. The DDN notifies MME to page the UE and change the UE state to active.
- When a subscriber is in the idle state and the Uplink data is received, the MME sends a Modify Bearer Request to SGW to change the UE state to active.
- If the SGW service has initiated DNN and it receives the Modify Bearer Request, the service aborts the DDN procedure and processes the Modify Bearer Request to change the UE state to active.

Call Flows

This section describes the key call flows for this feature.

Current Downlink Data Notification Handling Call Flow

This section describes the Current Downlink Data Notification Handling call flow.

Figure 1: Current Downlink Data Notification Handling Call Flow

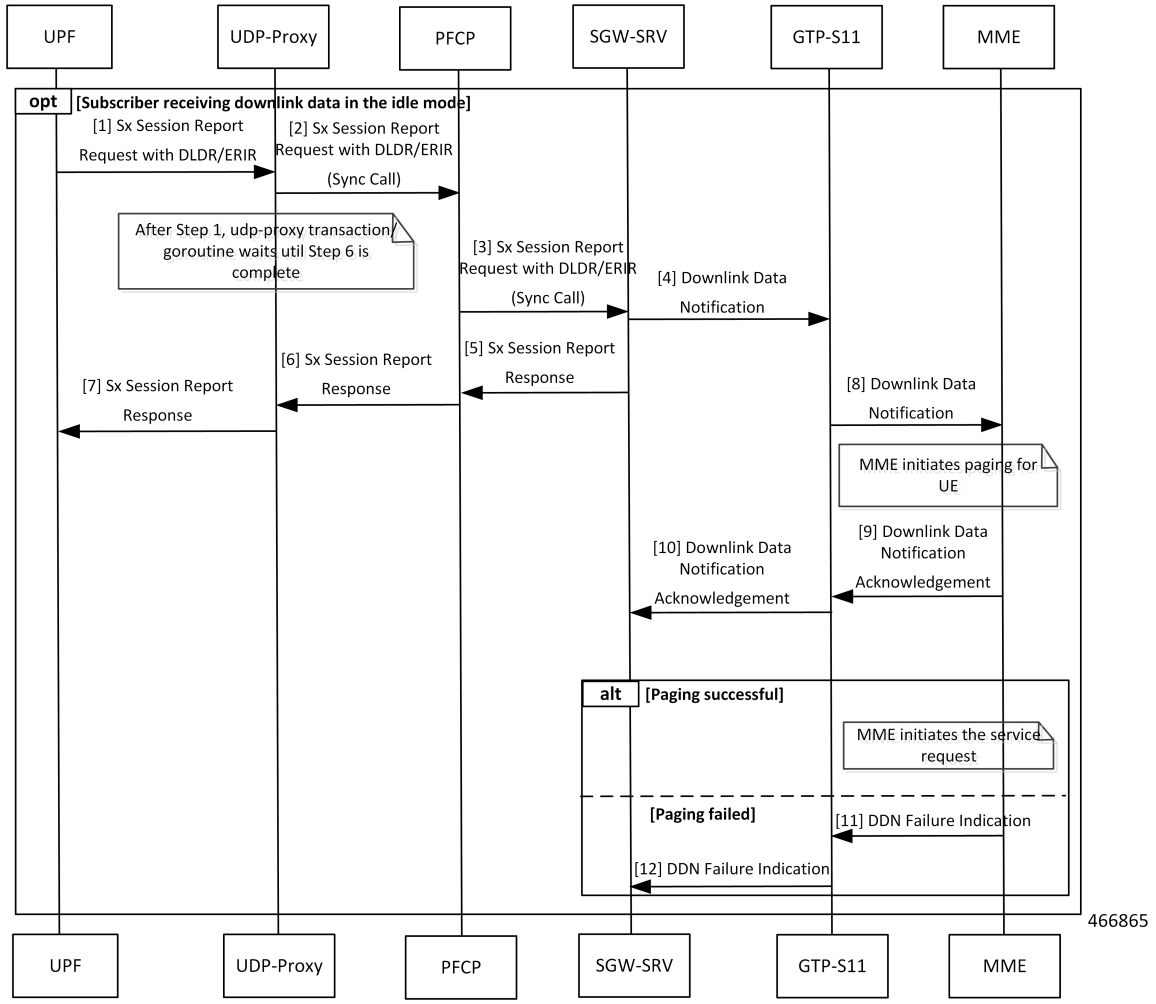


Table 3: Current Downlink Data Notification Handling Call Flow Description

Step	Description
1	UPF sends a Sx Session Report Request with DLDR or ERIR to UDP-Proxy.
2	UDP-Proxy sends the Sx Session Report Request with DLDR or ERIR to PFCP.
3	PFCP sends the Sx Session Report Request with DLDR or ERIR to SGW-SRV.
4	The SGW-SRV sends the Downlink Data Notification (DDN) to GTP-S11.
5	SGW-SRV sends Sx Session Report Response to PFCP.
6	PFCP sends the Sx Session Report Response to UDP-Proxy.
7	UDP-Proxy sends the Sx Session Report Response to UPF.
8	GTP-S11 sends the DDN Notification to MME.

Step	Description
9	MME initiates a paging for UE and sends the DDN Acknowledgment to GTP-S11.
10	GTP-S11 sends the DDN Acknowledgment to SGW-SRV.
11	When the paging fails, MME sends the DDN Failure Indication to GTP-S11.
12	GTP-S11 sends the DDN Failure Indication to SGW-SRV.

DDN Handling with Internal DDN Delay Timer Call Flow

This section describes the DDN Handling with Internal DDN Delay Timer call flow.

Figure 2: DDN Handling with Internal DDN Delay Timer Call Flow

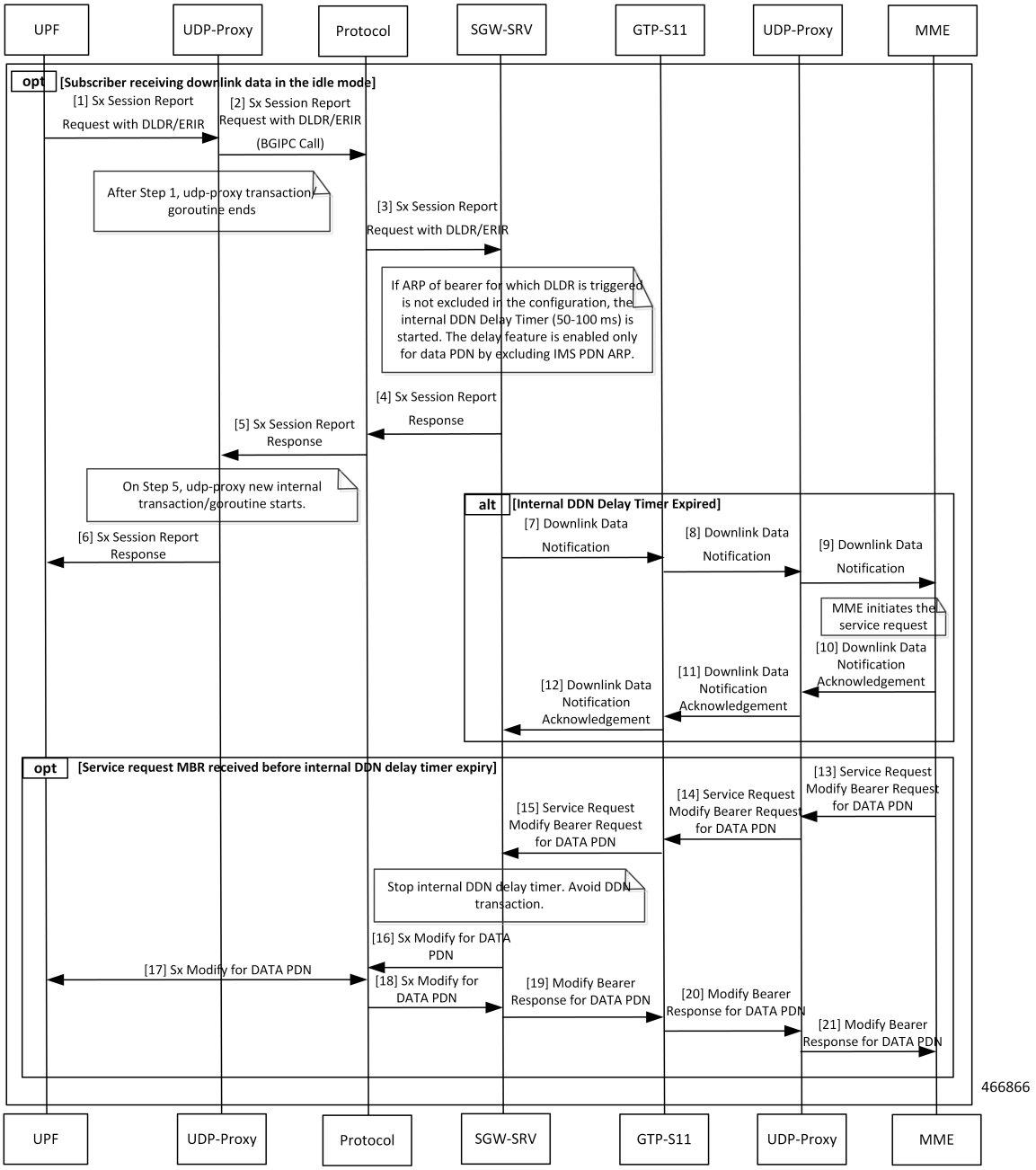


Table 4: DDN Handling with Internal DDN Delay Timer Call Flow Description

Step	Description
1	UPF sends a Sx Session Report Request with DLDR/ERIR to UDP-Proxy.
2	UDP-Proxy sends the Sx Session Report Request with DLDR/ERIR (BGIPC Call) to Protocol.
3	Protocol sends the Sx Session Report Request with DLDR/ERIR to SGW-SRV.

Step	Description
4	SGW-SRV sends a Sx Session Report Response to Protocol.
5	Protocol sends the Sx Session Report Request to UDP-Proxy.
6	UDP-Proxy sends the Sx Session Report Request to UPF.
7	SGW-SRV sends the Downlink Data Notification to GTP-S11.
8	GTP-S11 sends the Downlink Data Notification to UDP-Proxy.
9	UDP-Proxy sends the Downlink Data Notification to MME.
10	After MME initaites paging for UE, MME sends the Downlink Data Notification Acknowledgement to UDP-Proxy.
11	UDP-Proxy sends the Downlink Data Notification Acknowledgement to GTP-S11.
12	GTP-S11 sends the Downlink Data Notification Acknowledgement to SGW-SRV.
13	MME sends the Modify Bearer Request for DATA PDN to UDP-Proxy.
14	UDP-Proxy sends the Modify Bearer Request for DATA PDN to GTP-S11.
15	GTP-S11 sends the Modify Bearer Request for DATA PDN to SGW-SRV.
16	SGW-SRV sends the Modify Bearer Request for DATA PDN to Protocol.
17	Protocol and UPF processes the Sx Modify Bearer Request for DATA PDN.
18	Protocol sends Sx Modify Response for DATA PDN to SGW-SRV.
19	SGW-SRV sends Sx Modify Response for DATA PDN to GTP-S11.
20	GTP-S11 sends Sx Modify Response for DATA PDN to UDP-Proxy.
21	UDP-Proxy sends Sx Modify Response for DATA PDN to MME.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   profile sgw sgw_name
      ddn { delay-exclude-arplist number_priorities | delay-timer delay_duration
 }
       end
```

NOTES:

- **delay-exclude-arplist** *number_priorities*—Specify the priority-level for allocation and retention priorities [1-15] that must be excluded from delaying the DDN. *number_priorities* can accept a maximum of eight entries.

- **delay-timer** *delay_duration*—Specify the duration for which the DDN procedure is delayed. *delay_duration* accepts duration in milliseconds 0–5000. The default duration is 0 which indicates that the timer is disabled.

Configuration Example

The following is an example configuration.

```
config
    profile sgw sgw1
       ddn delay-timer 100 delay-exclude-arplist [ 3 4 ]
       end
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics

The following statistics are supported for the DDN Call Flow Optimization feature:

sgw_tmr_stats—The internal DDN delay timer for stop, start, and expired states.

Query:

```
sgw_tmr_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",
instance_id="0",service_name="sgw-service",status="expired",timer_type="internal_ddn_delay"}
1

sgw_tmr_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",instance_id="0",
service_name="sgw-service",status="start",timer_type="internal_ddn_delay"} 2

sgw_tmr_stats{app_name="smf",cluster="Local",data_center="DC",gr_instance_id="1",instance_id="0",
service_name="sgw-service",status="stop",timer_type="internal_ddn_delay"} 1
```

DDN Timeout Configuration

Feature Description

cnSGW-C lets you configure the DDN Timeout and Peer Not Responding configuration through the cnSGW-C Ops Center.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   profile sgw sgw_name
      ddn timeout-purge-session { true | false }
      end
```

NOTES:

ddn timeout-purge-session { true | false }—Configures the session when the MME does not send the DDN acknowledgment. The default value is false.

Domain-based User Authorization Using Ops Center

Feature Description

SMF and cnSGW-C support domain-based user authorization using the Ops Center. To control the access on a per-user basis, use the TACACS protocol in Ops Center AAA. This protocol provides centralized validation of users who attempt to gain access to a router or NAS.

Configure the NETCONF Access Control (NACM) rules in the rule list. Then, map these rules in the Ops center configuration to map the group to appropriate operational authorization. Use the configurations that are based on the following criteria and products:

- With the NACM rules and SMF domain-based group, configure the Ops center to allow only access or update SMF-based configuration.
- With the NACM rules and cSGW-C domain-based group, configure the Ops center to allow only access or update cSGW-C-based configuration.
- With the NACM rules and cSGW-C domain-based group, configure the Ops center to allow only access or update CCG-based configuration.

Note The NSO service account can access the entire configuration.

How it Works

To support this feature configuration in Ops Center, the domain-based-services configuration is added in the TACACS security configuration. The TACACS flow change works in the following way:

- If you have configured the **domain-based-services** parameter, then the configured user name that is sent to the TACACS process, splits user ID into user ID and domain. The split character, which is a domain delimiter, is configured in domain-based-services. These split characters can be "@", "/", or "\" and are used in the following format to get the domain and user ID information.
  - @ — <user id>@<domain>
  - / — <domain>/<user id>
  - \ — <domain>\<user id>
- The TACACS authenticates and authorizes as per the existing flow. However, if the domain-based-services feature is enabled and TACACS authenticates and authorizes the user, following steps are added to the TACACS flow procedure.
  - If Network Services Orchestrator (NSO) logs in as the NSO service account, then that session receives a specific NACM group that you configured in **domain-based-services nso-service-account group** *group-name*. This functionally is the same as the way NSO works.

- If the specified domain exists in the group mapping, then the NACM group that you configured in **domain-based-services domain-service** *domain* **group** *group-name* is applied.
- If the user does not have a domain or the domain does not exist in the domain to group mapping, then **no-domain** NACM group that you configured in **domain-based-services no-domain group** *group-name* is applied. If the **no-domain** configuration does not exist, then the user value is rejected.

To enable this feature, you must configure the **domain-based-services** CLI command with the following options:

- NSO service account
- Domain service
- Domain delimiter
- No domain

Feature Configuration

To enable domain-based user authorization using Ops Center, use the following sample configuration:

```
config
   tacacs-security domain-based-services [ domain-delimiter delimiter_option
 | domain-service domain_service_name [ group service_group_name  ] | no-domain
group service_group_name | nso-service-account [ group service_group_name | id
service_account_id ] ]
   end
```

NOTES:

- **domain-based-services [domain-delimiter** *delimiter_option* **| domain-service** *domain_service_name* **[group** *service_group_name* **] | no-domain group** *service_group_name* **| nso-service-account [group** *service_group_name* **| id** *service_account_id* **]]**: Configure the required domain-based-services value. The **domain-based-services** includes the following options:
  - **domain-delimiter**: Specify the delimiter to use to determine domain. This option is mandatory and allows the following values:
    - @—If domain-delimiter is "@", the user value is in the format: <user>@<domain>.
    - /—If domain-delimiter is "/", the user value is in the format: <domain>/<user>.
    - \—If domain-delimiter is "\", the user value is in the format: <domain>\<user>.
  - **domain-service**: Specify the list of domains and their group mapping. The key is the name of the domain and group is the group that is assigned to the domain. You must configure at least one option in this list.
  - **no-domain**: Specify the group that has no domain or if the domain is unavailable in the domain-service mapping, then this group is sent in the accept response.
  - **nso-service-account**: Specify the NSO service account that has the ID and group. If you configure this parameter, then you must configure the ID and group fields. The ID and group must have string values.

Configuration Example

The following is an example of the domain-based user authorization in the tacacs-security mode:

```
config
   tacacs-security domain-based-services nso-service-account id nsid
      tacacs-security domain-based-services nso-service-account group nso-group
   tacacs-security domain-based-services no-domain group read-operational
   tacacs-security domain-based-services domain-delimiter @
   tacacs-security domain-based-services domain-service etcd
      group etcd
exit
tacacs-security domain-based-services domain-service sgw
   group sgw_1
exit
tacacs-security domain-based-services domain-service smf
   group smf
exit
```

Configuration Verification

To verify the configuration, use the following show command:

show running-config tacacs-security

The output of this show command displays all the configurations of the domain-based services within the TACACS security.

```
[smf] smf# show running-config tacacs-security
tacacs-security service smf
tacacs-security server 1
address 209.165.200.234
key $8$+twbdL2ZCgmjVswgp7kFJp8+SMXDjQRTZgoPVa3oEwY=
exit
tacacs-security domain-based-services nso-service-account id nsid
tacacs-security domain-based-services nso-service-account group nso-group
tacacs-security domain-based-services no-domain group read-operational
tacacs-security domain-based-services domain-delimiter @
tacacs-security domain-based-services domain-service etcd
group etcd
exit
tacacs-security domain-based-services domain-service sgw
group sgw_1
exit
tacacs-security domain-based-services domain-service smf
group smf
exit
```

Edge Echo Implementation

Feature Description

In a nonmerged mode, the udp-proxy pod acts as an endpoint, and the gtpc-ep responds to the Echo Requests from the peer node.

The gtpc-ep experiences traffic when the system receives a high number of inputs CEPS leading to a discrepancy between the rate at which gtpc-ep picks up the messages from udp-proxy and the rate at which udp-proxy gets the messages.

If the gtpc-ep is loaded, the queue between the udp-proxy and gtpc-ep gets full, and some of the messages at udp-proxy might get dropped. The peer detects path failure if these are Echo Request messages because an Echo Response is not received. Further, the peer clears all the sessions sent to the sgw-service.

How it Works

This section describes how this feature works.

Nodemgr processes the Echo Request in the following steps:

- The nodemgr preserves a self-restart counter cache for each GR instance ID and the GTPC peer.
- When the udp-proxy pod receives an Echo Request from a peer and the self-restart counter value is not available in the self-restart counter cache, the udp-proxy pod forwards the Echo Request to gtpc-ep.
- The gtpc-ep sends the self-restart counter as part of the UDP proxy message metadata in the Echo Response. The udp-proxy stores the self-restart counter in the self-restart counter cache. When the udp-proxy receives an Echo Request from a peer, and a self-restart counter value is available in the self-restart counter cache, the udp-proxy sends an Echo Response with the restart counter.
- The udp-proxy forwards the Echo Request message to the gtpc-ep. The gtpc-ep processes the Echo Request and forwards it to nodemgr, if necessary.
- If the peer restart counter value is modified, the nodemgr detects a path failure.
- In the Echo Response, the gtpc-ep sends the self-restart counter in the UDP Proxy Message metadata to the udp-proxy. If the self-restart counter differs from the counter that is stored in the self-restart counter cache, the udp-proxy updates the self-restart counter in the cache and drops the Echo Response received from the gtpc-ep.

Note The Edge Echo feature is not supported when the gtpc-ep is started in the merged mode.

Heartbeat

To handle the Echo Request and Echo Response messages for the GTPV2 interface, a heartbeat queue is implemented between the gtpc-ep and the udp-proxy pod. The heartbeat queue is responsible for handling the HeartBeat Request and HeartBeat Response Messages between the protocol and udp-proxy pod for the PFCP interface.

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the Edge Echo Implementation feature:

- Heartbeat queue status:

```
sum(irate(ipc_response_total{rpc_name~=".ipc_stream_hb."}[10s])) by
(service_name,
instance_id, status, status_code, rpc_name, dest_host)
```

- Check the EdgeEcho messages:

```
sum(irate(udp_proxy_msg_total{ message_name ="edge_echo"}[30s])) by
(message_name,
message_direction, status)
```

To enable the Heartbeat queue and EdgeEcho messages statistics, configure the trace-level statistics for udp_proxy_msg_total using the following:

```
infra metrics verbose application
   metrics udp_proxy_msg_total level trace
   exit
```

Note Enabling the heartbeat and EdgeEcho messages statistics may lead to a performance degradation on the udp-proxy pod.

ETCD Peer Optimization Support

Feature Description

When large numbers of GTPC peers are connected with SMF or cnSGW-C, the performance of ETCD is impacted. Each peer is a considered as a record in the ETCD, and the timestamp is updated every 30 seconds for each peer. This causes continuous updates on ETCD and generates huge traffic that impacts the overall system performance.

The ETCD Peer Optimization feature facilitates optimization in peer management and enables reduced performance impact on ETCD.

How it Works

This section describes how this feature works.

Instead of considering each peer as an ETCD record entry, several peers are grouped as a peer group based on the hash value of the IP address of each peer. This reduces the number of entries in ETCD. By default, a maximum of 200 peer groups can be created. For any changes related to a peer in a peer group:

- For a new peer, the peer group is persisted immediately in ETCD.
- For the change in timestamp for existing peers, the peer group is updated once every 3 seconds. This update:
  - Results in a cumulative group update for many peers that have undergone timestamp change within each peer group.
  - Reduces frequent updates to ETCD.

Optimized GTPv2 Encoder and Decoder

Feature Description

cnSGW-C provides an optimized GTPv2 encoder and decoder for:

- Modify Bearer Request and Response messages when the subscriber is moving to ACTIVE state after receiving the Modify Bearer Request message.
- Release Access Bearer Request and Response messages when the subscriber is moving to IDLE state on receiving the Release Access Bearer Request message.

The optimized GTPv2 encoder and decoder is provided for the following messages:

- Bearer Resource Command
- Change Notification Request and Response
- Create Bearer Request and Response
- Create IDFT Request and Response
- Create Session Request and Response
- Delete Bearer Command
- Delete Bearer Failure Indication
- Delete Bearer Request and Response
- Delete IDFT Request and Response
- Delete Session Request and Response
- Downlink Data Notification Acknowledgment
- Downlink Data Notification Failure Indication
- Download Datalink Notification Request
- Echo Request and Response
- Modify Bearer Request and Response
- Modify Bearer Command
- Modify Bearer Failure Indication
- Update Bearer Request and Response

Feature Configuration

To configure this feature on S11, S5, and S5e interfaces, use the following configuration:

```
config
  instance instance-id instance_id
```

```
endpoint gtp
   replicas replica_count
   vip-ip ipv4_address  vip-port ipv4_port_number
   vip-ipv6 ipv6_address  vip-ipv6-port ipv6_port_number
   dual-stack-transport { true | false }
   enable-go-encdec { true | false }
   interface interface_name
      enable-go-encdec { true | false }
      end
```

NOTES:

- **enable-go-encdec { true | false }**—Enable the Go language-based GTPv2 encoder and decoder for the interface.
- **dual-stack-transport { true | false }**—Enable the dual stack feature that allows you to specify IPv6 or IPv4 address. Specify true to enable this feature.

Configuration Example

The following is an example configuration.

```
config
    instance instance-id 1
       endpoint gtp
          replica 2
          vip-ip 209.165.200.224 vip-port 2022
          vip-ipv6 ipv6_address 2001:db8:1::2 22
          dual-stack-transport true
          enable-go-encdec false
          exit
       interface s5e
          replica 3
          vip-ip 209.165.200.225 vip-port 2022
          vip-ipv6 ipv6_address 2001:db8:1::3 22
          dual-stack-transport true
          enable-go-encdec true
          exit
       interface s11
          replica 3
          vip-ip 209.165.200.226 vip-port 2022
          vip-ipv6 ipv6_address 2001:db8:1::4 22
          dual-stack-transport true
          enable-go-encdec true
          exit
       interface s5
          replica 3
          vip-ip 209.165.200.227 vip-port 2022
          vip-ipv6 ipv6_address 2001:db8:1::5 22
          dual-stack-transport true
          enable-go-encdec true
          end
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the Optimized GTPv2 Encoder and Decoder feature:

Grafana Query for Go based encoder decoder:

Query:

```
sum(irate(gtpc_golang_enc_dec_stats{namespace="$namespace"}[60s]))
by (gtpc_msg_type, gtpc_msg_operation, gtpc_msg_status)
```

Legend:

```
{{gtpc_msg_type}}-{{gtpc_msg_operation}}-{{gtpc_msg_status}}
```

GTPC Endpoint with GR Split

Feature Description

The GR Split feature enables handling the scaled GTP traffic and facilitates the optimal use of the CPU. The GR Split feature starts multiple active instances of GTPC-EP and performs traffic split which is based on GR instances. This helps in aiding the UDP proxy bypass feature.

How it Works

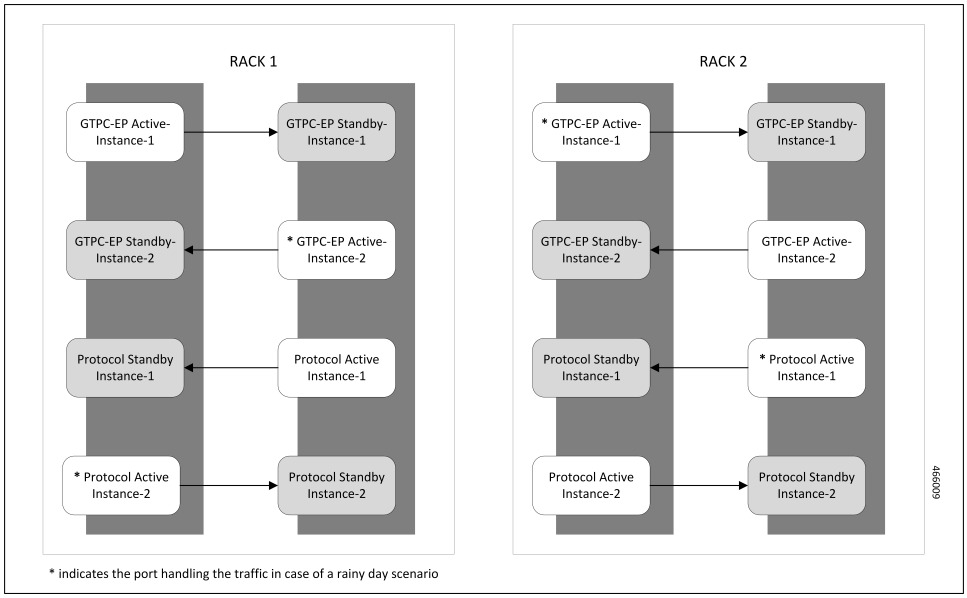
This section describes how this feature works.

The UDP Proxy merge mode is a prerequisite for this feature.

- For a sunny-day scenario, RACK1 (GTPC-EP Active Instance-1) and RACK2 (GTPC-EP Active Instance-2) handle the traffic from GR-Instance-1 and GR Instance-2, respectively.
- For a rainy-day scenario, the GR Instance-1 (S11, S5E, S5) and GR Instance-2 (S11, S5E, S5) traffic splits between two GTPC-EP instances.

  In the rainy-day scenario, assuming RACK2 is down, RACK1 handles all the traffic for GR Instance-1 and GR Instance-2. With the GR Split implementation, the GTPC-EP Active Instance-1 handles GTP traffic for all the interfaces for GR Instance-1, and GTPC-EP Active Instance-2 handles all the GTP traffic for GR Instance-2.

Figure 3: GTPC-EP with Merged Mode and GR Split



GTPC Endpoint Interface Split with S11 and S5

Feature Description

The GTPC Interface Split feature enables splitting the GTPC endpoint based on the interface. cnSGW-C splits the GTPC pod into two interfaces, S11 and S5 which enables handling all GTPC incoming and outgoing traffic for S11 (SGW-Ingress), S5E (SGW-Egress), and S5 (SMF-Ingress).

How it Works

This section describes how this feature works.

The GR Instance Split or UDP Proxy merge mode is a prerequisite for this feature.

This feature is disabled by default. While configuring this feature, provide separate internal and external VIP addresses for S11 and S5 GTPC endpoints, and deploy two GTPC-EP pods.

On configuring the feature, cnSGW-C sends the IPC call from SMF-service, SGW-service, and the Node Manager to the GTPC pod based on the GTPC interface and the GR instance ID.

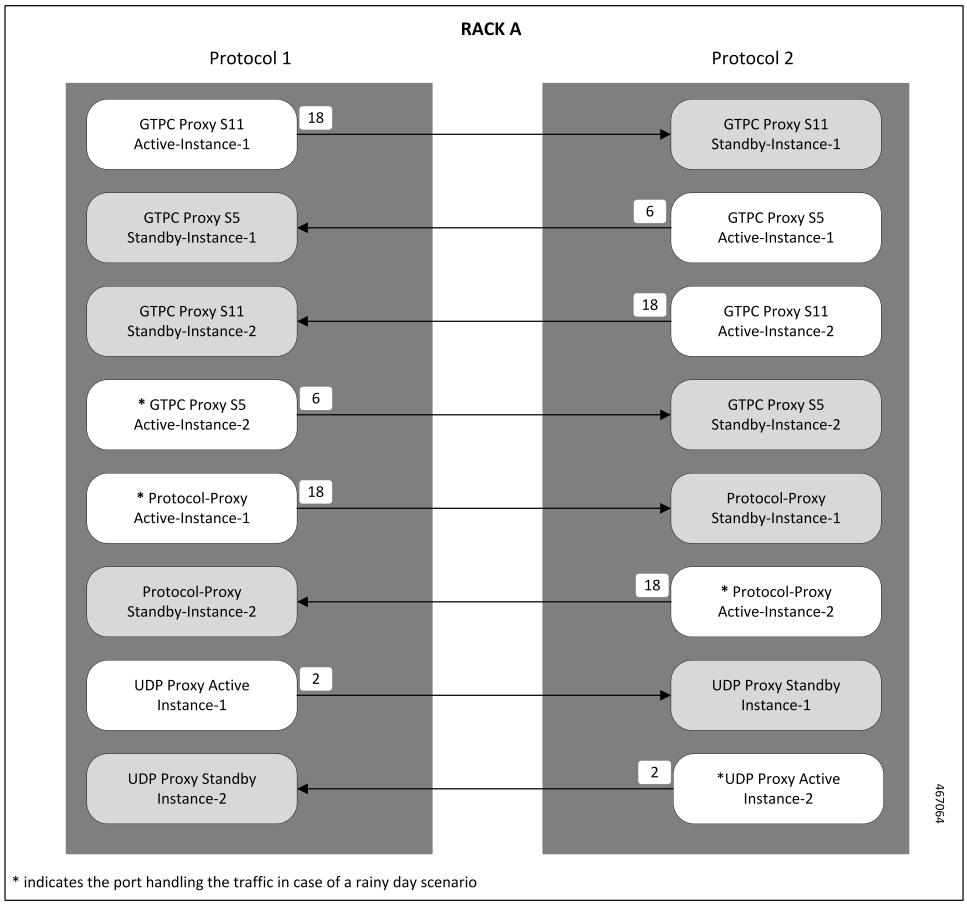
The following are the recommendations for the GTPC interface split feature:

- CPU Core:
  - S5 Interface: 6
  - S11 Interface: 18
  - UDP Proxy: 2

- VIP Configuration: Separate internal and external VIP for S5 and S11 interfaces
- The Dispatcher configuration must be the same configuration as existing before upgrade.

The following diagram indicates the GTPC Interface Split pod layout.

Figure 4: GTPC Interface Split pod layout



Feature Configuration

To configure this feature, use the following configuration:

```
config
  instance instance-id instance_id
    endpoint gtp
    interface s11
      standalone true
      cpu max-process cpu_core_value
      internal-vip internal_vip_address
      vip-ip ipv4_address vip-port ipv4_port_number
      vip-ipv6 ipv6_address vip-ipv6-port ipv6_port_number
      dual-stack-transport { true | false }
      vip-interface vip_interface_value
```

```
      exit
   exit
   interface s5
      vip-ip ipv4_address vip-port ipv4_port_number
      vip-ipv6 ipv6_address vip-ipv6-port ipv6_port_number
      dual-stack-transport { true | false }
      vip-interface vip_interface_value
      end
```

NOTES:

- **standalone true**: Configures the interface to run in standalone mode with a separate pod for the interface.
- **cpu max-process** *cpu_core_value*: Specify the CPU core value for the CPU for the interface. This sets the GO_MAX_PROCS parameter value for the pod.
- **dual-stack-transport { true | false }**—Enable the dual stack feature that allows you to specify IPv6 or IPv4 address. Specify true to enable this feature.

Configuration Example

The following is an example configuration.

```
config
   instance instance-id 1
      endpoint gtp
      interface s11
         standalone true
         cpu max-process 18
         internal-vip 209.165.200.225
         vip-ip 209.165.200.226 vip-interface bd1.gtp.2131
         exit
      exit
      interface s5
      vip-ip 209.165.200.228 vip-interface bd1.gtp.2131
      end
```

GTPC IPC Cross-rack Support

Feature Description

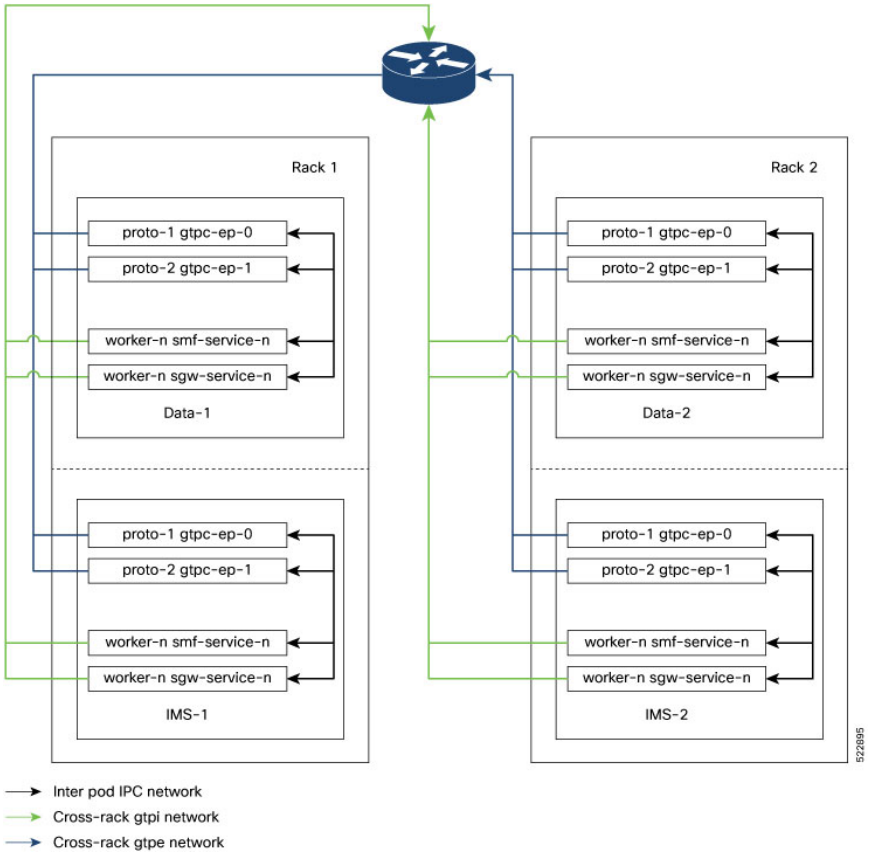
When you perform GR-setup activities with cnSGW-C and SMF, the GTPC message handling can be optimized between these two racks, as in the following scenarios:

- The set of IPC messages from cnSGW-C to SMF service pods flow over `gtpc-ep` pods twice leading to message encoding and decoding overheads.
- Within a GR pair, these IPC messages can avoid one more processing step, if service pods such as cnSGW-C and SMF can route messages to the corresponding peer GTPC nodes directly.

Note Before applying the configuration for enabling GTPC IPC on cnSGW or SMF interfaces, it is required to apply inter-rack routing networks using cluster sync. More configuration is required to add BGP routes for supporting new routable networks across rack servers.

The following figure represents a design of the new network layout that is required for supporting the feature, the core setup activities, and their interconnections.

Figure 5: SGW-GTPC Inter-rack IPC



The following steps are performed for the GTPC message handling optimization between two racks and deploying the cross-rack endpoints:

- The cnSGW-C in IMS-1 Rack-1 routes the IPC request internally to PGW GTPC-EP in DATA-2 Rack-2 passing through the cross-rack GTPI network to the router.
- The router will then use the GTPE network as the next-hop for forwarding requests to the `gtpc-ep` pod.
- The GTPI and GTPE network are new networks added to the Racks during the process of deployment.
- Also, the feature requires internal GTPC IPC messages, which are received on the active `gtpc-ep` pod.
- In this process, the IPC messages from cnSGW-C to SMF service pods flow over the GTPC-EP pods, leading twice to message encoding and decoding outlays.

- Within a GR pair, such IPC messages can avoid one extra hop of processing, if these service pods (cnSGW-C and SMF) can route messages to the corresponding peer GTPC nodes directly.

Note The configured protocol nodes must be in the same VIP group as S5 and S5e VIP groups are deployed.

How it Works

This section describes how this feature works:

- In `SMF-Ops-Center`, you can configure `GTPC-EP Geo` endpoints for each rack in IMS and data racks.
- In `SMF-Ops-Center` CLI, you can configure `GTPC-EP Geo` endpoints for each rack in IMS and DATA racks.

Note The optimization of GTPC messages can be applied to all four instances or a subset of instances of GTPC endpoints within these racks. A new element is added under the GTPC endpoints to configure a list of IP addresses where SMF and cnSGW-C service pods can route the GTPv2 messages over the IPC interface.

Upgrading and Enabling Inter-rack GTPC IPC

This section describes how to upgrade and enable the inter-rack GTPC IPC.

Before you configure, upgrade, and enable the GTPC IPC optimization on cnSGW-C and SMF interfaces, you must perform the following:

- The inter-rack routing must be applied to the branched core network.
- More configuration parameters such as GTPC, cnSGW-C, and SMF are enabled, when extra routes are added for supporting a new routable network across rack servers.

Note The following are configuration-related points:

- The SGW service pods for the S5 egress MBR request are used for IPC messages toward PGWc and SMF IP in the list.
- The SMF service pods for S5 CBR, UBR, and DBR requests are used for IPC messages toward the cnSGW-C in the list.
- IPC messages reuse the N3 or the T3 configuration for the respective interfaces to retry messages, whenever the timeout in the peer node occurs.

Call Flows

This section describes the key call flows for this feature.

cnSGW-C GTPC Optimization with Inter-rack IPC Call Flow

This section describes the cnSGW-C GTPC Optimization with Inter-rack IPC call flow.

Figure 6: cnSGW-C GTPC Optimization with Inter-rack IPC Call Flow

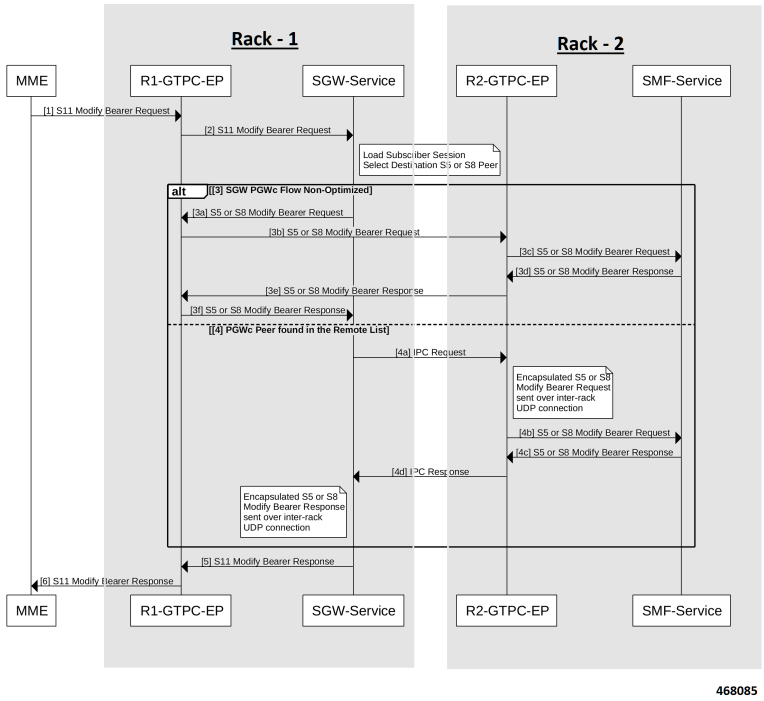


Table 5: cnSGW-C GTPC Optimization with Inter-rack IPC Call Flow Description

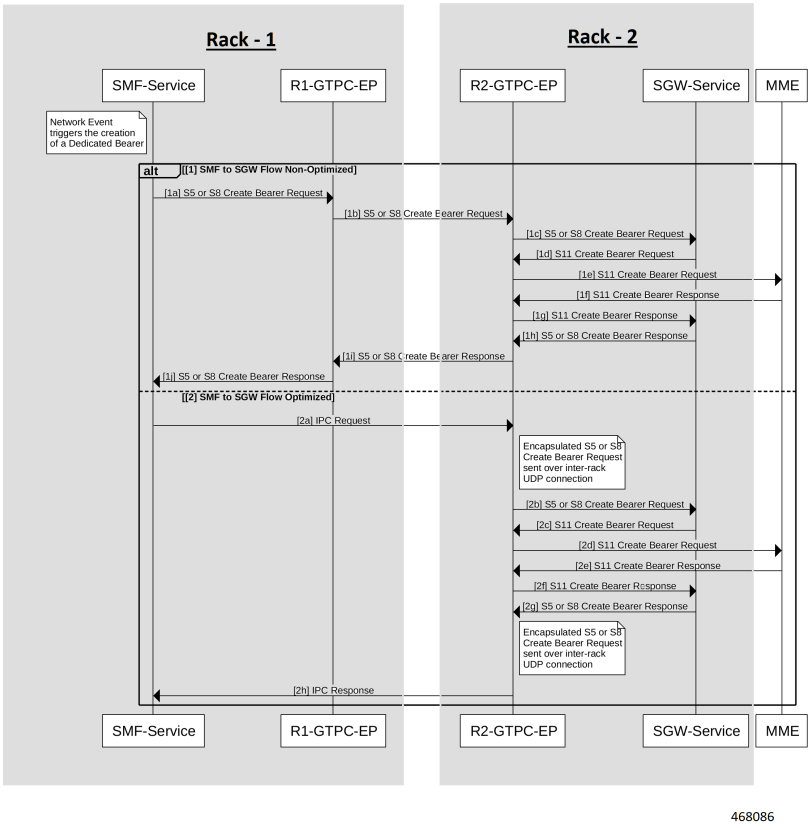
Step	Description
1	MME sends the S11 Modify Bearer Request to R1-GTPC-EP.
2	R1-GTPC-EP sends the S11 Modify Bearer Request to SGW-service. **Note** The SGW-service section performs the following: • Loads the subscriber session. • Selects the destination as the S5 or the S8 peer.
3	The following are sub-steps in the Alt SGW PGWc flow non-optimized scenario.
3a	SGW-service sends the S5 or the S8 Modify Bearer Request to R1-GTPC-EP.
3b	R1-GTPC-EP sends the S5 or the S8 Modify Bearer Request to R2-GTPC-EP.
3c	R2-GTPC-EP sends the S5 or the S8 Modify Bearer Request to SMF-service.
3d	SMF-service processes and sends the S5 or the S8 Modify Bearer Response to R2-GTPC-EP.

Step	Description
3e	R2-GTPC-EP sends the Modify Bearer Response to R1-GTPC-EP.
3f	R1-GTPC-EP sends the Modify Bearer Response to SGW-service.
4	Alternatively, the PGWc peer scenario is available in the remote list. The following are sub-steps in this scenario.
4a	SGW-service sends the IPC Request to R2-GTPC-EP. **Note** The R2-GTPC-EP section performs the following: • Encapsulates the S5 or the S8 Modify Bearer Request. • Sends it over the inter-rack UDP connection.
4b	R2-GTPC-EP sends the S5 or the S8 Modify Bearer Request to SMF-service.
4c	SMF-service processes and sends the S5 or the S8 Modify Bearer Response to R2-GTPC-EP.
4d	R2-GTPC-EP sends the IPC Response to SGW-service. **Note** The SGW-service section performs the following: • Encapsulates the S5 or the S8 Modify Bearer Response. • Sends it over the inter-rack UDP connection.
5	SGW-service sends the Modify Bearer Response to R1-GTPC-EP.
6	R1-GTPC-EP sends the Modify Bearer Response to MME.

SMF and PGWc GTPC Optimization with Inter-rack IPC Call Flow

This section describes the SMF and PGWc GTPC Optimization with Inter-rack IPC call flow.

Figure 7: SMF and PGWc GTPC Optimization with Inter-rack IPC Call Flow



468086

Table 6: SMF and PGWc GTPC Optimization with Inter-rack IPC Call Flow Description

Step	Description
1	The following are sub-steps in the Alt SMF to SGW flow non-optimized scenario. **Note** The SMF-service section performs the following: • Triggers the networking event. • Creates the resolute bearer.
1a	SMF-service sends the S5 or the S8 Create Bearer Request to R1-GTPC-EP.
1b	R1-GTPC-EP sends the S5 or the S8 Create Bearer Request to R2-GTPC-EP.
1c	R2-GTPC-EP sends the S5 or the S8 Create Bearer Request to SGW-service.
1d	SGW-service processes and sends the S11 Create Bearer Request to R2-GTPC-EP.
1e	R2-GTPC-EP sends the S11 Create Bearer Request to MME.
1f	MME processes and sends the S11 Create Bearer Response to R2-GTPC-EP.

Step	Description
1g	R2-GTPC-EP sends the S11 Create Bearer Response to SGW-service.
1h	SGW-service processes and sends the S5 or the S8 Create Bearer Response to R2-GTPC-EP.
1i	R2-GTPC-EP sends the S5 or the S8 Create Bearer Response to R1-GTPC-EP.
1j	R1-GTPC-EP sends the S5 or the S8 Create Bearer Response to SMF-service.
2	The following are sub-steps in the SMF to SGW flow-optimized scenario.
2a	SMF-service sends the IPC request to R2-GTPC-EP. **Note** The R2-GTPC-EP section performs the following: • Encapsulates the S5 or the S8 Create Bearer Request. • Sends it over the inter-rack UDP connection.
2b	R2-GTPC-EP sends the S5 or the S8 Create Bearer Request to SGW-service.
2c	SGW-service processes and sends the S11 Create Bearer Request to R2-GTPC-EP.
2d	R2-GTPC-EP sends the S11 Create Bearer Request to MME.
2e	MME processes and sends the S11 Create Bearer Response to R2-GTPC-EP.
2f	R2-GTPC-EP sends the S11 Create Bearer Response to SGW-service.
2g	SGW-service processes and sends the S5 or the S8 Create Bearer Response to R2-GTPC-EP. **Note** The R2-GTPC-EP section performs the following: • Encapsulates the S5 or the S8 Create Bearer Request. • Sends it over the inter-rack UDP connection.
2h	R2-GTPC-EP processes and sends the IPC Response to SMF-service.

Feature Configuration

To configure this feature, use the following configuration:

```
config
instance instance_id 1
   endpoint endpoint_name
      interface gtp-inter-rack gtp_inter_rack_name
         vip-ip vip_ip_address vip-port vip_port_address vip-interface
vip_interface_address
         gtpc-ipc gtpc_ipc_name
         gtp-peer-entry gtp_peer_entry_address port port_address
remote-gtp-peer-list remote_gtp_peer_list_addresses
         end
```

Note In the preceding new configuration example, the following are the enhanced scenarios:

- The cnSGW-C service pod for the S5 egress MBR request will be using IPC messages toward the PGWc or the SMF IP in the preceding list.
- Similarly, the SMF service pod for S5 CBR, UBR, and DBR requests are using IPC messages toward cnSGW in the preceding list.
- IPC messages reuse the N3 or T3 configuration for the respective interface to retry messages, when there is a timeout in the peer node.

NOTES:

- **instance** *instance_id 1*—Specify the instance ID.
- **endpoint** *endpoint_name*—Specify the endpoint name.
- **gtp-inter-rack** *gtp_inter_rack_name*—Specify the interface name. Specify the **gtp-inter-rack** name, you want to select. It is a new interface, added for cross-rack routing.
- **vip-ip** *vip_ip_address* **vip-port** *vip_port_address* **vip-interface** *vip_interface_address*—Specify the addresses for **vip-ip**, which is a GTP IPC endpoint server IP, **vip-port**, which is a GTP IPC endpoint server listening port, and **vip-interface**, which is a GTP IPC endpoint server interface VLAN.
- **gtpc-ipc** *gtpc_ipc_name*—Specify the interface name. Specify the **gtp-ipc** name, you want to select.
- **gtp-peer-entry** *gtp_peer_entry_address* **port** *port_address* **remote-gtp-peer-list** *remote_gtp_peer_list_addresses*—Specify the addresses for the list of **gtp-peer-entry**, which is a remote GTP IPC peer IP configured on other racks or instances (multiple rows), **port**, which is a remote GTP IPC peer port, and **remote-gtp-peer-list**, which is a list of S5 and S5e remote GTP peers endpoints on a rack or instances corresponding to the **gtp-peer-entry**.

Configuration Example

The following is an example configuration.

```
config
instance instance-id 1
 endpoint GTP
   interface gtp-inter-rack
     vip-ip 209.165.202.130 vip-port 9084 vip-interface bd2.gtpe.2101
     gtpc-ipc
     gtp-peer-entry 209.165.202.131 port 9084 remote-gtp-peer-list [ 209.165.202.140
209.165.202.141 ]
     end
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

KPI Support

The following statistics are supported for the GTPC IPC Cross-rack Support feature.

1. KPI Name: **udp_rpc_request_total**

   The following table lists **udp_rpc_request_total** KPI details.

Description	Functionality	Label Names	Possible Values
This KPI displays the total number of UDP client endpoint request messages.	The functionality output is inter-rack RPC requests total.	MessageName	gptv2 message types like CBR, UBR, MBR
		retry	Is attempt a retry
		status	Status of the request message, which is a success or a failure
		status_code	0/1/2

2. KPI Name: **udp_rpc_response_total**

   The following table lists **udp_rpc_response_total** KPI details.

Description	Functionality	Label Names	Possible Values
This KPI displays the total number of UDP client endpoint response messages.	The functionality output is inter-rack RPC response total.	MessageName	gptv2 message types like CBR, UBR, MBR
		status	Status of the response message, which is a success or a failure
		status_code	0/1/2

Note The current implementation supports KPIs only on the client-side, as they reside on service pods, where KPIs can be enabled without impacting performance.

Interservice Pod Communication

Feature Description

When the IMS PDN sgw-service and smf-service selected for a subscriber are on the same cluster and same RACK, the following message flow occurs when sgw-service sends a message to smf-service:

- The message is sent from S5e gtpc-ep interface to network interface.
- The message returns to the S5 interface from gtpc-ep to smf-service.

For the subscribers that are collocated, the communication happens between the sgw-service and the smf-service. This approach reduces the processing load on the gtpc-ep.

Note A direct communication between sgw-service and smf-service is not supported to transfer messages on monitor protocol and monitor subscriber.

How it Works

This section describes how this feature works.

The sgw-service communicates with smf-service for processing the following requests:

- Create Session Request
- Modify Bearer Request
- Delete Session Request

The smf-service communicates with sgw-service for processing the following requests:

- Create Bearer Request
- Update Bearer Request
- Delete Bearer Request

The sgw-service sends the Modify Bearer Command and Delete Bearer Command messages to SMF through gtpc-ep. If the Update Bearer Request and Delete Bearer Request is triggered, the command messages are sent to sgw-service through gtpc-ep.

Call Flows

This section describes the key call flows for this feature.

Collapsed Call Attach with SGW-Service to SMF-Service Configuration Call Flow

This section describes the Collapsed Call Attach with SGW-Service to SMF-Service Configuration call flow.

Figure 8: Collapsed Call Attach with SGW-Service to SMF-Service Configuration Call Flow

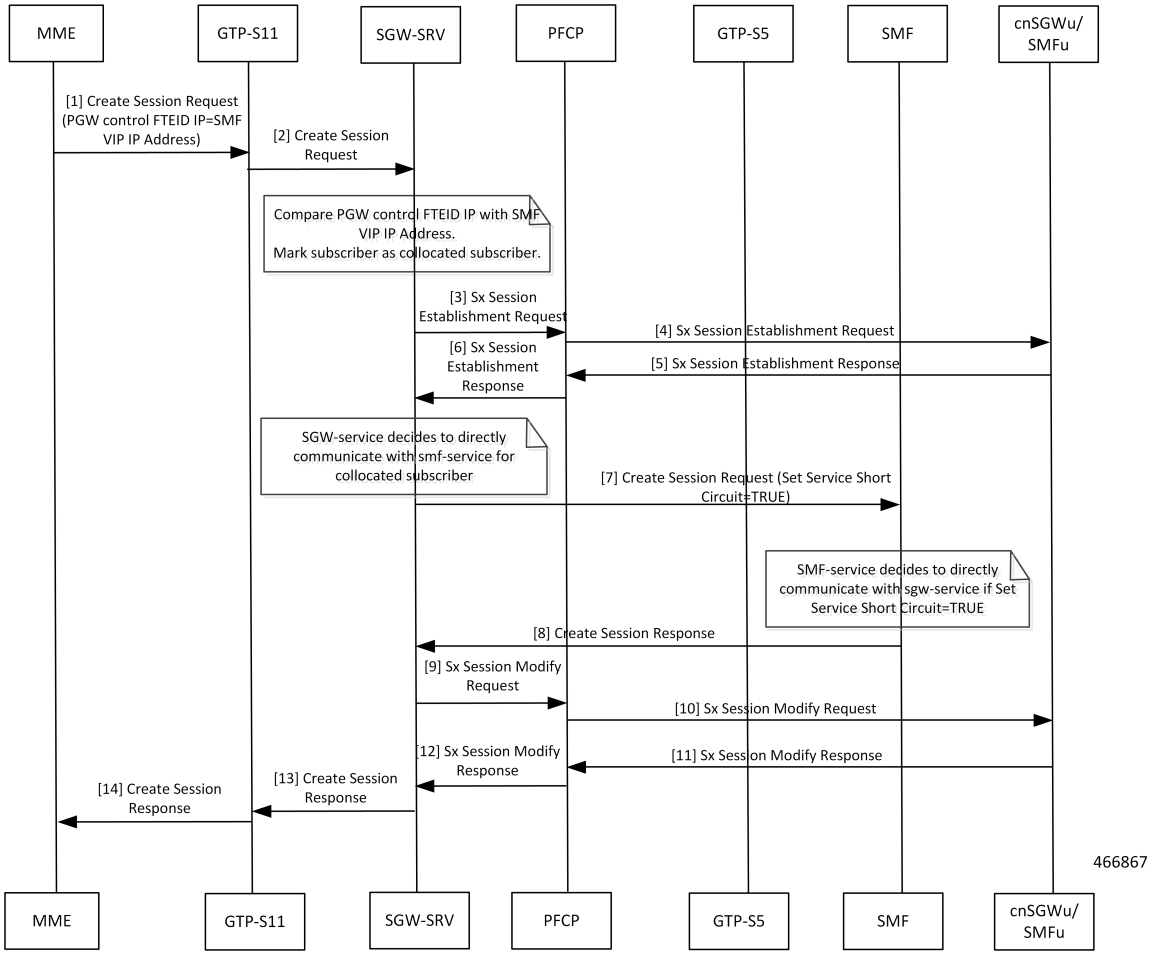


Table 7: Collapsed Call Attach with SGW-Service to SMF-Service Configuration Call Flow Description

Step	Description
1	MME sends the Create Session Request to GTP-S11.
2	GTP-S11 sends the Create Session Request to SGW-SRV.
3	SGW-SRV sends the Sx Session Establishment Request to PFCP.
4	PFCP sends the Sx Session Establishment Request to cnSGWu/SMFu.
5	cnSGWu/SMFu sends the Sx Session Establishment Response to PFCP.
6	PFCP sends the Sx Session Establishment Response to SGW-SRV.
7	SGW-SRV sends the Create Session Request to SMF. The Service Short Circuit is set to TRUE.
8	SMF sends the Create Session Response to SGW-SRV.
9	SGW-SRV sends the Sx Session Modify Request to PFCP.

Step	Description
10	PFCP sends the Sx Session Modify Request to cnSGWu/SMFu.
11	cnSGWu/SMFu sends the Sx Session Modify Response to PFCP.
12	PFCP sends the Sx Session Modify Response to SGW-SRV.
13	SGW-SRV sends the Create Session Response to GTP-S11.
14	GTP-S11 sends the Create Session Response to MME. For a collocated subscriber, when SGW-SRV receives: • Modify Bearer Request and Delete Session Request from the MME, the SGW-SRV communicates these messages to the SMF service. • Create Bearer Request, Update Bearer Request, and Delete Bearer Request from the SMF, the SGW-SRV communicates the response messages for these request messages to the SMF service.

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Statistics Support

To check for messages that are directly communicated to SMF service, add svc_to_svc field in sgw_service_stats query as shown below:

Query: sum(irate(sgw_service_stats{status=~"attempted"}[30s])) by (sgw_procedure_type,status, interface, svc_to_svc)

Legend: {{interface}} -> {{sgw_procedure_type}} | {{svc_to_svc}} | {{status}}

MBR Call Flow Optimization

Feature Description

cnSGW-C supports optimization of Modify Bearer Request and Modify Bearer Response (MBR) call flows to reduce I/O operation, reduce transaction wait time, and improve performance in multi-PDN scenarios.

How it Works

This section describes how this feature works.

The following functions explain the optimization of MBR call flows:

- To reduce I/O operations, cnSGW-C combines all Modify Bearer Requests towards SGW-service into a single GRPC call, and Sx Modify Requests from SGW-service pod to protocol pod in a single GRPC call.

- To reduce transaction wait time in GTPC-EP, cnSGW-C sends Modify Bearer Response immediately from GTPC-EP (except last MBR) after receiving Modify Bearer Request.

  GTPC-EP combines all Modify Bearer Requests in a single Modify Bearer Request List message and sends to SGW-service.

- SGW-service combines all Modify Bearer Responses into a single Modify Bearer Response List message and sends to GTPC-EP.

  SGW-service combines all Sx Modify Requests towards UPF into a single Sx Modify Request List message and sends to protocol pod. The protocol pod sends individual Sx Modify Requests to UPF.

- The protocol pod waits for all Sx Modify Responses from UPF and combines them into a single Sx Modify Response List and sends it to SGW-service.

- In non-merged mode, UDP proxy maintains the local TEID and remote TEID cache information. In merged mode, GTPC-EP maintains the local TEID and remote TEID cache information.

- If GTPC-EP does not find the TEID cache entry for the received Modify Bearer Request, the Modify Bearer Request will be forwarded to the SGW-service immediately.

  If all expected Modify Bearer Requests are not received within the MBR cache expiry, only the Modify Bearer Requests that are received will be sent to the SGW-service.

Call Flows

This section describes the key call flows for this feature.

Idle-Active Transition with Intra MME HO Call Flow

This section describes the current call flow with intra MME handover (NSA MBR) for moving from eNodeB to gNodeB.

Figure 9: Idle-Active Transition with Intra MME HO Call Flow

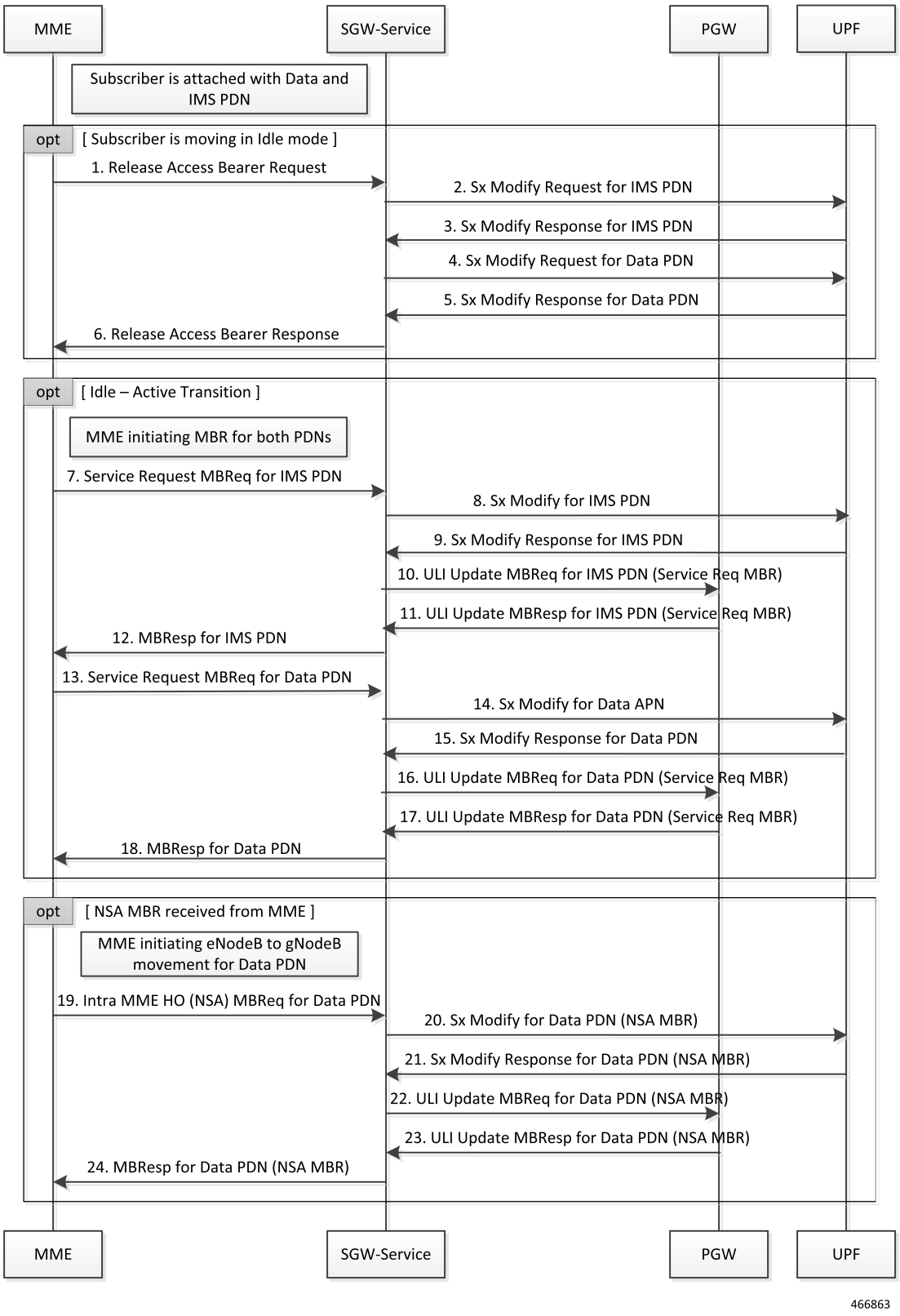

MME
SGW-Service
PGW
UPF
Subscriber is attached with Data and IMS PDN
opt
[Subscriber is moving in Idle mode]
1. Release Access Bearer Request
2. Sx Modify Request for IMS PDN
3. Sx Modify Response for IMS PDN
4. Sx Modify Request for Data PDN
5. Sx Modify Response for Data PDN
6. Release Access Bearer Response
opt
[Idle – Active Transition]
MME initiating MBR for both PDNs
7. Service Request MBReq for IMS PDN
8. Sx Modify for IMS PDN
9. Sx Modify Response for IMS PDN
10. ULI Update MBReq for IMS PDN (Service Req MBR)
11. ULI Update MBResp for IMS PDN (Service Req MBR)
12. MBResp for IMS PDN
13. Service Request MBReq for Data PDN
14. Sx Modify for Data APN
15. Sx Modify Response for Data PDN
16. ULI Update MBReq for Data PDN (Service Req MBR)
17. ULI Update MBResp for Data PDN (Service Req MBR)
18. MBResp for Data PDN
opt
[NSA MBR received from MME]
MME initiating eNodeB to gNodeB movement for Data PDN
19. Intra MME HO (NSA) MBReq for Data PDN
20. Sx Modify for Data PDN (NSA MBR)
21. Sx Modify Response for Data PDN (NSA MBR)
22. ULI Update MBReq for Data PDN (NSA MBR)
23. ULI Update MBResp for Data PDN (NSA MBR)
24. MBResp for Data PDN (NSA MBR)
MME
SGW-Service
PGW
UPF
466863

Table 8: Idle-Active Transition with Intra MME HO Call Flow Description

Step	Description
1	MME sends Release Access Bearer (RAB) request to SGW-service.
2	SGW-service sends Sx Modify Request for IMS PDN to UPF.
3	UPF sends Sx Modify Response for IMS PDN to SGW-service.
4	SGW-service sends Sx Modify Request for Data PDN to UPF.
5	UPF sends Sx Modify Response for Data PDN to SGW-service.
6	SGW-service sends the Release Access Bearer Response message to MME.
7	MME sends Service Request MBRequest for IMS PDN to SGW-service.
8	SGW-service sends Sx Modify Request for IMS PDN to UPF.
9	UPF sends Sx Modify Response for IMS PDN to SGW-service.
10	SGW-service sends ULI Update MBRequest for IMS PDN (Service Request MBR) to PGW.
11	PGW sends ULI Update MBResponse for IMS PDN (Service Request MBR) to SGW-service.
12	SGW-service sends MBResponse for IMS PDN to MME.
13	MME sends Service Request MBRequest for Data PDN to SGW-service.
14	SGW-service sends Sx Modify Request for Data PDN to UPF.
15	UPF sends Sx Modify Response for Data PDN to SGW-service.
16	SGW-service sends ULI Update MBRequest for Data PDN (Service Request MBR) to PGW.
17	PGW sends ULI Update MBResponse for Data PDN (Service Request MBR) to SGW-service.
18	SGW-service sends MBResponse for Data PDN to MME.
19	MME sends intra MME HO (NSA) MBRequest for Data PDN to SGW-service.
20	SGW-service sends Sx Modify Request for Data PDN (NSA MBR) to UPF.
21	UPF sends Sx Modify Response for Data PDN (NSA MBR) to SGW-service.
22	SGW-service sends ULI Update MBRequest for Data PDN (NSA MBR) to UPF.
23	UPF sends ULI Update MBResponse for Data PDN to SGW-service.
24	SGW-service sends MBResponse for Data PDN (NSA MBR) to MME.

MBR Optimization Call Flow

This section describes the high-level MBR Optimization call flow for idle↔active transition.

Figure 10: MBR Optimization Call Flow

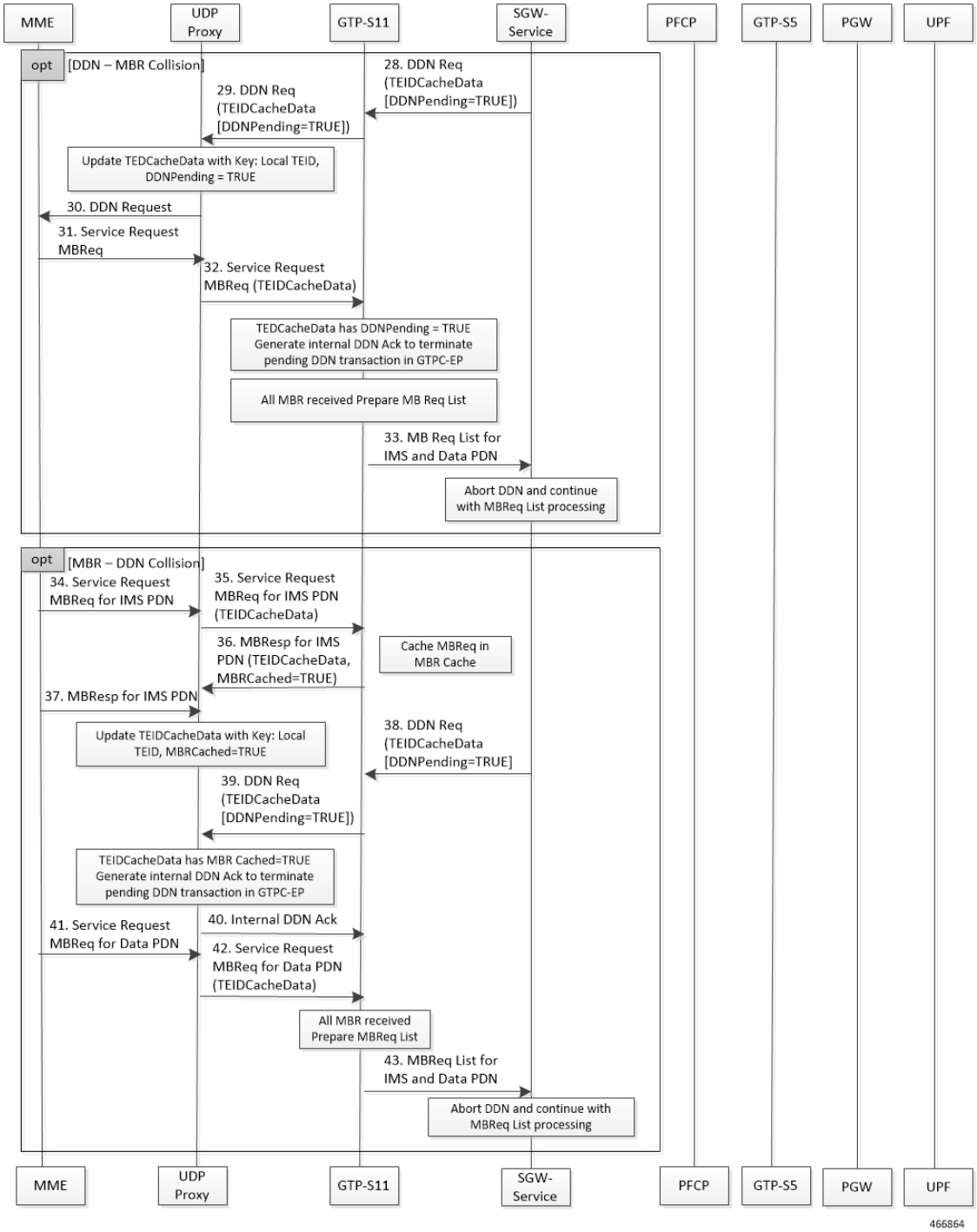


Table 9: MBR Optimization Call Flow Description

Step	Description
1	MME sends Release Access Bearer (RAB) request to UDP proxy.
2	UDP Proxy forwards RAB Request to GTP-S11.
3	GTP-S11 forwards RAB Request to SGW-service.
4	SGW-service sends Sx Modify List for all PDNs to PFCP.
5	SGW-service prepones RAB Response.
6	GTP-S11 sends RAB Response to UDP Proxy.
7	UDP Proxy sends RAB Response to MME.
8	PFCP sends and receives Sx Modify for IMS APN to and from UPF.
9	PFCP sends and receives Sx Modify for Data APN to and from UPF.
10	PFCP sends Sx Modify Response List for all PDNs to SGW-service.
11	MME sends Service Request MBRequest for IMS PDN to UDP Proxy.
12	UDP Proxy forwards Service Request MBRequest for IMS PDN with TEID cache data to GTP-S11.
13	GTP-S11 sends MBResponse for IMS PDN to UDP Proxy.
14	UDP Proxy forwards MBResponse for IMS PDN to MME.
15	MME sends Service Request MBRequest for Data PDN to UDP Proxy.
16	UDP Proxy sends Service Request MBRequest for Data PDN with TEID cache data to GTP-S11.
17	GTP-S11 sends MBRequest List for IMS and Data PDNs to SGW-service.
18	SGW-service sends Sx Modify List for IMS and Data PDN to PFCP.
19	PFCP sends and receives Sx Modify for IMS PDN to and from UPF.
20	PFCP sends and receives Sx Modify for Data PDN to and from UPF.
21	PFCP sends Sx Modify Response List for IMS and Data PDN to SGW-service.
22	SGW-service sends ULI Update MBRequest List for IMS and Data PDN (Async Notification) to GTP-S5.
23	SGW-service sends MBResponse List for Data PDN to GTP-S11.
24	GTP-S11 sends MBResponse for Data PDN to UDP Proxy.
25	UDP Proxy forwards MBResponse for Data PDN to MME.
26	GTP-S5 sends and receives ULI Update MBRequest for IMS PDN to and from PGW.

Step	Description
27	GTP-S5 sends and receives ULI Update MBRequest for Data PDN to and from PGW.
28	SGW-service sends DDN Request to GTP-S11.
29	S-11 forwards DDN Request to UDP Proxy.
30	UDP Proxy forwards DDN Request to MME.
31	MME sends Service Request MBRequest to UDP Proxy.
32	UDP Proxy sends Service Request MBRequest with TEIDCacheData to GTP-S11.
33	GTP-S11 sends MBResponse List for IMS and Data PDNs to SGW-service.
34	MME sends Service Request MBRequest for IMS PDN to UDP Proxy.
35	UDP Proxy Service Request MBRequest for IMS PDN with TEID cache data to GTP-S11.
36	GTP-S11 sends MBResponse for IMS PDN to UDP Proxy.
37	UDP Proxy sends MBResponse for IMS PDN to MME.
38	SGW-service sends DDN Request to GTP-S11.
39	GTP-S11 forwards DDN Request to UDP Proxy.
40	UDP Proxy sends internal DDN acknowledgement to GTP-S11.
41	MME sends Service Request MBRequest for Data PDN to UDP Proxy.
42	MME sends Service Request MBRequest for Data PDN with TEID cache data to GTP-S11.
43	GTP-S11 sends MBRequest List for IMS and Data PDNs to SGW-service.

Feature Configuration

To configure this feature, use the following sample configuration:

```
config
   instance istance-id instance_id
      endpoint endpoint_name
         mbr-optimization [ enable { false | true } | mbr-cache-expiry
mbr_cache | teid-cache-expiry teid_cache ]
         exit
      exit
   exit
```

NOTES:

- **mbr-optimization [enable { false | true } | mbr-cache-expiry** *mbr_cache* **| teid-cache-expiry** *teid_cache* **]**: Specify the MBR optimization configuration.
  - **enable { false | true }**: Enable or disable MBR optimization. Default: Disabled.

- **mbr-cache-expiry** *mbr_cache* : Specify the MBR cache expiry time interval in milliseconds, as an integer from 1 millisecond to 6 seconds. Default: 50 milliseconds.

  Note that the value of **mbr-cache-expiry** can be changed during the runtime.

- **teid-cache-expiry** *teid_cache* : Specify the TEID cache expiry time interval in milliseconds, as an integer from 1000 milliseconds to 1 hour. Default: 120000 milliseconds.

Configuration Example

The following is an example configuration.

```
config
   instance instance-id 1
      endpoint gtp
         mbr-optimization enable true mbr-cache-expiry 60 teid-cache-expiry 180000
         end
```

Configuration Verification

To verify the configuration:

```
show running-config instance instance-id 1 endpoint gtp
instance instance-id 1
endpoint gtp
  replicas                1
  nodes                   1
  mbr-optimization
   enable            true
   teid-cache-expiry 180000
   mbr-cache-expiry  60
  exit
  enable-cpu-optimization true
  ……
```

OAM Support

This section describes operations, administration, and maintenance support for the MBR Optimization feature.

Bulk Statistics Support

The following statistics are supported for the MBR Optimization feature.

The SGW-service supports the *service_request_list* procedure type to handle Modify Bearer Request list from GTPC-EP.

Query: `sum(rate(sgw_service_stats{namespace=~"$namespace", interface="interface_sgw_ingress",sgw_procedure_type="service_request_list", status="attempted"}[1m]))`

Legend: `IDLE -> ACTIVE (List)`

The MBR short circuit statistics are enhanced to capture statistics for Modify Bearer Request list sent to SGW-service.

Query: `sum(irate(gtpc_msg_short_circuit_stats{namespace=~"$namespace"}[60s])) by (gtpc_msg_type, gtpc_short_circuit_category)`

Legend: `{{gtpc_msg_type}}, {{gtpc_short_circuit_category}}`

This feature supports the following statistics:

- RxModifyBearerReq, SendSCMBResp—The number of early MB responses sent from GTPC-EP.
- RxModifyBearerReq, SendMBReqListToSrv—The number of MB request lists sent to SGW-service.
- RxModifyBearerReq, SendMBReqToSrv—The number of MB requests sent to SGW-service.
- RxModifyBearerReq, MBREventExpired—The number of MB requests expired in MBR cache.

Maintenance Mode

Feature Description

The maintenance mode feature allows a cluster in the GR setup to undergo an in-service upgrade (rolling upgrade) without any service disruption. Maintenance mode works with routing the traffic to the mated cluster to perform the responsibility of the source cluster.

How it Works

When the maintenance mode flag is set to true, the cluster role changes and GR is triggered for the rack. The standby cluster takes over the responsibility of the cluster that is in the maintainence mode. During this period, the monitoring threads check the runtime value of the flag and pause the execution when the maintenance mode flag is set to true. By default, for fresh installation, the flag is set to false.

Both, the source and standby clusters (racks) can be under the maintenance mode at the same time. You can enable the maintenance mode for the rack server regardless of its state.

You can push the system to the maintenance mode when the maintenance procedure is in-progress for the mated cluster. Before you start the maintenance activity, set the `geo maintenance mode` flag value to true. When the maintenance is complete, reset the flag to false after confirming the health of the system.

For information on how to configure the maintenance mode flag, see Enabling or Disabling Maintenance Mode, on page 46.

When the maintenance mode is enabled:

- Automated GR-switchover such as pod monitoring, BFD link monitoring from the rack server is not supported.
- Only CLI-based GR-switchover is supported from the rack (with maintenance mode enabled) to the partner rack.
- GR-switchover, including CLI-based, is not supported from the partner rack to the rack where the maintenance mode is enabled.
- If both partner racks are in the maintenance mode, GR-switchover is not supported.
- All the monitoring activities are paused.
- The mated cluster cannot trigger the failover when it detects the local failure.

- Replication activities continue on the cluster.
- Maintenance mode doesn't implicitly change the instance roles of the site. However, role change is possible using the **geo switch-role role** CLI command.

  GR trigger is not supported towards and from the cluster that is under maintenance. Only CLI-based failover is supported from the cluster under the maintenance.

Limitations

This feature has the following limitation in this release:

The maintenance mode feature does not overwrite the multicompute failure switchover case. However, the multicompute failure switchover scenario is supported when the partner rack is also in maintenance mode.

Note The multicompute failure is a switchover case that occurs when multiple servers in a rack fail causing the partner rack to handle the traffic.

Enabling or Disabling Maintenance Mode

To enable or disable this feature, use the following command:

`geo maintenance mode { true | false }`

NOTES:

geo maintenance mode { true | false }—Specify to enable or disable the maintenance mode. To enable the maintenance mode, specify true. If the maintenance mode flag is set to true, the cluster role changes and GR is triggered for the rack.

The value for the maintenance mode is stored in the `.etcd` file.

Enabling or Disabling Maintenance Mode Example

The following is an example of the command:

```
geo maintenance mode true
```

```
geo maintenance mode false
```

Verifying the Maintenance Mode State

To verify the maintenance mode state:

```
show geo maintenance mode
result "geo maintenance mode is disabled"
```

Partial CDL Update for Idle-Active Call Flow

Feature Description

cnSGW-C supports partial CDL update of the subscriber data. Partial CDL update helps in saving the CPU requirement in the CDL pod for processing the subscriber data.

With each transition of the subscriber from Idle-to-Active state and Active-to-Idle state, cnSGW-C updates only the following fields:

- eNodeB FTEID
- Subscriber state
- Bearer state

How it Works

This section describes how this feature works.

cnSGW-C stores the update bearer information in the Flags database in the CDL.

cnSGW-C uses partial CDL update when the subscriber moves from:

- Active state to Idle state on receiving Release Access Bearer Request
- Idle state to Active state on receiving Modify Bearer Request

With partial CDL update, the session-state-flag displays the following value in the **cdl show sessions summary slice-name <n>** CLI output:

- sgw_active: when the session is Active
- sgw_inactive: when the session is Idle

The following is a sample output for an Active session:

```
cdl show sessions summary slice-name 1
message params: {session-summary cli session {0 100  0 [] 0 0 false 4096 [] []} 1}
session {
    primary-key 2#/#imsi-123456789012348
    unique-keys [ "2#/#16777229" ]
    non-unique-keys [ "2#/#id-index:1:0:32768" "2#/#id-value:16777229"
    "2#/#imsi:imsi-123456789012348" "2#/#msisdn:msisdn-223310101010101"
    "2#/#imei:imei-123456786666660" "2#/#upf:209.165.201.1"
    "2#/#upfEpKey:209.165.201.1:209.165.201.30" "2#/#s5s8Ipv4:209.165.202.129"
"2#/#s11Ipv4:209.165.201.1"
    "2#/#namespace:sgw" ]
    flags [ byte-flag1:00:13:03:53:00:00:06:85:0A:01:01:1B session-state-flag:sgw_active ]
    map-id 1
    instance-id 1
    app-instance-id 1
    version 1
    create-time 2022-01-20 11:37:15.181259564 +0000 UTC
    last-updated-time 2022-01-20 11:37:15.703032336 +0000 UTC
    purge-on-eval false
```

```
    next-eval-time 2022-01-27 11:37:26 +0000 UTC
    session-types [ SGW:rat_type:EUTRAN ]
    data-size 925
```

The following is a sample output for an Idle session:

```
cdl show sessions summary slice-name 1
message params: {session-summary cli session {0 100  0 [] 0 0 false 4096 [] []} 1}
session {
    primary-key 2#/#imsi-123456789012348
    unique-keys [ "2#/#16777229" ]
    non-unique-keys [ "2#/#id-index:1:0:32768" "2#/#id-value:16777229"
    "2#/#imsi:imsi-123456789012348" "2#/#msisdn:msisdn-223310101010101"
    "2#/#imei:imei-123456786666660" "2#/#upf:209.165.201.1"
"2#/#upfEpKey:209.165.201.1:209.165.201.30"
    "2#/#s5s8Ipv4:209.165.202.129" "2#/#s11Ipv4:209.165.201.1" "2#/#namespace:sgw" ]
    flags [ byte-flag1:00:25:00:55:00:65 session-state-flag:sgw_inactive ]
    map-id 1
    instance-id 1
    app-instance-id 1
    version 3
    create-time 2022-01-20 11:37:15.181259564 +0000 UTC
    last-updated-time 2022-01-20 11:37:18.102852792 +0000 UTC
    purge-on-eval false
    next-eval-time 2022-01-27 11:37:28 +0000 UTC
    session-types [ SGW:rat_type:EUTRAN ]
    data-size 1644
```

Limitations

cnSGW-C doesn't support partial CDL update for IPv6 TEID.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   cdl
      datastore datastore_session_name
         slot metrics report-idle-session-type { true | false }
         end
```

NOTES:

- **slot metrics report-idle-session-type { true | false }**—Enable or disable Idle or Active session count in CDL db_records_total.

Configuration Example

The following is an example configuration.

```
config
   cdl
      datastore session
         slot metrics report-idle-session-type true
         end
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the Partial CDL Update Idle-Active Call Flow feature.

Grafana Query to find cnSGW-C Idle and Active Session count:

```
SGW ACTIVE COUNT :-
(db_records_total{namespace=~"$namespace",session_type=~"sgw_active"})
SGW_IDLE COUNT :
(db_records_total{namespace=~"$namespace",session_type=~"sgw_inactive"})
```

PFCP Session Report with DLDR Throttling Support

Feature Description

In a live network deployment, due to some external events, all or most of the idle sessions become active at the same time. When idle sessions become active, the UPF sends the session report request with the report type DLDR to the cnSGW-C.

When the cnSGW-C receives the session report with the report type as DLDR, the cnSGW-C sends the DDN message to page UE. To turn the UE active, the MME initiates the paging procedure for the UE. If paging is successful, the MME initiates the service request Modify Bearer Request. On delivering data to the UE, the UE initiates the Release Access Bearer Request and turns idle. This call flow increases an overall load on the system. When the entire call flow occurs for all subscribers in a short time, there's a huge process overhead on the system.

The PFCP Session Report with the DLDR Throttling Support feature enables the cnSGW-C to limit the number of session report requests that enter the system to prevent the process overload on the system.

How it Works

This section describes how this feature works.

The cnSGW-C uses the app-infra feature for SBA overload control to throttle the incoming messages.

The system takes appropriate action for the incoming messages that are based on the interface-level threshold configuration. The incoming messages get added to different queues and they are processed based on the message-level priority configuration.

For more information on Overload Support, see the *SMF Overload Support* chapter, in the *UCC 5G Session Management Function Configuration and Administration Guide*.

You can exclude session reports for emergency call, voice calls, and empty calls from throttling.

To exclude these session reports, configure the ddn delay-exclude-arplist configuration in profile sgw. If the session report is received for one of the configured ARPs, the cnSGW-C omits that session report from the session report throttling.

Feature Configuration

To configure this feature, use the following configuration:

```
config
   instance instance-id instance_id
      endpoint pfcp
         interface sxa
            overload-control msg-type session-report
            rate-limit rate_limit
            queue-size queue_size
            reject-threshold reject_threshold
            pending-request pending_request
            exit
config
   profile sgw sgw_name
      ddn delay-exclude-arplist number_priorities
      end
```

NOTES:

- **overload-control msg-type session-report**—Configure the virtual message specifications for interface overload.
- **rate-limit** *rate_limit*—Specify the rate limit for the virtual queue.
- **queue-size** *queue_size*—Specify the packet count or capacity of each virtual queue.
- **reject-threshold** *reject_threshold*—Specify the limit to reject incoming messages when this threshold percentage of pending requests is reached.
- **pending-request** *pending_request*—Specify the pending requests count in the virtual queue.
- **ddn delay-exclude-arplist** *number_priorities*—Specify the priority-level for allocation and retention priorities [1-15] that must be excluded from delaying the DDN/Session report throttling.

Configuration Example

The following is an example configuration.

```
config
   instance instance-id 1
      endpoint pfcp
         interface sxa
            overload-control msg-type session-report
            rate-limit 4500 queue-size 2500 reject-threshold 80 pending-request 2400
            exit
         profile sgw sgw1
         ddn delay-exclude-arplist [ 9 ]
         end
```

Configuration Verification

To verify the configuration:

```
#show running-config instance instance-id 1 endpoint pfcp
instance instance-id 1
endpoint pfcp
```

```
.
.
.
interface sxa
.
.
.
overload-control msg-type session-report
msg-priority high rate-limit 5 priority 1 queue-size 11 reject-threshold 80 pending-request
 10
discard-behavior reject
discard-behavior reject-code 1
exit
.
.
.
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the PFCP Session Report with DLDR Throttling Support feature.

Protocol Pod-level SXa Statistics

```
Query: sum(irate(proto_udp_req_msg_total{interface_type="SXA",
message_name=~"session_report_.*"}[1m])) by (message_name, status)
```

```
Legend: {{message_name}} | {{status}}
```

- **accepted**: Session reports accepted because of ARP configured in exclude-arp list
- **throttle_allow**: Session reports allowed by rate limit framework
- **throttled_pending_req_limit**: Session reports throttled by rate limit framework.

SGW Service-level Session Report Statistics

```
Query: sum(irate(sgw_sx_session_report_stats{}[30s])) by (sx_session_report_type,
 status)
```

```
Legend: {{sx_session_report_type}} -> {{status}}
```

Timesaver

For more information on bulk statistics support for SMF, see the *UCC 5G SMF Metrics Reference* document.

For more information on bulk statistics support for cnSGW-C, see the *UCC 5G cnSGW-C Metrics Reference* document.

Resiliency Handling

Feature Description

The Resiliency Handling feature introduces a CLI-controlled framework to support the service pod recovery, when you observe a system fault or a reported crash. It helps in recovering one of the following service pods:

- sgw-service pod
- smf-service pod
- gtpc-ep pod
- protocol pod

These service pods are software modules containing the logic to handle several session messages. The service pods are fault-prone due to any one of the following or a combination of multiple scenarios:

- Complex call flow and collision handling
- Inconsistent session state
- Incorrect processing of inbound messages against the session state
- Unexpected and unhandled content in the inbound messages

Whenever you observe the system fault or a crash, the fault behavior results into a forced restart of the service pod. It impacts the ongoing transaction processing of other sessions. The crash reoccurs even after the pod restart.

To mitigate this risk, use the CLI-based framework with actions defined to clean up subscriber sessions or terminate the current processing.

How it Works

This section describes how you can use the fault recovery framework to define actions for the crash. The framework allows you to define any of the following actions:

- Terminate—When a fault occurs, this action terminates the faulty transactions, and clears the subscriber session cache. It’s applicable for smf-service and sgw-service pods.

Note The pod doesn't get restarted. The database doesn't get cleared during this action.

- Cleanup—When a fault occurs, this action clears the faulty subscriber session and releases the call. It’s applicable for smf-service and sgw-service pods.
- Graceful reload—When a fault occurs, this action restarts the pod. It’s applicable for gtpc-ep, protocol, and pods. It handles the fault signals to clean up resources, such as the keepalive port and closes it early. It also allows the checkport script to detect the pod state and initiates the VIP switch processing for the corresponding pods.

- Reload—When the pod crashes, it initiates the reloading activity. It's a default setting or value applicable for all the pods.

Feature Configuration

To configure this feature and to enable the system fault recovery, use the following sample configuration:

```
config
   system-diagnostics {   gtp | pfcp | service | sgw-service }
      fault
         action { abort | cleanup { file-detail | interval | num | skip
 { ims | emergency | wps } } | graceful-Reload | reload }
         end
```

NOTES:

- **system-diagnostics { gtp | pfcp | service | sgw-service }**—Specify the required type of service pods for system diagnostics. The available pod options are , gtp, pfcp, smf-service, and sgw-service.
- **fault**—Enables fault recovery while processing sessions.
- **action { abort | cleanup | graceful-Reload | reload }**—Specify one of the following actions to take on fault occurrence. The default action is reload.
  - **abort**—Deletes the faulty transaction and clears its session cache. The database doesn't get cleared.

Note It's an exclusive option to the smf-service pod.

  - **cleanup { file-detail | interval | num | skip }**—Enable the cleanup activity. It has the following selections to mitigate the fault action:
    - **file-detail**—Lists the file names with line numbers. It excludes the file name details from the recovery.
    - **interval**—Specifies the duration of the interval in minutes. This duration specifies the permissible interval within which it allows the maximum number of faults. Must be an integer in the range 1–3600.
    - **num**—Specifies the maximum number of tolerable faults in an interval. Must be an integer in the range 0–50.
    - **skip { ims | emergency | wps }**—Enable the skip cleanup of a subscriber session for an active voice call, or the WPS, or an emergency call.
      - To detect the active voice calls, use the following command:

        profile dnn *dnn_name* **ims mark qci** *qos_class_id*

      - When you enable the skip cleanup configuration, the SMF deletes the faulty transaction, and clears its session cache.
      - When a fault occurs during the session setup or the release state, the SMF performs the following:

        – Deletes the transactions on the session end.

– Overrides the configured fault action during these states.

– Clears the session cache and database entries for the faulty transaction.

• It allows the dynamic configuration change.

Note It's an exclusive option to smf-service and sgw-service pods.

• **graceful-Reload**—Specify the option to gracefully reload the pod. The service pod handles fault signals to clean up resources like the keepalive port and continues with crash processing (pod restart processing).

Note It's an exclusive option to , gtpc-ep, and protocol service pods.

• **reload**—Reloads the pod, when it crashes due to a faulty behavior. It's an option applicable to all the service pods. It’s also the default option.

Configuration Example

The following example configuration allows three crashes of smf-service or sgw-service pods, within a duration of 10 minutes interval, and with the fault occurrence action as subscriber cleanup.

```
config
    system-diagnostics { service | sgw-service }
        fault
            num 3 interval 10
            action cleanup
            end
```

The following example configuration allows graceful fault handling for the or gtpc-ep pod or the protocol pod to close the keepalive port on receiving a fault signal.

Configuration Verification

To verify the configuration:

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following bulk statistics are supported for the resiliency handling feature.

recover_request_total—This statistic includes the following new labels:

• **action**—Defines the fault action.

• **reason**—Defines the fault reason.

- **status**—Defines the fault status.

The following is an example of bulk statistics for the resiliency handling feature.

```
recover_request_total{action="panic_recovery_cleanup",
app_name="SMF",cluster="Local",data_center="DC",instance_id="0",
reason="creating panic",service_name="sgw-service",status="success"} 1
```

For more information on bulk statistics support for SMF, see the *UCC 5G SMF Metrics Reference* document.

For more information on bulk statistics support for cnSGW-C, see the *UCC 5G cnSGW-C Metrics Reference* document.

Roaming Peer Path Management Optimization

Feature Description

cnSGW-C supports inbound roaming. When inbound roaming is enabled, cnSGW-C communicates with remote PGW which is located in the roamer home network.

cnSGW-C generates Echo Request messages towards the roaming peers and detects path failure, thereby handling echo request messages from the roaming peers.

How it Works

This section describes how this feature works.

cnSGW-C uses subscriber policy and operator policy to categorize peer as a roamer or a home peer. cnSGW-C applies the following functionalities to the roaming peer:

- cnSGW-C responds immediately to the echo request message received from the roaming peer. If the restart counter value changes in the echo request, cnSGW-C doesn't detect path failure towards the peer.
- cnSGW-C continues to generate echo request towards the roaming peer after reaching the configured echo interval. If the restart counter value changes in the echo response, cnSGW-C detects path failure towards the peer.
- If the restart counter value changes in the first Create Session Response message and the SGW Relocation Modify Bearer Response message, cnSGW-C detects path failure towards the peer.
- cnSGW-C doesn't update the last activity time of roaming when it receives echo request from the roaming peer.
- In the NodeMgr pod, the variable ROAMING_PEER_ECHO_MODULATOR controls the echo request generation towards the roaming peer. The default value for ROAMING_PEER_ECHO_MODULATOR is 3. cnSGW-C generates echo request towards the roaming peer after reaching ROAMING_PEER_ECHO_MODULATOR * Echo Interval.

  For example, if the ROAMING_PEER_ECHO_MODULATOR is 3, and the Echo Interval is 60, the cnSGW-C generates the Echo Request after 180 seconds. Similarly, if the ROAMING_PEER_ECHO_MODULATOR is 0, cnSGW-C doesn't generate the echo request towards the roaming peer.

- In the GTPC-EP pod, the variable GTPC_UPDATE_LAST_MSG_RECV_TIME_AFTER controls the last activity time updates. The default value for this variable is 30 seconds. The cnSGW-C updates the last activity time for the peer after 30 seconds. You can increase this value to reduce the last activity time update notifications towards the NodeMgr.

Feature Configuration

Configuring this feature involves the following steps:

- Configure the operator policy with the roaming status as roamer, and associate the operator policy with the subscriber policy to identify the operator as roaming peer. For more information, see Configuring the Operator Policy and Subscriber Policy, on page 56.
- Configure the default gateway to be used, while adding the BGP route dynamically. For more information, see Configuring the Default Gateway, on page 57.

Configuring the Operator Policy and Subscriber Policy

To configure this feature, use the following configuration:

```
config
  policy
    subscriber subscriber_name
      precedence precedence_value
        imsi mcc mcc_value
        imsi mnc mnc_value
        operator-policy policy_name
        exit
      exit
    operator operator_policy
      roaming-status roamer
      exit
  profile sgw sgw_profile_name
      subscriber-policy policy_name
      end
```

NOTES:

- **precedence** *precedence_value*—Specify the precedence for entry. Must be an integer in the range of 1-2048.
- **mcc** *mcc_value*—Specify the Mobile Country Code (MCC). Must be a three-digit number.
- **mnc** *mnc_value*—Specify the Mobile Network code (MNC). Must be a two or three-digit number.
- **operator-policy** *policy_name*—Specify the operator policy name. Must be a string.
- **policy-operator** *operator_policy*—Specify the operator policy. Must be one of the following:
  - <any string>
  - defOprPoll
  - opPolHomer

- opPolRoaming
- opPolVisiting
- opPolVisiting_hrt
- opPolVisiting_hrt_overriden

- **roaming-status roamer**—Specify the roaming status of the peer. This is disabled by default.
- **subscriber-policy** *policy_name*—Specify the subscriber policy name. Must be a string.

Configuration Example

The following is an example configuration.

```
config
   policy subscriber polSubSgw
      precedence 1
         imsi mcc 310
         imsi mnc 260
         operator-policy Home_op1
         exit

      precedence 2
         imsi mcc 311
         imsi mnc 660
         operator-policy Home_op1
         exit

      precedence 3
         imsi mcc 310
         imsi mnc 240
         operator-policy Home_op1
         exit

      precedence 4
         operator-policy Roaming_SGW_op1
         exit
      exit

      policy operator Roaming_SGW_op1
         roaming-status roamer
         exit

      profile sgw sgw1
         subscriber-policy polSubSgw
         end
```

Configuring the Default Gateway

To configure this feature, use the following configuration:

```
config
   router bgp local_as_number
   policy-name policy_name
      source-prefix source_prefix_value
      mask-range mask_range
      interface interface_id
```

```
gateWay gateway_address
end
```

NOTES:

- **router bgp** *local_as_number*—Specify the identification number for the AS for the BGP router.
- **policy-name** *policy_name*—Specify the policy name.
- **source_prefix** *source_prefix_value*—Specify the source prefix value.
- **mask-range***mask_range*—Specify the mask range.
- **gateWay** *gateway_address*—Specify the gateway address.

Configuration Example

The following is an example configuration.

```
config
   router bgp 6500
   policy-name sgw_bgp
      source-prefix 209.165.201.12/32
      mask-range 32..32
      interface ens224.2084
      gateWay 209.165.201.28
      end
```

Configuration Verification

To verify the configuration:

- **show subscriber namespace sgw imsi 123456789012345 full subscriber-details**

```
{
"subResponses": [
  "pdnInfoList": {
     "pdnInfo": [
          {
           "plmnType": "VISITOR"
           "s5ePeerType": "ROAMER"
           }
      ]
   }
]
}
```

- **show peers | include S5**

```
1 S5E 209.165.201.12:212320.20.20.124:2123Inbound nodemgr-1 Udp 2 minutes PGW
MaxRemoteRcChange: N/A,Recovery: 10 S5
1 S5E 209.165.201.12:212320.20.20.127:2123Inbound nodemgr-0 Udp 35 seconds PGW
MaxRemoteRcChange: N/A,Recovery: 10,S5E PeerType: Roaming
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the Roaming Peer Path Management Optimization feature.

gtpc-ep statistics indicating Echo Request Handling from Roaming Peer is Suppressed:

```
gtpc_roaming_peer_path_mgmt{app_name="SGW",cluster="Local",data_center="DC",
gtpc_peer_type="ROAMER",instance_id="1",interface_type="S5E",service_name="gtpc-ep",
status="suppressed"} 1
```

upd_proxy statistics indicating Total number of bgp add request:

```
# HELP upd_proxy_bgp_routes_count Total number of bgp add request
# TYPE upd_proxy_bgp_routes_count counter
upd_proxy_bgp_routes_count{app_name="SGW",cluster="Local",data_center="DC",
gr_instance_id="1",instance_id="0",service_name="udp-proxy",status="success"} 1
```

Flag DB Database Updates

Feature Description

cnSGW-C updates the CDL whenever the subscriber state changes from idle to active, and the ULI, UeTz, UCI, or the serving network is modified.

When the transaction requests driven to CDL increases, cnSGW-C incurs a higher CPU utilization. To prevent the needless CPU utilization, cnSGW-C updates only a subset of the CDL with the changed attributes.

Flag DB Database for the DDN Procedure

When the DDN procedure completes, sgw-service updates the CDL which impacts the CPU utilization. To optimize the CPU usage, the CDL is notified about the DDN only with the partial updates.

DDN Internal Timer

cnSGW-C implements the DDN Retry Timer by applying the CDL's timer functionality. Every DDN transaction starts the DDN Retry Timer that requires the complete CDL instance to be updated, which results in an increase in the CPU usage of the CDL and sgw-service.

cnSGW-C is modified to have an integrated DDN Retry Timer that is configurable from sgw-profile. With this approach, the performance is improved because the cnSGW-C does not communicate with the CDL for starting the DDN Retry Timer as it is an internal timer. The DDN Retry Timer is started for a duration of 10 seconds.

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the Flag DB Database Updates feature:

- mbr_partial_cdl_update: Captures the total number of partial CDL update procedures invoked by the Modify Bearer Request.

  Sample query:

```
sgw_cdl_update_stats{app_name="smf",cdl_update_type="
mbr_partial_cdl_update",cluster="Local",data_center="DC",gr_instance_id="1",
instance_id="0",rat_type="EUTRAN",service_name="sgw-service"} 1
```

- ddn_partial_cdl_update: Captures the total number of partial CDL update procedures that DDN has invoked

  Sample query:

```
sgw_cdl_update_stats{app_name="smf",cdl_update_type="
ddn_partial_cdl_update",cluster="Local",data_center="DC",gr_instance_id="1",
instance_id="0",rat_type="EUTRAN",service_name="sgw-service"} 1
```

For more information on bulk statistics support, see *UCC Serving Gateway Control Plane Function Metrics Reference*.

UDP Proxy Functionality Merged into Protocol Microservices

Feature Description

The UDP proxy microservices provide UDP transport termination for protocols (PFCP, GTPC, and RADIUS) that require UDP protocol as the transport layer protocol. The UDP proxy provides user space packet forwarding and IPC communication to protocol microservices. It uses host networking for source IP address observability and operates in Active-Standby mode.

Multiple protocol microservices depend on UDP proxy for UDP transport. Therefore, UDP proxy is a scaling bottleneck. A surge of messages can lead to packet drop.

The incoming and outgoing messages use the UDP proxy pod for forwarding messages. With minimal packet processing, the UDP proxy forwards the messages to the GTPC-EP pod. This requires the IPC communication for message forwarding, along with marshal or unmarshal of the packet.

The UDP proxy functionality merges into the respective protocol microservice to mitigate the scaling bottleneck. The protocol pod receives the messages directly, and avoids forwarding the messages and IPC communication.

The UDP proxy bypass improves the CPU usage by reducing one hop across microservices in the signaling path. cnSGW-C supports UDP proxy bypass for the PFCP and GTPC protocols.

PFCP Protocol Endpoint with UDP Proxy Bypass

With the UDP proxy mode, all message exchanges for the protocols, such as N4, Sxa, and GTP-U, occur through the UDP proxy. The UDP proxy is responsible for connecting or receiving connections from the UPF. The service or the UPF initiates all node-related or session-related messages, and the responses pass through the UDP proxy. The UDP proxy handles all node-related messages and forwards the messages to the protocol node.

With the outbound UDP proxy bypass mode, the session-related messages flow directly from the protocol to the UPF. The node-related messages continue to take the current path, which is through the UDP proxy to the protocol pod or the node manager.

With the inbound and outbound UDP proxy bypass mode, the service sends the session-related messages directly to UPF through the protocol pod, with UDP proxy bypassed. The protocol also establishes a connection with the UPF as and when the app service initiates a PFCP message toward the UPF.

For more information about UDP Proxy Bypass for PFCP, see *UCC 5G SMF Configuration and Administration Guide*.

GTPC Protocol Endpoint with UDP Proxy Bypass

With the UDP proxy mode, all message exchanges for the GTPv2 protocol occur through the UDP proxy. The UDP proxy is responsible for connecting or receiving connections on the S11 and S5e, S2b, and S5/S8 interface. The service or the GTP peer initiates the session-related or node-related messages, and the responses pass through the UDP proxy. The UDP proxy handles all node-related messages and forwards the messages to the protocol node.

With the inbound and outbound UDP proxy bypass mode, the service-initiated session-related messages are sent directly to the GTP peer through the GTPC-EP pod, with the UDP proxy bypassed. For node-related messages, the GTPC-EP starts a GTP endpoint for peers to connect with it on the S11, S5e, S2b, and S5/S8 interfaces. The GTPC-EP pod also establishes a connection with the GTP peer as and when the app service initiates a GTPv2 message toward the GTP peer.

The following features are integrated from UDP proxy:

- Transaction SLA
- DSCP marking for GTP packets
- Adding BGP routes for roamer subscribers on the fly
- Supporting Dispatcher feature and incoming retransmission
- SGW cache integration for DDN
- MBR cache integration

The following features are integrated from GTPC-EP:

- Retransmissions based on n3t3 configuration for outbound requests
- Monitor protocol and monitor subscriber
- Echo message handling

All existing features supporting the UDP proxy mode are supported with and without the UDP proxy bypass mode.

How it Works

This section describes how this feature works.

The GTPC-EP k8 service is disabled when the bypass feature is enabled.

The GTPC protocol endpoint with the UDP Proxy Bypass feature requires the GTPC-EP pod to run in the host environment in Active-Standby mode. When the GTPC-EP pod runs in the host environment in the Active-Standby mode, the k8-service is disabled. Further, if the pods (SGW-Service and the node manager)

must communicate with the GTPC-EP pod, an extra endpoint is required at the GTPC-EP pod. This infra endpoint initializes at the GTPC-EP app start and the internal IP is used for the same.

When the internal IP is not configured, the available GTP VIPs are used for initializing the infra endpoint.

The S11, S5, S5e, and S2b interfaces are used to configure the GTP VIPs instead of the base GTP VIP IP address.

The UDP sockets are created at the GTPC-EP pod for handling GTP packets.

No new CLI or keyword is added to enable or disable bypass UDP proxy functionality. The existing endpoint configuration is used in the following manner to enable or disable bypass UDP proxy functionality:

- GTP VIPs must be configured under the endpoint protocol for using UDP proxy (no bypass).
- GTP VIPs must be configured under the GTPC endpoint to enable bypass UDP proxy.
- If the GTP VIPs are configured under both the protocol endpoint and the GTP endpoint, the UDP proxy is used by default.
- The GTPC feature-specific configurations, such as Retransmission n3t3-based, ECHO, SLA, Dispatcher, and the DSCP must be configured under endpoint GTP irrespective of the bypass feature.

Note Prior to this feature, if endpoint GTP was configured, the UDP proxy mode was the default behaviour. With this feature onwards, if the endpoint GTP is configured with the GTP interface VIP (s5, s11, s5e, or s2b), the UDP proxy bypass is enabled by default. For UDP proxy bypass to be disabled, the endpoint protocol must be configured with the GTP interface VIP (s5, s11, s5e, or s2b).

Call flows

This section describes the key call flows for this feature.

GTPC Protocol Endpoint with UDP Proxy Bypass Call Flow

This section describes the GTPC Protocol Endpoint with UDP Proxy Bypass call flow.

Figure 11: GTPC Protocol Endpoint with UDP Proxy Bypass Call Flow

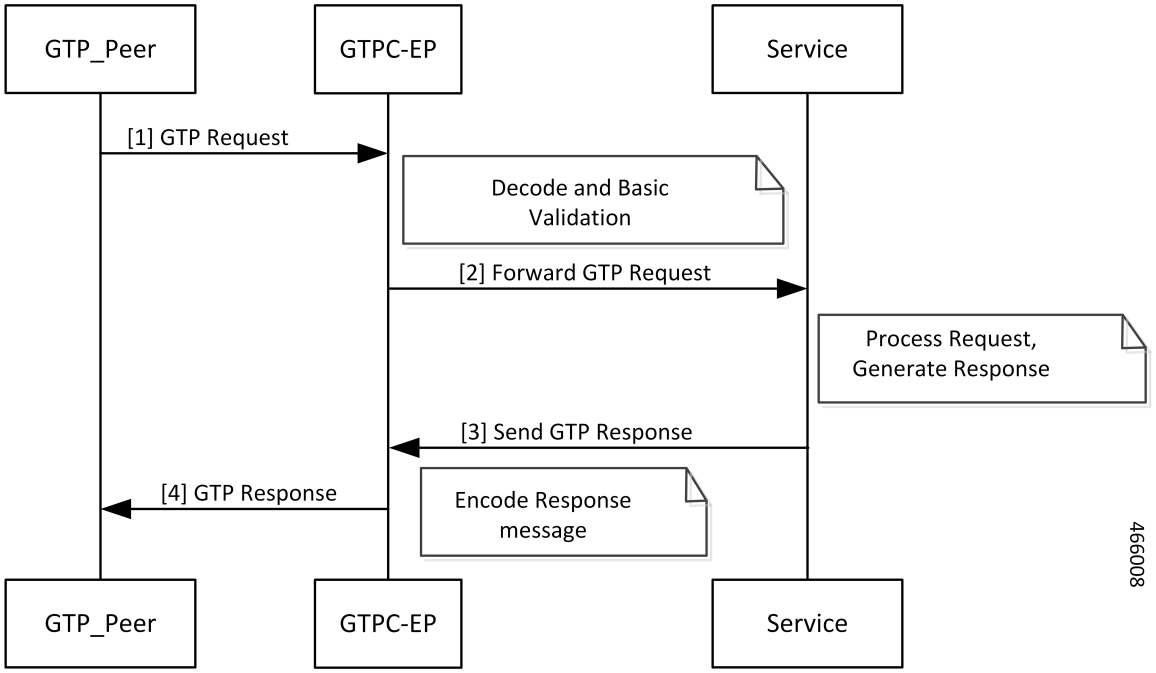


Table 10: GTPC Protocol Endpoint with UDP Proxy Bypass Call Flow Description

Step	Description
1	The GTP_Peer sends the GTP Request to the GTPC-EP pod.
2	The GTPC-EP pod decodes the message, performs the basic validation, and forwards the GTP Request to the Service.
3	The Service processes the GTP Request, generates the GTP Response, and sends the GTP Response to the GTPC-EP pod.
4	The GTPC-EP pod encodes the GTP Response message and forwards the GTP Response message to the GTP_Peer.

Feature Configuration

Configuring this feature involves the following steps:

- Configuring Internal VIP
- Configuring GTP VIPs

Configuring Internal VIP

To configure the internal VIP, use the following configuration:

```
config
  instance instance-id instance_id
    endpoint gtp
```

```
internal-vip  vip_ip_address
vip-ip ipv4_address vip-port ipv4_port_number
vip-ipv6 ipv6_address vip-ipv6-port ipv6_port_number
dual-stack-transport { true | false }
end
```

NOTES:

- **internal-vip** *vip_ip_address*—Specify the internal VIP IP address of the additional endpoint of the GTPC-EP pod. The Service pod sends the messages directly to this IP address.
- **dual-stack-transport { true | false }**—Enable the dual stack feature that allows you to specify IPv6 or IPv4 address. Specify true to enable this feature.

Configuring GTP VIPs

To configure the GTP VIPs under the interface for initializing the infra-GTP endpoint, use the following configuration:

```
config
  instance instance-id instance_id
    endpoint gtp
      vip-ip ipv4_address vip-port ipv4_port_number
      vip-ipv6 ipv6_address vip-ipv6-port ipv6_port_number
      dual-stack-transport { true | false }
    interface interface_name
      vip-ip ipv4_address vip-port ipv4_port_number
      vip-ipv6 ipv6_address vip-ipv6-port ipv6_port_number
      dual-stack-transport { true | false }
      end
```

NOTES:

- **vip-ip** *ipv4_address* **vip-port***ipv4_port_number*—Specify the IPv4 address of the interface.
- **vip-ipv6** *ipv6_address* **vip-ipv6-port***ipv6_port_number*—Specify the IPv6 address of the interface.
- **dual-stack-transport { true | false }**—Enable the dual stack feature that allows you to specify IPv6 or IPv4 address. Specify true to enable this feature.

Configuration Example

The following are example configurations.

Example configuration with the internal VIP for GTPC-EP, with UDP Proxy bypass enabled:

```
config
    instance instance-id 1
        endpoint gtp
          internal-vip 209.165.201.15
          end
```

Example configuration with the bypass feature enabled (UDP Proxy bypassed):

```
config
    instance instance-id 1
        endpoint gtp
          vip-ip 209.165.201.20
          interface s5
```

```
            vip-ip 209.165.201.20
            exit
          interface s5e
            vip-ip 209.165.201.8
            exit
          interface s2b
            vip-ip 209.165.201.20
            exit
          interface s11
            vip-ip 209.165.201.8
            end
```

Example configuration with bypass feature disabled (UDP Proxy used for GTP messages):

```
config
    instance instance-id 1
        endpoint protocol
          vip-ip 209.165.201.20
          interface s5
            vip-ip 209.165.201.20
            exit
          interface s5e
            vip-ip 209.165.201.8
            exit
          interface s2b
            vip-ip 209.165.201.20
            exit
          interface s11
            vip-ip 209.165.201.8
            end
```