



# Enhancements to Streaming Telemetry

---

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

- [Hardware Timestamp, on page 2](#)
- [Stream QoS Statistics Telemetry Data, on page 4](#)
- [Enhanced Syslog Notifications for Unresolved Line Card Forwarding Paths, on page 9](#)
- [Target-Defined Mode for Cached Generic Counters Data, on page 10](#)
- [FQDN as the gRPC Tunnel Destination, on page 13](#)
- [Stream Telemetry Data about PBR Decapsulation Statistics, on page 16](#)

# Hardware Timestamp

Table 1: Feature History Table

Feature Name	Release Information	Description
Enhancements to Hardware Timestamp	Release 7.3.4	<p>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.</p> <p>The interface counters in the following paths are enhanced for hardware timestamp:</p> <ul style="list-style-type: none"><li>• Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters</li><li>• Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters</li><li>• openconfig-network-instance:network-instances/network-instance/mpls/lsp/constrained-path/tunnels</li><li>• openconfig-interfaces:interfaces/interface</li></ul>
Hardware Timestamp	Release 7.3.1	<p>Whenever periodic statistics are streamed, the collector reads the data from its internal cache, instead of fetching the data from the hardware.</p> <p>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.</p>

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_1[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	<p>You can use the <code>Cisco-IOS-XR-qos-ma-oper.yang</code> 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.</p> <p>In earlier releases, QoS statistics was stored on line cards, and any bundle member or line card failure caused loss of statistics data.</p>

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-13/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-13/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	<p>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.</p> <p>This feature introduces support for the following sensor path:</p> <pre>Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters</pre>

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-sub)#sensor-group-id grpl mode target-defined
Router(config-model-driven-sub)#source-interface Interface1
Router(config-model-driven-sub)#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	<p>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.</p> <p>Model-driven telemetry and Event-driven telemetry are supported.</p> <p>The feature introduces these changes:</p> <p><b>CLI:</b></p> <ul style="list-style-type: none"> <li>• The keywords <b>address-family ipv4</b> and <b>address-family ipv6</b> are introduced in the <b>gRPC tunnel</b> command.</li> </ul>

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 <code>Cisco-IOS-XR-infra-policymgr-oper.yang</code> 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",
"cmmap-stats-arr":[{"cmmap-name":"cmmap1", "matched-bytes":"1713611136", "matched-packets":"13387587",
"transmit-bytes":"1713611136", "transmit-packets":"13387587"}]}]}],
"collection_end_time":"1601361559183"}
----- snipped for brevity -----
```

---