Enhancements to Streaming Telemetry

This section provides an overview of the enhancements made to streaming telemetry data.

Hardware Timestamp

Table 1: Feature History Table

Feature Name	Release Information	Description
Enhancements to Hardware Timestamp	Release 7.3.4	Telemetry messages carry a timestamp per interface to indicate the time when data is collected from the hardware. With this feature, the support for hardware timestamp is extended to MPLS Traffic Engineering (MPLS TE) counters, Segment Routing for Traffic Engineering (SR-TE) interface counters, protocol statistics, and bundle protocol counters. The interface counters in the following paths are enhanced for hardware timestamp: • Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters • Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters • openconfig-network-instance:network-instances/network-instance/mpls/lsps/constrained-path/tunnels • openconfig-interfaces:interfaces/interface
Hardware Timestamp	Release 7.3.1	Whenever periodic statistics are streamed, the collector reads the data from its internal cache, instead of fetching the data from the hardware. When the data is read from the cache, the rate at which data is processed shows spikes because the timestamp from the collector is off by several seconds. With hardware timestamping, the inconsistencies that are observed when reading data from the cache file is removed.

Whenever periodic stats are streamed, the collector reads the stats from its internal cache, instead of fetching the stats from the hardware. When the data is read from the sensor paths of Stats manager cache, the rate calculation shows spikes. This behavior is due to the timestamp from the collector that is off by several seconds.

Therefore, timestamp of some other collector takes precedence because timestamps of collectors are not in synchronization with the current timestamp. This is observed when there are multiple collectors providing stats updates for the same interface.

The YANG data model for Stats manager `Cisco-IOS-XR-infra-statsd-oper.yang` is enhanced to enable the collector to read periodic stats data from the router using hardware timestamp.

The hardware timestamp is taken into account when a primary collector (for generic or proto stats) provides stats updates from the hardware to the Stats manager. With hardware timestamping in rate computation while streaming periodic stats, the spikes due to the timestamp issue is resolved.

The hardware timestamp is updated only when the collector attempts to read the counters from hardware. Else, the value remains `0`. The latest stats can be streamed at a minimum cadence of 10 seconds and periodic stats at a cadence of 30 seconds. The support is available only for physical interfaces and subinterfaces, and bundle interface and subinterfaces.

When there is no traffic flow on protocols for an interface, the hardware timestamp for the protocols is published as 0. This is due to non-synchronized timestamps sent by the collector for protocols in traffic as compared to non-traffic scenarios.

A non-zero value is published for protocols that have stats published by a primary collector for both traffic and non-traffic scenarios.

Note The hardware timestamp is supported only for primary collectors. When the hardware has no update, the timestamp will be same. However generic counters are computed for primary and non-primary collectors. The non-primary collectors show the latest stats, but not the timestamp.

When the counters are cleared for an interface using **clear counters interface** command, all counter-related data including the timestamps for the interface is cleared. After all counter values are cleared and set to `0`, the last data time is updated only when there is a request for it from a collector. For example, last data time gets updated from a collector:

```
Router#:Aug  7 09:01:08.471 UTC: statsd_manager_l[168]: Updated last data time for ifhandle
 0x02000408,
stats type 2 from collector with node 0x100, JID 250, last data time 1596790868.
INPUT: last 4294967295 updated 1596469986. OUTPUT: last 4294967295 updated 1596469986
```

All other counter values and hardware timestamp are updated when the counters are fetched from the hardware. In this case, all counters including the hardware timestamp is 0:


```
{"node_id_str":"MGBL_MTB_5504","subscription_id_str":"app_TEST_200000001",
"encoding_path":"Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters",
"collection_id":"7848",
"collection_start_time":"1596790879567",
"msg_timestamp":"1596790879571","data_json":
[{"timestamp":"1596790879570","keys":[{"interface-name":"FortyGigE0/1/0/11"}],
"content":{"packets-received":"0","bytes-received":"0","packets-sent":"0",
"bytes-sent":"0","multicast-packets-received":"0","broadcast-packets-received":"0",
"multicast-packets-sent":"0","broadcast-packets-sent":"0","output-drops":0,"output-queue-drops":0,
"input-drops":0,"input-queue-drops":0,"runt-packets-received":0,"giant-packets-received":0,
"throttled-packets-received":0,"parity-packets-received":0,"unknown-protocol-packets-received":0,
"input-errors":0,"crc-errors":0,"input-overruns":0,"framing-errors-received":0,"input-ignored-packets":0,
"input-aborts":0,"output-errors":0,"output-underruns":0,"output-buffer-failures":0,"output-buffers-swapped-out":0,
"applique":0,"resets":0,"carrier-transitions":0,"availability-flag":0,
"last-data-time":"1596790868","hardware-timestamp":"0",
"seconds-since-last-clear-counters":15,"last-discontinuity-time":1596469946,"seconds-since-packet-received":0,
"seconds-since-packet-sent":0}}],"collection_end_time":"1596790879571"}
```

Stream QoS Statistics Telemetry Data

Table 2: Feature History Table

Feature Name	Release Information	Description
Stream QoS Statistics Telemetry Data	Release 7.3.3	You can use the `Cisco-IOS-XR-qos-ma-oper.yang` data model to stream telemetry data on QoS statistics from the route processor (RP). The bundle statistics are now stored in the RP, where data is persistent, and its retrieval is unaffected by bundle member or line card failure. In earlier releases, QoS statistics was stored on line cards, and any bundle member or line card failure caused loss of statistics data.

You can collect QoS statistics for physical, virtual, bundle interfaces and subinterfaces using a push mechanism where data is streamed out of the router at a cadence. You can also collect data about ingress, egress policy-map statistics and VoQ statistics of an egress policy-map. When data is collected from the hardware, the statistics data is time-stamped.

To enable the feature, use the **hw-module profile qos qos-stats-push-collection** command in XR Config mode. You must reload the router for the configuration to take effect. To clear the QoS statistics on an interface, use **clear qos counters interface** *interface name* command in XR Exec mode. To clear the statistics for all interfaces, use **clear qos counters interface all** command.

> **Note** When the counters are cleared, the counters for SNMP statistics are also cleared.

For more information on modular QoS on link bundles, see *Modular QoS Command Reference for Cisco 8000 Series Routers*.

With this release, the interface statistics can be displayed using `Cisco-IOS-XR-qos-ma-oper.yang` data model. You can stream telemetry data from the sensor path:

```
Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics

Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics

Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface[interface-name=<interface-name>]/input/service-policy-names/service-policy-instance[service-policy-name=egress]/statistics

Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface[interface-name=<interface-name>]/output/service-policy-names/service-policy-instance[service-policy-name=egress]/statistics
```

Note This feature is not supported for all rate counters like matched-rate, transmitted-rate, dropped-rate in the QoS statistics. The bundle member statistics, location-based SPI statistics, sensor path for bundle members statistics, are not supported.

Usage Guidelines and Limitations

The feature has a limitation where only one type of QoS egress statistics, either marking or queuing, is available when multi-policy maps are configured on a bundle interface using the **show operational QoS InterfaceTable Interface/InterfaceName=Bundle-Ether100 Output Statistics** CLI command.

To view both egress statistics, you must use the alternative command **show operational QoS InterfaceTable Interface/InterfaceName=Bundle-Ether100 Output**.

The following steps show the configuration to stream data about bundle statistics to the collector.

Procedure

Step 1 Configure QoS on link bundles.

Example:

QoS Profile:

```
Router(config)# hw-module profile qos qos-stats-push-collection
Wed Dec 22 06:35:48.251 UTC
In order to activate this new qos profile, you must manually reload the chassis/all line cards
Router(config)#commit
```

Class-map:

```
Router(config)#class-map TC3
Router(config-cmap)#match traffic-class 1
Router(config-cmap)#commit
```

Policy-map:

```
Router(config)#policy-map egress
Router(config-pmap)#class TC3
Router(config-pmap-c)#shape average 1 mbps
Router(config-pmap-c)#commit
```

Step 2 Attach the service policy to an interface.

Example:

```
Router(config)#int hundredGigE 0/0/1/0.5
Router(config-if)#service-policy output egress
Router(config-if)#commit
```

Step 3 Configure telemetry subscription.

Example:

```
Router#show run telemetry model-driven
Wed Dec 22 10:27:10.846 UTC
telemetry model-driven
 destination-group destination-4
  address-family ipv4 5.16.4.114 port 51023
   encoding json
```

```
    protocol grpc no-tls
   !
  !
  sensor-group qos-grp
   sensor-path Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface
   [interface-name=HundredGigE0/0/1/0.5]/output/service-policy-names/service-policy-instance
   [service-policy-name=egress]/statistics
  !
 subscription qos-subs
   sensor-group-id qos-grp sample-interval 10000
   destination-id destination-4
  !
 !
```

Verification

Verify the QoS telemetry configuration.

Policy Map:

```
Router#show policy-map pmap-name egress detail
Wed Dec 22 06:50:53.779 UTC
class-map match-any TC3
 match traffic-class 1
 end-class-map
!
policy-map egress
 class TC3
  shape average 1 mbps
 !
 class class-default
 !
 end-policy-map
!
```

Statistics Data:

```
Router#show policy-map interface HundredGigE0/0/1/0.5 output
Wed Dec 22 08:29:18.603 UTC

HundredGigE0/0/1/0.5 output: egress

Class TC3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :               55563/55563000             0
    Transmitted         :               55401/55401000             0
    Total Dropped       :                 162/162000               0
  Queueing statistics
    Queue ID                             : 1377
    High watermark  (Unknown)
    Inst-queue-len  (Unknown)
    Avg-queue-len   (Unknown)
    Taildropped(packets/bytes)           : 162/162000
    Queue(conform)      :                    0/0                   0
    Queue(exceed)       :                    0/0                   0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :             1077710/1077710000           0
    Transmitted         :             1077710/1077710000           0
    Total Dropped       :                    0/0                   0
  Queueing statistics
```

```
    Queue ID                            : 1376
    High watermark  (Unknown)
    Inst-queue-len  (Unknown)
    Avg-queue-len   (Unknown)
    Taildropped(packets/bytes)          : 0/0
    Queue(conform)      :                  0/0                    0
    Queue(exceed)       :                  0/0                    0
Policy Bag Stats time: 1637137751027  [Local Time: 12/22/21 08:29:11.027]
```

QoS Statistics Using Telemetry:

The sample output shows the telemetry data streamed from the router:


```
{
    "node_id_str": "R1",
    "subscription_id_str": "qos-subs",
    "encoding_path":
"Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics",

    "collection_id": "21226",
    "collection_start_time": "1637144915201",
    "msg_timestamp": "1637144915203",
    "data_json": [
        {
            "timestamp": "1637144891032",
            "keys": [
                {
                    "interface-name": "HundredGigE0/0/1/0.5"
                },
                {
                    "service-policy-name": "egress"
                }
            ],
            "content": {
                "policy-name": "egress",
                "state": "active",
                "class-stats": [
                    {
                        "counter-validity-bitmask": "270532608",
                        "class-name": "TC3",
                        "cac-state": "unknown",
                        "general-stats": {
                            "transmit-packets": "239519",
                            "transmit-bytes": "239519000",
                            "total-drop-packets": "798",
                            "total-drop-bytes": "798000",
                            "total-drop-rate": 0,
                            "match-data-rate": 0,
                            "total-transmit-rate": 0,
                            "pre-policy-matched-packets": "240317",
                            "pre-policy-matched-bytes": "240317000"
                        },
                        "queue-stats-array": [
                            {
                                "queue-id": 1377,
                                "tail-drop-packets": "798",
                                "tail-drop-bytes": "798000",
                                "queue-drop-threshold": 0
"forced-wred-stats-display": false,
                                "random-drop-packets": "0",
                                "random-drop-bytes": "0",
                                "max-threshold-packets": "0",
                                "max-threshold-bytes": "0",
                                "conform-packets": "0",
                                "conform-bytes": "0",
```

```
                                "exceed-packets": "0",
                                "exceed-bytes": "0",
                                "conform-rate": 0,
                                "exceed-rate": 0
                            }
                        ]
                    },
                    {
                        "counter-validity-bitmask": "270532608",
                        "class-name": "class-default",
                        "cac-state": "unknown",
                        "general-stats": {
                            "transmit-packets": "4661952",
                            "transmit-bytes": "4661952000",
                            "total-drop-packets": "0",
                            "total-drop-bytes": "0",
                            "total-drop-rate": 0,
                            "match-data-rate": 0,
                            "total-transmit-rate": 0,
                            "pre-policy-matched-packets": "4661952",
                            "pre-policy-matched-bytes": "4661952000"
                        },
 "queue-stats-array": [
                            {
                                "queue-id": 1376,
                                "tail-drop-packets": "0",
                                "tail-drop-bytes": "0",
                                "queue-drop-threshold": 0,
                                "forced-wred-stats-display": false,
                                "random-drop-packets": "0",
                                "random-drop-bytes": "0",
                                "max-threshold-packets": "0",
                                "max-threshold-bytes": "0",
                                "conform-packets": "0",
                                "conform-bytes": "0",
                                "exceed-packets": "0",
                                "exceed-bytes": "0",
"conform-rate": 0,
                                "exceed-rate": 0
                            }
                        ]
                    }
                ],
                "satid": 0,
                "policy-timestamp": "1637144891032"
            }
        }
    ],
    "collection_end_time": "1637144915203"
}
```

Enhanced Syslog Notifications for Unresolved Line Card Forwarding Paths

Table 3: Feature History Table

Feature Name	Release Information	Description
Enhanced Syslog Notifications for Unresolved Line Card Forwarding Paths	Release 7.5.2Release 7.3.3	This feature notifies you of Line Card and Route Processor paths not resolving in the Forwarding Information Base. Both Model-Driven Telemetry (MDT) and Event Driven Telemetry (EDT) notifications are supported. In earlier releases, notifications for route processors were supported. This feature provides for improved diagnostics.

Telemetry now supports syslog notification from line cards. This is in addition to the existing notification support from route processors. You will be notified of line card and route processor paths not resolving in the Forwarding Information Base (FIB), through MDT and EDT notifications.

MDT is configured for cadence-based telemetry, while EDT is configured for event-based notification. Notifications are generated only when the device goes into error or OOR state, and during device recovery. Errors and OOR are tracked for a device as a whole, and not for individual nodes. The IPv4 Error, IPv6 Error, IPv4 OOR, and IPv6 OOR telemetry notifications are supported.

The following notification is an example of IPv4 error state, if a line card and route processor paths do not resolve in the FIB:

```
GPB(common) Message
[5.13.9.177:38418(PE1)/Cisco-IOS-XR-fib-common-oper:oc-aft-l3/protocol/ipv4/error/state msg
 len: 168]
{
    "Source": "5.13.9.177:38418",
    "Telemetry": {
        "node_id_str": "PE1",
        "subscription_id_str": "Sub2",
       "encoding_path": "Cisco-IOS-XR-fib-common-oper:oc-aft-l3/protocol/ipv4/error/state",

        "collection_id": 243,
        "collection_start_time": 1637858634881,
        "msg_timestamp": 1637858634881,
        "collection_end_time": 1637858634883
    },
    "Rows": [
        {
            "Timestamp": 1637858634882,
            "Keys": null,
            "Content": {
                "is-in-error-state": "true"
            }
        }
```

```
    ]
}
```

Note The parameters denote`"Content": {"is-in-error-state": "true"}` that the system is in error state.

The following notification is an example of IPv4 OOR, if a line card and route processor are in OOR state:


```
GPB(common) Message
[5.13.9.177:50146(PE1)/Cisco-IOS-XR-fib-common-oper:oc-aft-l3/protocol/ipv4/oor/state msg
len: 163]
{
    "Source": "5.13.9.177:50146",
    "Telemetry": {
        "node_id_str": "PE1",
        "subscription_id_str": "Sub1",
        "encoding_path": "Cisco-IOS-XR-fib-common-oper:oc-aft-l3/protocol/ipv4/oor/state",
        "collection_id": 11,
        "collection_start_time": 1637815892624,
        "msg_timestamp": 1637815892624,
        "collection_end_time": 1637815892626
    },
    "Rows": [
        {
            "Timestamp": 1637815892625,
            "Keys": null,
            "Content": {
                "is-in-oor-state": "true"
            }
        }
    ]
}
```


Note The parameters denote`"Content": {"is-in-oor-state": "true"}` that the system is in OOR state.

Target-Defined Mode for Cached Generic Counters Data

Table 4: Feature History Table

Feature Name	Release Information	Description
Target-Defined Mode for Cached Generic Counters Data	Release 7.5.1 Release 7.3.3	This feature streams telemetry data for cached generic counters using a TARGET_DEFINED subscription. This subscription ensures that any change to the cache streams the latest data to the collector as an event-driven telemetry notification. This feature introduces support for the following sensor path: `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters`

Streaming telemetry pushes the subscribed data from the router to one or more collectors. The telemetry infrastructure retrieves the data from the system database when you send a subscription request. Based on the subscription request or the telemetry configuration the cached generic counters data can be retrieved periodically based on the sample-interval. Data, such as interface statistics, is cached and refreshed at certain intervals. The `TARGET_DEFINED` subscription mode can be used to retrieve data when the cache gets updated, and is not based on a timer.

The application can register as a data producer with the telemetry library and the SysdB paths it supports. One of the data producers, Statsd, uses the library with a `TARGET_DEFINED` subscription mode. As part of this mode, the producer registers the sensor paths. The statistics infrastructure streams the incremental updates for statsd cache sensor path `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters`. With this path in the subscription, whenever cache is updated, the statsd application pushes the updates to the telemetry daemon. The daemon sends these incremental updates to the collector. The cache updates are pushed for physical interfaces, physical subinterfaces, bundle interfaces, and bundle subinterfaces. You can subscribe to the sensor path for the cached generic counters with `TARGET_DEFINED` mode instead of the sensor path for the latest generic counters (`Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`) to reduce the system load.

Configure the router to stream telemetry data from cache for generic counters using the following instructions:

Create a `TARGET_DEFINED` subscription mode for cached generic counters using one of the two options:

- **Option 1:** gRPC Network Management Interface (gNMI) subscribe request

```
{
    "name": "SubscribeRequest",
    "subscribe": {
        "prefix": {"origin":
                "Cisco-IOS-XR-infra-statsd-oper"

              },
         "mode": "STREAM", "encoding": "PROTO", "updates_only": "false",
        "subscription": [
         { "path": {"elem": [ {"name":"infra-statistics"},
                              {"name":"interfaces"},
                              {"name":"interface"},
                              {"name":"cache"},
                              {"name":"generic-counters"}
                           ]
                },
           "mode":   "TARGET_DEFINED"
         }
                     ]
    }
}
```

- **Option 2:** Model-driven telemetry configuration for non-gNMI requests

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription sub1
Router(config-model-driven-subs)#sensor-group-id grp1 mode target-defined
Router(config-model-driven-subs)#source-interface Interface1
Router(config-model-driven-subs)#commit
```

After the subscription is triggered, updates to the stats cache are monitored. The statsd application pushes the cached generic counters to the client (collector).

View the number of incremental updates for the sensor path.

```
Router#show telemetry model-driven subscription .*
Fri Nov 12 23:36:27.212 UTC
Subscription: GNMI__16489080148754121540
-------------
   Collection Groups:
    ------------------
      Id: 1
      Sample Interval:      0 ms   (Incremental Updates)
      Heartbeat Interval:   NA
      Heartbeat always:     False
      Encoding:             gnmi-proto
      Num of collection:    1
      Incremental updates:  12
      Collection time:      Min:     5 ms Max:     5 ms
      Total time:           Min:     6 ms Avg:     6 ms Max:     6 ms
      Total Deferred:       1
      Total Send Errors:    0
      Total Send Drops:     0
      Total Other Errors:   0
      No data Instances:    0
      Last Collection Start:2021-11-12
                23:34:27.1362538876 +0000
      Last Collection End:  2021-11-12   23:34:27.1362545589
              +0000
      Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/
                            interface/cache/generic-counters
```

In this example, the incremental updates of `12` indicates that the cache is updated 12 times.

You can also retrieve the detailed operational data about the subscription using the following command. In this example, `statsd-target` is the subscription name.

```
Router#show telemetry model-driven subscription statsd-target internal
Fri Nov 12 08:51:16.728 UTC
Subscription: statsd-target
-------------
State: ACTIVE
Sensor groups:
Id: statsd
Sample Interval: 0 ms (Incremental Updates)
Heartbeat Interval: NA
Sensor Path: Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/
             generic-counters
Sensor Path State: Resolved

Destination Groups:
Group Id: statsd-target
Destination IP: 192.0.2.1
Destination Port: 56000
Encoding: json
Transport: grpc
State: Active
TLS : False
Total bytes sent: 623656
Total packets sent: 13
Last Sent time: 2021-08-16 08:51:15.1304821089 +0000

Collection Groups:
------------------
Id: 2
Sample Interval: 0 ms (Incremental Updates)
Heartbeat Interval: NA
Heartbeat always: False
```

```
Encoding: json
Num of collection: 1
Incremental updates: 3
Collection time: Min: 94 ms Max: 94 ms
Total time: Min: 100 ms Avg: 100 ms Max: 100 ms
Total Deferred: 0
Total Send Errors: 0
Total Send Drops: 0
Total Other Errors: 0
No data Instances: 0
Last Collection Start:2021-08-16 08:51:04.1293895665 +0000
Last Collection End: 2021-08-16 08:51:04.1293996284 +0000
```

The sample interval of `0` indicates that the data is streamed whenever an event occurs. Here, the event represents the updates to the cache state.

Related Commands:

- **show tech telemetry model-driven**
- **show running-config telemetry model-driven**
- **show telemetry producers trace** *producer name* **info**
- **show telemetry producers trace** *producer name* **err**

FQDN as the gRPC Tunnel Destination

Table 5: Feature History Table

Feature Name	Release Information	Description
FQDN as the gRPC Tunnel Destination	Release 7.10.1	You can now specify a fully qualified domain name (FQDN) as gRPC tunnel destination. FQDNs are easy to remember compared to numeric IP addresses and helps to resolve the domain names to IPv4 or IPv6 address and establish tunnel towards the destination. Model-driven telemetry and Event-driven telemetry are supported. The feature introduces these changes: **CLI:** • The keywords **address-family ipv4** and **address-family ipv6** are introduced in the **gRPC tunnel** command.

Starting Cisco IOS XR Software Release 7.10.1, you can configure a FQDN as the gRPC tunnel destination. A FQDN can be associated with multiple IP addresses of IPv4 and IPv6. Configuration address-family specifies the desired address-family for the returned addresses from DNS. This configuration is applicable only to FQDN as gRPC tunnel destination. The `get addr-info` returns the corresponding address(es) of the FQDN. If the **address-family ipv4** is configured, then only IPv4 address(es) are used to establish gRPC tunnel(s). Similarly, if the **address-family ipv6** is configured, only IPv6 address(es) are used to establish gRPC tunnel(s). If address-family configuration is not present, then IP addresses of both IPv4 and IPv6 address-family are considered for establishing gRPC tunnel(s).

For more information about gNMI dial-out via gRPC tunnel, see the Github repository.

Configure FQDN as the gRPC Tunnel Destination

Perform the following steps to configure FQDN as gRPC tunnel destination:

Procedure

Step 1 Configure FQDN as gRPC tunnel destination.

Example:

In this example, you set up FQDN as gRPC tunnel destination (IPv4):

```
Router#config
Router(config)#grpc
Router(config-grpc)#tunnel
Router(config-grpc-tunnel)#destination test.tunnel.dn port 59500
Router(config-grpc-tunnel-dest)#address-family ipv4
Router(config-grpc-tunnel-dest)#target xr
Router(config-grpc-tunnel-dest)#commit
```

Example:

In this example, you set up FQDN as gRPC tunnel destination (IPv6):

```
Router#config
Router(config)#grpc
Router(config-grpc)#tunnel
Router(config-grpc-tunnel)#destination test.tunnel.dn port 59500
Router(config-grpc-tunnel-dest)#address-family ipv6
Router(config-grpc-tunnel-dest)#target xr
Router(config-grpc-tunnel-dest)#commit
```

Step 2 Verify that the gRPC tunnel configuration is successful on the router.

Example:

The following example shows how to display the successful gRPC tunnel configuration (IPv4):

```
Router#show running-config grpc

grpc
tunnel
  destination test.tunnel.dn port 59500
   target xr
   address-family ipv4
  !
!
!
```

Example:

The following example shows how to display the successful gRPC tunnel configuration (IPv6):

```
Router#show running-config grpc

grpc
tunnel
  destination test.tunnel.dn port 59500
   target xr
   address-family ipv6
  !
!
!
```

Step 3 Verify the configured FQDN.

Example:

The following example shows how to display the successful FQDN configuration (IPv4 and IPv6):

```
Router#show host

Default domain is not set
Name/address lookup uses domain service
Name servers: 255.255.255.255
Host                       Flags    Age(hr) Type Address(es)
test.tunnel.dn           (perm, OK)   0      IP  192.168.0.2
test.tunnel.dn           (perm, OK)   0      IPV6  2001:DB8:A:B::1
```

```
Router#show running-config domain

domain ipv4 host test.tunnel.dn 192.168.0.2
domain ipv6 host test.tunnel.dn 2001:DB8:A:B::1
```

Step 4 Copy the public certificate for the collector to `/misc/config/grpc/<domain name>:<port>.pem` directory. The router uses this certificate to verify the tunnel server, and establish a dial-out session.

Step 5 Run the collector.

Step 6 View the status of tunnel sessions.

Example:

The following example shows how to display the status of the tunnel sessions:

```
Router#show grpc tunnel sessions

test.tunnel.dn:59500
Target:             xr
Status:             Connected
Listen address:     192.168.0.25:59500
Last connected:     2023-02-06 07:14:36

test.tunnel.dn:59500
Target:             xr
Status:             Connected
Listen address:     [2001:DB8:A:B::11]:59500
Last connected:     2023-02-06 07:15:23
```

Stream Telemetry Data about PBR Decapsulation Statistics

Table 6: Feature History Table

Feature Name	Release Information	Description
Stream Telemetry Data about PBR Decapsulation Statistics	Release 7.3.2	This feature streams telemetry data about header decapsulation statistics for traffic that uses the Policy-Based Routing (PBR) functionality to bypass a routing table lookup for egress. You use the `Cisco-IOS-XR-infra-policymgr-oper.yang` data model to capture the decapsulation data for Generic Routing Encapsulation (GRE) and Generic UDP Encapsulation (GUE) tunneling protocols. Decapsulation data helps you understand if all encapsulated packets are decapsulated and alerts you to issues if there is a mismatch in the number of packets.

You can stream telemetry data about PBR decapsulation statistics for GRE and GUE encapsulation protocols that deliver packets using IPv4 or IPv6. The encapsulated data has source and destination address that must match with the source and destination address in the classmap. Both encapsulation and decapsulation interfaces collect statistics periodically. The statistics can be displayed on demand using **show policy-map type pbr [vrf vrf-name] address-family ipv4/ipv6 statistics** command. For more information on PBR-based decapsulation, see *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*.

With this release, the decapsulation statistics can be displayed using `Cisco-IOS-XR-infra-policymgr-oper.yang` data model and telemetry data. You can stream telemetry data from the sensor path:

`Cisco-IOS-XR-infra-policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type/vrf-table/vrf/afi-table/afi/stats`

The following steps show the PBR configuration and the decapsulation statistics that is streamed as telemetry data to the collector.

Procedure

Step 1 Check the running configuration to view the configured PBR per VRF.

Example:

```
Router#show running-config
Building configuration...
!! IOS XR Configuration 0.0.0
!!
vrf vrf1
 address-family ipv4 unicast
```

```
 !
 address-family ipv6 multicast
 !
!
netconf-yang agent
 ssh
!
!
class-map type traffic match-all cmap1
 match protocol gre
 match source-address ipv4 161.0.1.1 255.255.255.255
 match destination-address ipv4 161.2.1.1 255.255.255.255
 end-class-map
!
policy-map type pbr gre-policy
 class type traffic cmap1
  decapsulate gre
 !
 class type traffic class-default
 !
 end-policy-map
!
interface GigabitEthernet0/0/0/1
 vrf vrf1
 ipv4 address 2.2.2.2 255.255.255.0
 shutdown
!
vrf-policy
 vrf vrf1 address-family ipv4 policy type pbr input gre-policy
!
end
```

Step 2 View the output of the VRF statistics.

Example:

```
Router#show policy-map type pbr vrf vrf1 addr-family ipv4 statistics

VRF Name:       vrf1
Policy-Name:    gre-policy
Policy Type:    pbr
Addr Family:    IPv4

Class:     cmap1
     Classification statistics      (packets/bytes)
       Matched             :       13387587/1713611136
     Transmitted statistics         (packets/bytes)
       Total Transmitted   :       13387587/1713611136

Class:     class-default
     Classification statistics      (packets/bytes)
       Matched             :              0/0
     Transmitted statistics         (packets/bytes)
       Total Transmitted   :              0/0
```

After you have verified that the statistics are displayed correctly, stream telemetry data and check the streamed data at the collector. For more information about collectors, see *Operate on Telemetry Data for In-depth Analysis of the Network* section in the Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure chapter.

```
ios.0/0/CPU0/ $ mdt_exec -s Cisco-IOS-XR-infra-policymgr-oper:policy-manager
/global/policy-map/policy-map-types/policy-map-type/vrf-table/vrf/afi-table/afi/stats -c 100
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"Cisco-IOS-XR-infra-policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type
/vrf-table/vrf/afi-table/afi/stats","collection_id":"1","collection_start_time":"1601361558157",
```

```
"msg_timestamp":"1601361559179","data_json":[{"timestamp":"1601361559178","keys":[{"type":"ipv6"},
{"vrf-name":"vrf_gue_ipv4"},{"type":"ipv4"}],"content":{"pmap-name":"gre-policy","vrf-name":
"vrf1","appln-type":2,"addr-family":1,"rc":0,"plmgr-vrf-stats":[{"pmap-name":"gre-policy",
"cmap-stats-arr":[{"cmap-name":"cmap1","matched-bytes":"1713611136","matched-packets":"13387587",
"transmit-bytes":"1713611136","transmit-packets":"13387587"}]}]}}}],
"collection_end_time":"1601361559183"}
---------------------------------  snipped for brevity -----------------------------------
```


Stream Telemetry Data for LLDP Statistics

Table 7: Feature History Table

Feature Name	Release Information	Description
Stream Telemetry Data for LLDP Statistics	Release 24.2.11	You can now oversee and diagnose your network infrastructure in real time by periodically streaming the Link Layer Discovery Protocol (LLDP) information of a router through a gRPC Network Management Interface (gNMI) client. By continuously monitoring LLDP data from a switch or router, you gain immediate insights into network topology and the attributes of devices on the network, facilitating proactive management and troubleshooting.

Analyze Network Infrastructure Using LLDP Telemetry Data

Starting from Release 24.2.11, you can monitor Link Layer Discovery Protocol (LLDP) advertisement information, such as device identification, port identification, system capabilities, and management address using the gNMI `subscribe` operation.

Using the gNMI `subscribe` operation, you can subscribe to a stream of updates for specific paths within the device's data model. The following table lists the different subscription modes.

Table 8: Subscription Modes in a gNMI `Subscribe` Operation and Their Description

Subscription Mode	Description
Once	A one-time retrieval of the data at the specified paths.
Poll	Sets up a repeated send of the data at the specified paths at the input interval
Stream	Enables a continuous stream of updates from the server as changes occur. Within Stream mode, there are two types of subscriptions: • ◦ **Sample**: The server periodically sends the current state of the subscribed paths at a defined interval. • **On Change**: The server sends updates only when the values of the subscribed paths change.

gNMI Sensor Path to Stream LLDP Telemetry

You can stream telemetry data from the following gNMI sensor paths using **On Change** subscription mode:

- `openconfig-lldp:lldp/state`
- `openconfig-lldp:lldp/interfaces/interface/state`
- `openconfig-lldp:lldp/interfaces/interface/neighbors/neighbor`

Configure Telemetry for LLDP Statistics

Stream LLDP updates from a router or switch directly through the gNMI client.

Procedure

Step 1 Configure gNMI and LLDP

Example:

```
Router# config
Router(config)# grpc gnmi port 1024
Router(config)# lldp
Router(config)# commit
```

Step 2 Configure gRPC Request message on the gNMI Client

Example:

```
"openconfig-lldp:lldp":
  {
    "state":{}
  }
}
```

The gNMI client initiates a gRPC request message to get the latest LLDP information from the router.

Step 3 Verify gNMI response from the Router to the Client

Example:

```
"openconfig-lldp:lldp":
  {
    "state":
    {
      "enabled":true,
      "hello-timer":"30",
      "system-name":"ios",
      "system-description":"Cisco8k",
      "chassis-id":"abcd.abcd.abcd",
      "chassis-id-type":"MAC_ADDRESS",
      "counters":
      {
          "frame-in":"0",
          "frame-out":"0",
          "frame-error-in":"0",
          "frame-discard":"0",
          "tlv-discard":"0",
          "tlv-unknown":"0",
          "last-clear":"2024-06-05T16:09:36.388+05:30",
```

```
            "tlv-accepted":"0",
            "entries-aged-out":"0"
          }
        }
      }
    }
```


Stream Telemetry Data for ASIC Error Statistics

Table 9: Feature History Table

Feature Name	Release Information	Description
Stream Telemetry Data for ASIC Error Statistics	Release 24.2.11	You can now stream and monitor the telemetry data remotely on a gNMI interface, after subscribing to a sensor path. This data is gathered directly from the Network Processor Unit (NPU) driver at regular, predefined intervals for each block. This streaming enables real-time monitoring and analysis of router health and network performance, including error reporting and key metrics, allowing for rapid response to dynamic network conditions. Previously, you needed to log into the router to check the ASIC statistics.

Analyze Network Performance Using ASIC Telemetry Data

Starting from Cisco IOS XR Release 24.2.11, you can stream telemetry data of Application-Specific Integrated Circuit (ASIC) error statistics through a gRPC Network Management Interface (gNMI). The ASIC telemetry capabilities enable precise monitoring of individual ASICs within the router. To analyze network performance, you can stream telemetry data from various subsystems, including the ASIC Interface, Queueing, Lookup, Host Interface, and Fabric Interface, and examine their error counters.

The telemetry data is streamed for a particular ASIC within a router by using its comprehensive packet, drop, and error counters. Telemetry data is collected for the following critical errors through gNMI.

- All reset action interrupts
- Link Errors
- Memory corruption interrupts multi bit error (MBE), single bit error (SBE), and parity (Error detection).
- Packet loss interrupts on threshold reached

The error counters allow for a granular assessment of the router's operational health, with the data which is collected directly from the Network Processing Unit (NPU) driver at a specified frequency for each block.

ASIC Error Types Tracked Using gNMI

Telemetry data is collected for the following types of critical errors through gNMI:

Error Type	Details
Reset action interrupts	These interrupts includes the following types of reset actions: • Hard Reset—It doesn't stop any processes but reinitializes the underlying ASIC. • PON Reset—The power is switched OFF and switched ON. • Replace Device—The board gets shut down.
Link Errors	These are malfunctions or disruptions in the data connections.
Memory corruption interrupts (Multibit error (MBE), Single-bit error (SBE), Parity (Error detection))	These result in unintended alterations to the data in the chip's memory, causing unexpected behavior or failures.
Packet loss interrupts on threshold reached	These interrupts trigger an alert when there's a potential drop of packets. You can configure the threshold duration to three minutes or five minutes. When there's a potential drop of packets in this configuration duration continuously, the syslog is updated and notified to the user.

gNMI Sensor Path to Stream ASIC Error Telemetry

You can stream telemetry data from the gNMI sensor path:
`openconfig-platform:components/component/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors`.

Configure Telemetry for ASIC Error Statistics

Procedure

Step 1 Verify your gRPC configuration by running the **show running-config grpc** command.

Example:

```
Router# show running-config grpc
Wed May 15 01:10:57.595 UTC
grpc
 port 57400
 no-tls
 max-streams 128
 max-streams-per-user 128
 address-family dual
 service-layer
 !
 local-connection
 max-request-total 256
```

```
 max-request-per-user 32
!
```

Step 2 View the output of the ASIC interrupts statistics, stream the telemetry data for the following ASIC Errors:

- View Reset Action Interrupts, on page 22
- View Link Errors, on page 23
- View Memory Corruption Interrupts, on page 25
- View Packet Loss Interrupts, on page 26

View Reset Action Interrupts

To view Reset action interrupts on the router, use the **show asic-errors all detail location** command.

```
Router# show asic-errors all detail location 0/0/cpu0
Thu May 16 09:28:57.001 UTC
************************************************************
*                          0_0_CPU0                        *
************************************************************
************************************************************
*                   NPU ASIC Error Summary                 *
************************************************************
************************************************************
*                       Instance : 0                       *
************************************************************
************************************************************
*                        Reset Errors                      *
************************************************************
8000, 88-LC0-36FH, 0/0/CPU0, npu[0]
Name             : hard-reset
Block ID         : 0x6
Addr             : 0x100
Leaf ID          : 0xc02002
Thresh/period(s): 5/day
Error count      : 1
Last clearing    : Thu May 16 09:27:42 2024
Last N errors    : 1
--------------------------------------------------------------
First N errors.
@Time, Error-Data
------------------------------------------
May 16 09:27:42.942652
 Error description: Id:169 Bit:0x2 Action:hard-reset OTHER: NTS-09:27:42.942729
STS-09:27:42.942652
--------------------------------------------------------------
```

gNMI Output for ASIC Reset Action Errors

Following is the corresponding gNMI output for the `Reset Action` ASIC errors after subscribing to the reset action interrupt:


```
{
  "source": "1.1.111.3:57400",
  "subscription-name": "default-1715847417",
  "timestamp": 1715851665318438321,
  "time": "2024-05-16T02:27:45.318438321-07:00",
  "prefix": "openconfig-platform:",
```

```
  "updates": [
    {
      "Path":
"components/component[name=0/0/CPU0-NPU0]/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors",

      "values": {

"components/component/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors":
 {
          "queueing-block": {
            "queueing-block-error": {
              "name": "dics.general_interrupt_register.dics2mmu_fifo_overflow",
              "state": {
                "action": [
                  "NPU_RESET"
                ],
                "count": "1",
                "level": "MAJOR",
                "name": "dics.general_interrupt_register.dics2mmu_fifo_overflow",
                "threshold": "0"
              }
            }
          }
        }
      }
    }
  ]
}
```


View Link Errors

To view link error statistics, use the **show asic-errors all detail location** command.

```
Router# show asic-errors all detail location 0/rp0/cpu0
Thu May 16 14:28:38.474 UTC
************************************************************
*                        0_RP0_CPU0                        *
************************************************************
************************************************************
*                  NPU ASIC Error Summary                  *
************************************************************
…
…
…
************************************************************
*                      Instance : 6                        *
************************************************************
************************************************************
*                       Reset Errors                       *
************************************************************
************************************************************
*                    Single Bit Errors                     *
************************************************************
************************************************************
*                   Multiple Bit Errors                    *
************************************************************
************************************************************
*                      Parity Errors                       *
************************************************************
************************************************************
*                    Unexpected Errors                     *
************************************************************
************************************************************
*                       Link Errors                        *
```

```
************************************************************
8000, 8804-FC0, 0/FC6, npu[6]
Name            : slice[5].ifg[0].mac_pool8[1].rx_link_status_down.rx_link_status_down4
Block ID        : 0x21c
Addr            : 0x100
Leaf ID         : 0x43802004
Thresh/period(s): 100/day
Error count     : 2
Last clearing   : Thu May 16 14:27:32 2024
Last N errors   : 2
--------------------------------------------------------------
First N errors.
@Time, Error-Data
------------------------------------------
May 16 14:27:32.282300
 Error description: Id:49 Bit:0x4 Action:port-reset LINK_BAD: (Slc/Ifg/FstSer):(5/0/12)
NTS-14:27:32.282423 STS-14:27:32.282300
May 16 14:27:32.342088
 Error description: Id:50 Bit:0x4 Action:none LINK_DN: (Slc/Ifg/FstSer):(5/0/12) RxLnDnSt:1
 RxRmtLnStDn:0 PcsLnStDn:1 PcsAlnStDn:1 HiBer:0 HiSerIntr:0 SigLos:0 1 1 0 0 0 0 0
FifoOverflow:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  Link:220
NTS-14:27:32.342427 STS-14:27:32.342088 TEMP_SENSOR:44.0C
--------------------------------------------------------------
```

gNMI Output for ASIC Link Errors

Following is the corresponding gNMI output for the `Link errorr` ASIC errors after subscribing to the Link error interrupt:

```
{
  "source": "1.1.111.3:57400",
  "subscription-name": "default-1715865468",
  "timestamp": 1715869652342383198,
  "time": "2024-05-16T07:27:32.342383198-07:00",
  "prefix": "openconfig-platform:",
  "updates": [
    {
      "Path":
"components/component[name=0/FC6-NPU6]/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors",

      "values": {

"components/component/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors":
 {
          "interface-block": {
            "interface-block-error": {
            "name": "slice[5].ifg[0].mac_pool8[1].rx_link_status_down.rx_link_status_down4",

              "state": {
                "action": [
                  "LOG"
                ],
                "count": "1",
                "level": "MINOR",
                "name":
"slice[5].ifg[0].mac_pool8[1].rx_link_status_down.rx_link_status_down4",
                "threshold": "0"
              }
            }
          }
        }
      }
    }
```

```
  ]
}
```

View Memory Corruption Interrupts

To view memory corruption statistics, use the **show asic-errors all detail location** command.

```
Router# show asic-errors all detail location 0/0/cpu0
Thu May 16 20:33:49.135 UTC
************************************************************
*                         0_0_CPU0                         *
************************************************************
************************************************************
*                   NPU ASIC Error Summary                 *
************************************************************
************************************************************
*                       Instance : 0                       *
************************************************************
************************************************************
*                        Reset Errors                      *
************************************************************
************************************************************
*                     Single Bit Errors                    *
************************************************************
8000, 88-LC0-36FH, 0/0/CPU0, npu[0]
Name            : slice[0].ifg[0].sch.vsc_token_bucket_cfg
Block ID        : 0xc1
Addr            : 0x400000
Leaf ID         : 0x4000800
Thresh/period(s): 100/day
Error count     : 1
Last clearing   : Thu May 16 20:32:11 2024
Last N errors   : 1
--------------------------------------------------------------
First N errors.
@Time, Error-Data
-----------------------------------------
May 16 20:32:11.966346
 Error description: Id:124 Bit:0x0 Action:none MEM_PROTECT: Err:0x0 Inst_Addr:0x400000
Entry:0x0 NTS-20:32:11.966447 STS-20:32:11.966346
--------------------------------------------------------------
```

gNMI Output for ASIC Memory Corruption Interrupt Errors

Following is the corresponding gNMI output for the `Memory Corruption` ASIC errors after subscribing to the memory corruption interrupt:


```
{
  "source": "1.1.111.3:57400",
  "subscription-name": "default-1715887309",
  "timestamp": 1715891533779872124,
  "time": "2024-05-16T13:32:13.779872124-07:00",
  "prefix": "openconfig-platform:",
  "updates": [
    {
      "Path":
"components/component[name=0/0/CPU0-NPU0]/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors",

      "values": {

"components/component/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors":
 {
          "queueing-block": {
```

```
                "queueing-block-error": {
                  "name": "slice[0].ifg[0].sch.vsc_token_bucket_cfg_ecc1b",
                  "state": {
                    "action": [
                      "LOG"
                    ],
                    "count": "1",
                    "level": "MINOR",
                    "name": "slice[0].ifg[0].sch.vsc_token_bucket_cfg_ecc1b",
                    "threshold": "0"
                  }
                }
              }
            }
          }
        }
      ]
    }
```

View Packet Loss Interrupts

Prerequisites

Configure the packet loss beyond threshold level using the **hw-module profile packet-loss-alert** commad.

Monitor Packet Loss Interrupts

To view packet loss statistics, use the **show asic-errors npu 0 generic location** command.

```
Router# sh asic-errors npu 0 generic location 0/0/CPU0
Fri May 17 04:15:04.569 UTC
************************************************************
*                          0_0_CPU0                        *
************************************************************
************************************************************
*                       Generic Errors                     *
************************************************************
8000, 88-LC0-36FH-M, 0/0/CPU0, npu[0]
Name            : slice[0].tx.pdr.general_interrupt_register.mc_flb_to_uc_oq
Block ID        : 0x1f
Addr            : 0x100
Leaf ID         : 0x3e02000
Thresh/period(s): 1/day
Error count     : 36
Last clearing   : Fri May 17 03:52:12 2024
Last N errors   : 36
--------------------------------------------------------------
First N errors.
@Time, Error-Data
------------------------------------------
May 17 03:52:12.018091
        Error description: Id:198 Bit:0x0 Action:packet-loss OTHER: NTS-03:52:12.018236
STS-03:52:12.018091
May 17 03:52:17.023359
        Error description: Id:200 Bit:0x0 Action:packet-loss OTHER: NTS-03:52:17.023462
STS-03:52:17.023359
Last N errors.
@Time, Error-Data
------------------------------------------
May 17 03:54:17.123480
        Error description: Id:248 Bit:0x0 Action:packet-loss OTHER: NTS-03:54:17.123566
STS-03:54:17.123480
```

```
May 17 03:54:22.128472
        Error description: Id:250 Bit:0x0 Action:packet-loss OTHER: NTS-03:54:22.128562
STS-03:54:22.128472
        Error description: Id:268 Bit:0x0 Action:packet-loss OTHER: NTS-03:55:07.164378
STS-03:55:07.164275
--------------------------------------------------------------
```

gNMI Output for ASIC Packet Loss Interrupt Errors

Following is the corresponding gNMI output for the `Packet Loss` ASIC errors after subscribing to the packet loss interrupt:


```
{
  "source": "6.4.91.115:57400",
  "subscription-name": "default-1715917870",
  "timestamp": 1715918114950928813,
  "time": "2024-05-16T20:55:14.950928813-07:00",
  "prefix": "openconfig-platform:",
  "updates": [
    {
      "Path":
"components/component[name=0/0/CPU0-NPU0]/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors",

      "values": {

"components/component/integrated-circuit/openconfig-platform-pipeline-counters:pipeline-counters/errors":
 {
          "queueing-block": {
            "queueing-block-error": {
              "name": "slice[0].tx.pdr.general_interrupt_register.mc_flb_to_uc_oq",
              "state": {
                "action": [
                  "LOG"
                ],
                "count": "1",
                "level": "INFORMATIONAL",
                "name": "slice[0].tx.pdr.general_interrupt_register.mc_flb_to_uc_oq",
                "threshold": "0"
              }
            }
          }
        }
      }
    }
  ]
}
```


Trouble Shoot ASIC Errors

Error Category	Specific Error Message	Action
All reset actions	NA	After performing Power On Reset (PON—reload) reset and hard-reset, if the error persists, collect **show tech fabric link-include** statistics and reach out to CISCO TAC.
RMA (Return Merchandise Authorization)	NA	Collect **show tech fabric link-include** statistics and reach out to CISCO TAC.

Error Category	Specific Error Message	Action
Link	NA	Collect **show tech fabric link-include** statistics and reach out to CISCO TAC.
Memory Corruption	SBE	On the 100^{th} error, the board is reloaded and SBE errors are correctable.
	MBE/PARITY	Router gets reloaded on hard-reset for the fifth error. If the error persists, collect **show tech fabric** statistics and reach out to CISCO TAC.
Packet Loss Beyond Threshold	NA	Isolate the board, collect the following statistics, and reach out to CISCO TAC: • **show tech fabric link-include** • **show tech qos** • **show tech ofa**

Stream telemetry for IPv4 and IPv6 data on network interfaces

Streaming telemetry for IPv4 and IPv6 data on network interfaces is a method of continuously collecting and transmitting real-time data using standardized OpenConfig data models and gNMI sensor paths. This approach:

- enables real-time monitoring and reporting of IPv4 and IPv6 operational states and configuration changes, and
- improves network management with standardized data models for seamless multi-vendor compatibility,

Table 10: Feature History Table

Feature Name	Release Information	Description
Stream telemetry for IPv4 and IPv6 data on network interfaces	Release 25.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q200, P100]) You can now enhance network reliability and resource optimization by monitoring IPv4 and IPv6 performance and operational states across platforms. This ensures consistent management, proactive troubleshooting, and optimization in multi-vendor environments using openconfig-if-ip.yang data models and telemetry-enabled sensor paths.

Streaming telemetry data for openconfig data model through gNMI supports monitoring of various data points, include:

- operational status,
- configuration changes, and
- performance metrics.

gNMI sensor paths to stream IPv4 and IPv6 telemetry data

You can stream telemetry data from these gNMI sensor paths using On Change subscription mode or at a cadence of 30 seconds or higher (with a scale of 2000 interfaces). For more details on gNMI subscription, see the GitHub repository.

- openconfig-interfaces/interface/state

IPv4 sub-interface level:

- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv4/state
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv4/addresses
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv4/neighbor
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv4/proxy-arp

IPv6 sub-interface level:

- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv6/state
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv6/addresses
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv6/router-advertisement
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv6/openconfig-if-ip-ext:autoconf
- openconfig-interfaces:interfaces/interface/subinterfaces/subinterface/openconfig-if-ip:ipv6/neighbor

Verify telemetry data for IPv4 and IPv6 on network interfaces

You can verify the telemetry data for IPv4 and IPv6 data on network interfaces.

Procedure

Verify the output of the interface IP statistics.

Example:

This example shows IPv4 state information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.1:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv4/state'
{
    "source": "192.168.2.1:17933",
    "timestamp": 1746203252773061780,
    "time": "2025-05-02T12:27:32.77306178-04:00",
```

```
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv4/state",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv4/state": {
            "counters": {
              "in-multicast-octets": "0",
              "in-multicast-pkts": "0",
              "in-octets": "0",
              "in-pkts": "0",
              "out-multicast-octets": "0",
              "out-multicast-pkts": "0",
              "out-octets": "0",
              "out-pkts": "0"
            },
            "dhcp-client": false,
            "mtu": 1500
          }
        }
      }
    ]
  }
]
```

Example:

This example shows IPv4 address information.

```
auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.2:17933 --username xxxxx --password xxxxxxxx
--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv4/addresses'
[
  {
    "source": "192.168.2.2:17933",
    "timestamp": 1746203286230765971,
    "time": "2025-05-02T12:28:06.230765971-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv4/addresses",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv4/addresses": {
            "address": [
              {
                "config": {
                  "ip": "192.168.10.1",
                  "prefix-length": 24,
                  "type": "PRIMARY"
                },
                "ip": "192.168.20.1",
                "state": {
                  "ip": "192.168.20.1",
                  "origin": "STATIC",
                  "prefix-length": 24,
                  "type": "PRIMARY"
                }
              }
            ]
          }
        }
      }
```

```
    ]
  }
]
```


Example:

This example shows IPv4 neighbor information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.3:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv4/neighbors'
```


```
[
  {
    "source": "192.168.2.3:17933 ",
    "timestamp": 1746203327097683095,
    "time": "2025-05-02T12:28:47.097683095-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv4/neighbors",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv4/neighbors": {
            "neighbor": [
              {
                "ip": "192.168.20.1",
                "state": {
                  "ip": "192.168.20.1",
                  "link-layer-address": "78:bf:38:b6:66:08",
                  "origin": "OTHER"
                }
              },
              {
                "ip": "192.168.20.2",
                "state": {
                  "ip": "192.168.20.2",
                  "link-layer-address": "00:00:00:00:00:00",
                  "origin": "OTHER"
                }
              }
            ]
          }
        }
      }
    ]
  }
]
```


Example:

This example shows IPv4 proxy-arp information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.4:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv4/proxy-arp'
```


```
[
  {
    "source": "192.168.2.4:17933",
    "timestamp": 1746203562540052546,
    "time": "2025-05-02T12:32:42.540052546-04:00",
    "updates": [
      {
        "Path":
```

```
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv4/proxy-arp",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv4/proxy-arp": {
            "config": {
              "mode": "ALL"
            },
            "state": {
              "mode": "ALL"
            }
          }
        }
      }
    ]
  }
]
```


Example:

IPv6 state

This example shows IPv6 state information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.5:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv6/state'
```


```
[
  {
    "source": "192.168.2.5:17933",
    "timestamp": 1746202812260230182,
    "time": "2025-05-02T12:20:12.260230182-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv6/state",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv6/state": {
            "counters": {
              "in-multicast-octets": "0",
              "in-multicast-pkts": "0",
              "in-octets": "0",
              "in-pkts": "0",
              "out-multicast-octets": "0",
              "out-multicast-pkts": "0",
              "out-octets": "0",
              "out-pkts": "0"
            },
            "dup-addr-detect-transmits": 5,
            "enabled": true,
            "mtu": 1500
          }
        }
      }
    ]
  }
]
```


Example:

This example shows IPv6 address information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.6:17933 --username xxxxx --password xxxxxxxx
```

```
--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv6/addresses'
[
  {
    "source": "192.168.2.6:17933",
    "timestamp": 1746202852330541300,
    "time": "2025-05-02T12:20:52.3305413-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv6/addresses",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv6/addresses": {
            "address": [
              {
                "config": {
                  "ip": "10:10:3::1",
                  "prefix-length": 119,
                  "type": "GLOBAL_UNICAST"
                },
                "ip": "10:10:3::1",
                "state": {
                  "ip": "10:10:3::1",
                  "origin": "STATIC",
                  "prefix-length": 119,
                  "status": "PREFERRED",
                  "type": "GLOBAL_UNICAST"
                }
              },
              {
                "ip": "fe80::7abf:38ff:feb6:6608",
                "state": {
                  "ip": "fe80::7abf:38ff:feb6:6608",
                  "origin": "STATIC",
                  "prefix-length": 128,
                  "status": "PREFERRED",
                  "type": "LINK_LOCAL_UNICAST"
                }
              }
            ]
          }
        }
      }
    ]
  }
]
```

Example:

This example shows IPv6 neighbor information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.7:17933 --username xxxxx --password xxxxxxxx
 --
skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv6/neighbor'
[
  {
    "source": "192.168.2.7:17933",
    "timestamp": 1746202898868407604,
    "time": "2025-05-02T12:21:38.868407604-04:00",
    "updates": [
      {
        "Path":
```

```
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]",

         "values": {
           "interfaces/interface/subinterfaces/subinterface": null
         }
       }
     ]
  }
]
```

Example:

This example shows IPv6 state duplicate addres transmits information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.8:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv6/state/dup-addr-detect-transmits'
[
  {
    "source": "192.168.2.8:17933",
    "timestamp": 1746202980834877546,
    "time": "2025-05-02T12:23:00.834877546-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv6/state/dup-addr-detect-transmits",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv6/state/dup-addr-detect-transmits": 5

        }
      }
    ]
  }
]
```

Example:

This example shows IPv6 router advertisement information.

```
/auto/tftpboot-ottawa/b4/bin/gnmic --address 192.168.2.9:17933 --username xxxxx --password xxxxxxxx

--skip-verify --encoding JSON_IETF get --path '/interfaces/interface[name=FourHundredGigE0/0/0/1]/
subinterfaces/subinterface[index=0]/ipv6/router-advertisement/'
[
  {
    "source": "192.168.2.9:17933",
    "timestamp": 1746202744369280518,
    "time": "2025-05-02T12:19:04.369280518-04:00",
    "updates": [
      {
        "Path":
"openconfig:interfaces/interface[name=FourHundredGigE0/0/0/1]/subinterfaces/subinterface[index=0]/ipv6/router-advertisement",

        "values": {
          "interfaces/interface/subinterfaces/subinterface/ipv6/router-advertisement": {
            "config": {
              "enable": true,
              "interval": 4,
              "lifetime": 9000,
              "managed": false,
              "other-config": true,
              "suppress": true
            },
```

```
            "prefixes": {
              "prefix": [
                {
                  "config": {
                    "disable-autoconfiguration": true,
                    "enable-onlink": true,
                    "preferred-lifetime": 5000,
                    "prefix": "300:0:2::/124",
                    "valid-lifetime": 6000
                  },
                  "prefix": "300:0:2::/124",
                  "state": {
                    "disable-autoconfiguration": true,
                    "enable-onlink": true,
                    "preferred-lifetime": 5000,
                    "prefix": "300:0:2::/124",
                    "valid-lifetime": 6000
                  }
                }
              ]
            },
            "state": {
              "enable": true,
              "interval": 4,
              "lifetime": 9000,
              "managed": false,
              "other-config": true,
              "suppress": true
            }
          }
        }
      }
    ]
  }
]
```

Timestamp in nano seconds

From Release 25.2.1, the telemetry messages for all sensor paths are populated with the `timestamp_nano` attribute. This is the time at which the data is collected from the underlying source, or the time that the message is generated, if provided by the underlying source. This is the number of nanoseconds since the Unix Epoch. For reference telemetry messages, see Github.

The primary benefit is the improved timestamp accuracy, which is now in nanoseconds rather than the milliseconds available earlier. Additionally, XR dial-in and dial-out telemetry includes the `timestamp_nano` field in the telemetry messages, ensuring more precise time tracking.