Telemetry Configuration Guide for Cisco 8000 Series Routers, IOS XR Releases

Cisco 8000 Series Routers
Updated September 8, 2026

1 What's changed in this book

Topics:

- Getting Started

Summarizes release updates for the Telemetry Configuration Guide for Cisco 8000 Series Routers.

Use this chapter to track publication updates and the first Cisco IOS XR release covered by this cumulative Telemetry Configuration Guide.

Getting Started

Outlines the release history and update summary for the Telemetry Configuration Guide for Cisco 8000 Series Routers.

This cumulative guide provides a single, continuously updated version of telemetry configuration information for Cisco 8000 Series Routers. It simplifies navigation by letting you use one book for the latest supported Cisco IOS XR releases instead of moving between release-specific versions.

Specific changes or updates tied to individual releases are called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes or the IOS XR Feature Finder.

The table lists the release numbers for which this document has been updated since its initial publication.

Table 1: Changes to this document

Release	Summary
September 2026	Republished for Cisco IOS XR Release 26.3.1
Release 7.3.1	First published for Release 7.3.1

2 YANG data models for telemetry features

Topics:

- YANG data models

Lists the YANG data models that support telemetry features.

This reference lists the YANG data models that support telemetry features.

This reference provides information about the YANG data models for telemetry features.

YANG data models

Explains how YANG data models enable programmatic configuration and operational data collection on network devices, and describes how to access and visualize these models for automated network management.

A YANG data model is a network device configuration and operational data schema that

- enables programmatic configuration and data collection
- provides a standardized structure for automation, and
- supports visualization and exploration for scalable network management

Cisco IOS XR supports programmatic configuration and operational data collection using YANG data models. While CLI-based configurations are easier and more human-readable, model-driven programmability with YANG data models enables automation and scalability.

Accessing and visualizing YANG data models for network automation

Describes how to access and visualize YANG data models for automated network management.

Data models are available in the release image and are also published in the GitHub repository. To view supported data models and their definitions, navigate to the release folder of interest. Each data model defines a complete and cohesive schema or augments an existing model with additional XPaths. For a comprehensive list of supported data models in a release, refer to the Available-Content.md file in the repository.

You can also view data model definitions using the YANG Data Models Navigator tool. This GUI-based tool helps you explore data model structures and view dependencies between containers. You can browse models supported across Cisco IOS XR releases and platforms, locate specific models, and view containers, lists, leaves, and leaf lists in a visual tree structure. This visualization provides insights into nodes that support network automation.

For more information on using data models, refer to the *Programmability Configuration Guide*.

Example of YANG data model usage

Examples include using YANG data models to automate device configuration, collect operational data, and visualize model structures for scalable network management.

3 Scale up your network monitoring strategy using telemetry

Topics:

- Benefits of shifting network monitoring from pull models to telemetry push model
- Telemetry data streaming mechanisms
- Elements that enable streaming telemetry data
- Telemetry session establishment methods

Explains how telemetry enables scalable, automated, and real-time network monitoring compared to traditional polling methods.

Scale up your network monitoring strategy using telemetry is a network feature that addresses the limitations of traditional polling methods by providing real-time, automated data streaming for improved visibility and control.

- Traditional polling methods such as SNMP, Syslog, and CLI may not provide sufficient data for effective network monitoring as network complexity and scale increase.
- These methods use a pull model, where the data collector requests information at regular intervals, which can result in information gaps, resource-intensive operations, and limited automation.
- Telemetry uses a push model to stream data automatically, supporting scale, speed, and automation for network monitoring.

Telemetry versus traditional polling models for network monitoring

Explains the differences between traditional polling methods and telemetry, focusing on scalability, automation, and real-time data collection for network monitoring.

Common questions and challenges in network monitoring include:

- What percentage of the network bandwidth does the network traffic currently consume?
- Do all the links in the network run at a hundred percent utilization rate?
- If an unmanned router fails, is the network operator notified in real time about the issue and its related consequences?
- Is the CPU over- or under-utilized?
- Can the efficiency of the network be calculated based on traffic and data loss?
- What are the possible performance issues that cause traffic loss or network latency?
- How do you proactively prevent issues that may arise? Does the data support the study of network patterns in real time?

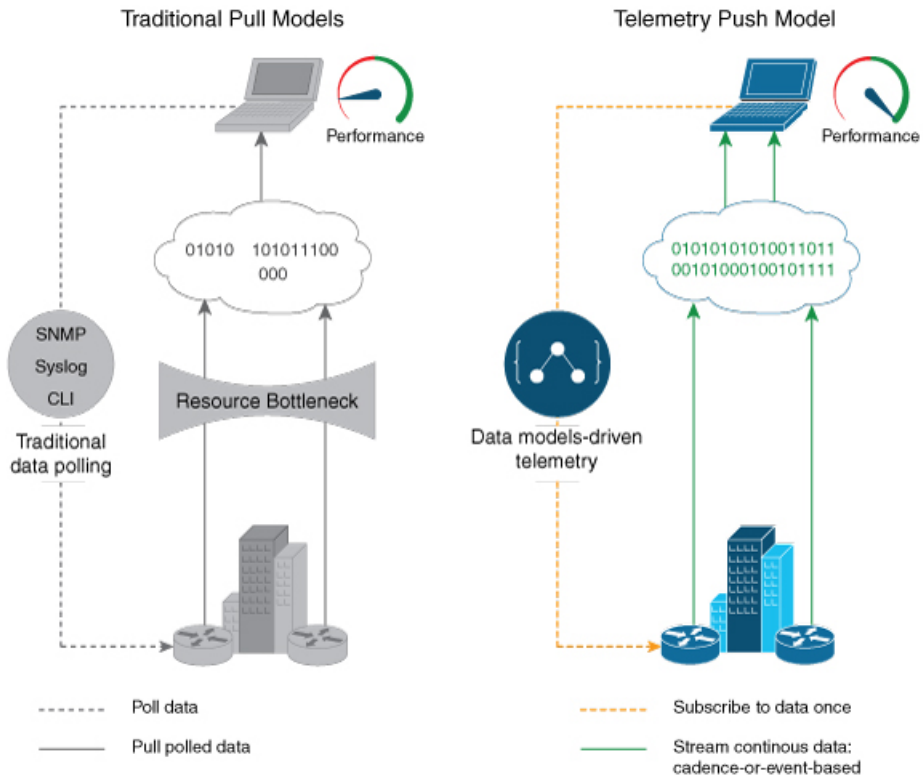
Traditional polling methods require continual manual intervention and limit scalability and automation, inhibiting network visibility and control. As networks grow, a monitoring strategy that adds resiliency and stability is needed.

Telemetry uses a push model that automatically streams data from network devices in real time, eliminating the need for periodic requests from collectors. This model supports scale, speed, and automation, allowing selection of relevant data and transmission in a structured format to remote management stations.

Telemetry provides finer granularity and higher frequency of data, enabling DevOps engineers to quickly locate and investigate issues as they occur and collaborate for improved network control.

The following image compares the benefits of streaming telemetry data using the push model with traditional pull models. Pull models can create resource bottlenecks, while the push model removes these bottlenecks and delivers data efficiently.

Figure 1: Comparison Between Traditional Pull Models and Telemetry Push Model



Watch this video to learn how telemetry data can unlock network intelligence for proactive prediction and troubleshooting.

Note

Starting from Cisco IOS XR, Release 7.0.1, telemetry is part of the base image (<platform>-mini-x.iso). In earlier releases, telemetry was part of the Manageability package (<platform>-mgbl-3.0.0.0-<release>.x86_64.rpm).

This article describes the benefits of using telemetry data and the methods to stream meaningful data from your network device.

Benefits of shifting network monitoring from pull models to telemetry push model

Lists the benefits of using telemetry push models for network monitoring, including real-time data, remote management, traffic optimization, troubleshooting, data visualization, and distributed device monitoring.

This reference outlines the advantages of adopting telemetry push models over traditional pull-based network monitoring approaches.

Telemetry push models provide real-time data that is useful in the following areas:

- Managing network remotely: Telemetry enables you to monitor the state of a network element from a remote location. After deployment, you can analyze, leverage, and act on network insights without being physically present at the site.
- Optimizing traffic: Monitoring link utilization and packet drops at frequent intervals makes it easier to add or remove links, redirect traffic, and modify policing. Technologies like fast reroute allow the network to switch to a new path and reroute faster than traditional SNMP poll intervals. Streaming telemetry data provides quick response times for faster traffic transport.
- Preventive troubleshooting: Network state indicators, statistics, and critical infrastructure information are exposed to the application layer to enhance operational performance and reduce troubleshooting time. The finer granularity and higher frequency of telemetry data enable better performance monitoring and troubleshooting.
- Visualizing data: Telemetry data serves as a data lake for analytics toolchains and applications to visualize valuable insights into network deployments.
- Monitoring and controlling distributed devices: The monitoring function is decoupled from storage and analysis, reducing device dependency and providing flexibility to transform data using pipelines. These pipelines consume telemetry data, transform it, and forward the content to downstream consumers such as Apache Kafka, Influxdata, Prometheus, and Grafana.

Streaming telemetry converts the monitoring process into a big data proposition, enabling rapid extraction and analysis of large data sets to improve decision-making.

Table 2: Telemetry Push Model Benefits Lookup Table

Benefit Area	Description	Key Features	Downstream Consumers
Remote management	Monitor network elements from remote locations and act on insights without physical presence.	Real-time monitoring, remote access	
Traffic optimization	Monitor link utilization and packet drops to optimize traffic and reroute quickly.	Fast reroute, frequent updates	
Preventive troubleshooting	Expose network state and statistics for enhanced performance and reduced troubleshooting time.	High-frequency data, granular metrics	
Data visualization	Serve as a data lake for analytics and visualization tools.	Analytics integration	Grafana, Prometheus

Benefit Area	Description	Key Features	Downstream Consumers
Distributed device monitoring	Decouple monitoring from storage and analysis, enabling flexible data transformation and distribution.	Pipeline processing	Apache Kafka, Influxdata

Telemetry data streaming mechanisms

Explains how routers stream telemetry data using cadence-driven or event-driven mechanisms to support network monitoring and analysis.

Telemetry data streaming mechanisms are network features that enable routers to transmit operational data for monitoring and analysis.

- Routers can stream telemetry data using cadence-driven or event-driven mechanisms.
- Cadence-driven streaming sends data at regular intervals, while event-driven streaming transmits data when specific events occur.
- Platform-specific documentation and monitoring of CPU and NPU utilization are important when configuring collection intervals.

Telemetry streaming types and operational considerations

Routers support two primary telemetry streaming mechanisms: cadence-driven and event-driven. Cadence-driven streaming transmits data at fixed intervals, while event-driven streaming sends data in response to specific network events.

Consider the following when selecting a streaming mechanism:

- Cadence-driven streaming is suitable for continuous monitoring.
- Event-driven streaming is efficient for capturing changes or anomalies.

To configure telemetry streaming, follow these general steps:

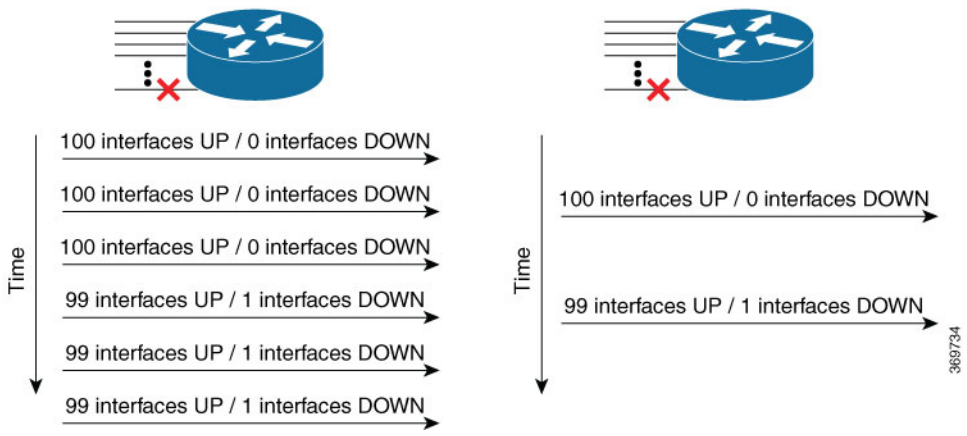
1. Identify the sensor path and determine if it retrieves data from forwarding hardware (such as NPU or ASIC).
2. Set the collection interval based on platform capabilities and operational requirements.

Sensor paths containing `-npu-` in their names are likely to require hardware database retrieval. Naming conventions and data sources can vary by platform and software release.

- Cadence-driven: Sends data at regular intervals.
- Event-driven: Sends data when specific events occur.

Comparison of streaming mechanisms:

Figure 2: Cadence-driven and Event-driven Telemetry



Contrast between streaming mechanisms:

Table 3: Streaming Mechanism Contrast

Cadence-driven	Event-driven
Regular interval transmission	Event-based transmission

Lookup table for sensor path considerations:

Table 4: Sensor Path Lookup Table

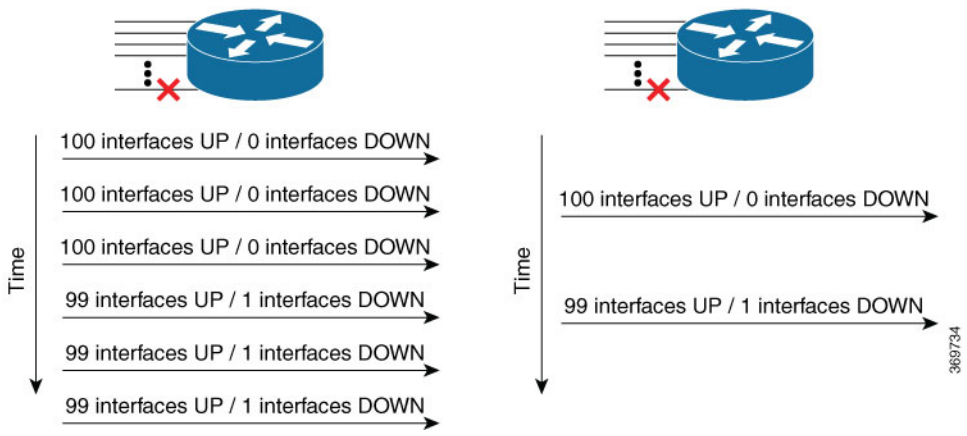
Items	Attribute 1	Attribute 2
Sensor path with -npu-	Requires hardware database retrieval	Monitor CPU/NPU utilization
Other sensor paths	May use software database	Platform-dependent

Note

Short telemetry collection intervals can increase router load, especially when sensor paths retrieve data from forwarding hardware such as NPU or ASIC. Consult platform-specific documentation and monitor CPU and NPU utilization when configuring shorter collection intervals.

Visual representation of telemetry streaming mechanisms:

Figure 3: Cadence-driven and Event-driven Telemetry



Cadence-driven telemetry

Explains the cadence-driven telemetry feature that streams operational statistics and state transitions at a configured cadence to help identify network patterns.

Cadence-driven telemetry is a network feature that streams operational statistics and state transitions at a configured cadence to help identify network patterns.

- Streams data, including operational statistics and state transitions, at a configured cadence.
- Provides a continuous stream of data after a configured time interval.
- Helps users identify emerging patterns in the network.

Cadence-driven telemetry streaming and pattern identification

Cadence-driven telemetry streams data, including operational statistics and state transitions, at a configured cadence. The higher frequency of continuously streamed data helps you identify emerging patterns in the network.

Cadence-driven telemetry provides a continuous stream of data after a configured time interval.

Stem sentence for unordered list

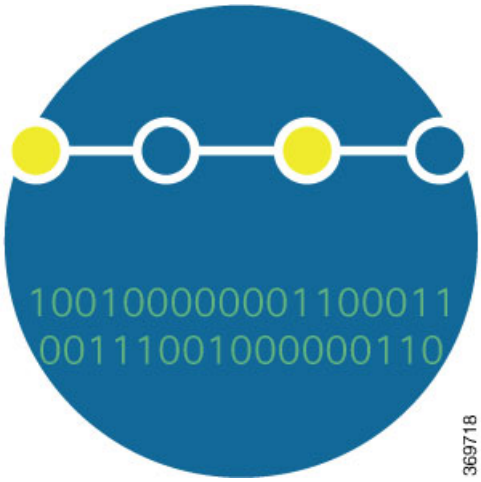
Stem sentence for ordered list.

Stem sentence for item description list.

- Item: Description

Stem sentence for comparison table.

Figure 4: Cadence-driven Telemetry



Stem sentence for contrast table.

Stem sentence for lookup table.

Table 5: Look up Table Title

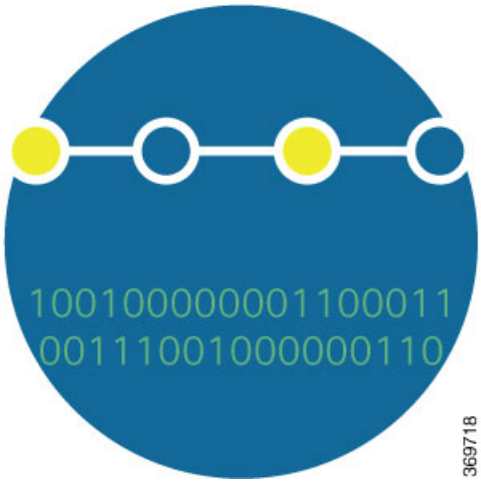
Items	Attribute 1	Attribute 2
Item 1		
Item 2		

Note

TIP: Choose the appropriate note type from the attributes.

Cadence-driven telemetry provides a visual representation of continuous data streaming after a configured time interval.

Figure 5: Cadence-driven Telemetry



Cadence-driven telemetry streaming example

Cadence-driven telemetry streams operational statistics and state transitions at a configured cadence, providing a typical instance of continuous data streaming for network pattern identification.

Cadence-driven telemetry counter-example

Cadence-driven telemetry analogy

Event-driven telemetry

Explains the event-driven telemetry mechanism that streams data only when a state transition occurs, optimizing data collection.

Event-driven telemetry is a network feature that streams data to the receiver only when a state transition occurs.

- optimizes the amount of data collected by sending updates only on state changes
- provides information about interface state transitions and IP route updates, and
- helps users understand when and how to apply event-driven telemetry for efficient monitoring.

Event-driven telemetry operation and use cases

Event-driven telemetry streams data only when a monitored state changes, reducing unnecessary data transmission and improving efficiency.

Common use cases include monitoring interface state transitions and IP route updates.

Event-driven telemetry provides the following benefits:

- Reduces bandwidth usage by sending data only on relevant events.
- Improves responsiveness to network changes.

Event-driven telemetry operation steps:

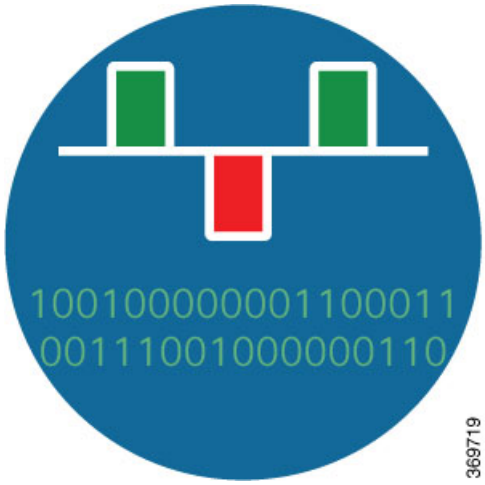
1. Monitor network element for state changes.
2. Stream data to receiver when a state transition occurs.

Event-driven telemetry attributes and descriptions:

- State transition: Triggers data streaming.
- Receiver: Collects streamed data.

Comparison of event-driven and periodic telemetry:

Figure 6: Event-driven Telemetry



Contrast between event-driven and periodic telemetry:

Table 6: Telemetry Mode Contrast

Event-driven telemetry	Periodic telemetry
Streams data only on events	Streams data at set intervals

Lookup table for telemetry modes:

Table 7: Telemetry Modes Lookup

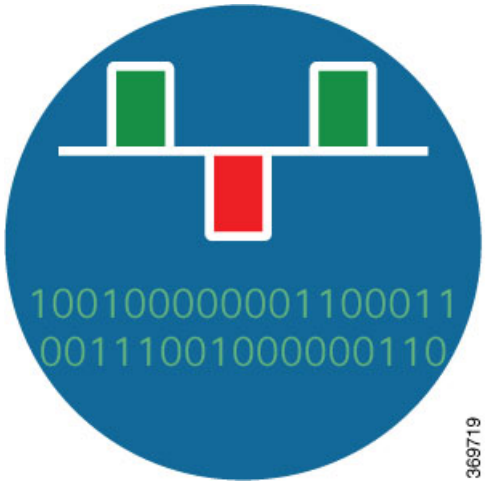
Mode	Trigger	Data Frequency
Event-driven	State change	As needed
Periodic	Timer	Regular intervals

Note

TIP: Use event-driven telemetry to minimize unnecessary data transmission and improve network monitoring efficiency.

The following image shows a stream of data after a state change:

Figure 7: Event-driven Telemetry



Event-driven telemetry example

For example, event-driven telemetry streams data about interface state transitions and IP route updates only when these events occur, reducing the volume of transmitted data.

Elements that enable streaming telemetry data

Lists the essential elements that enable telemetry in a network and outlines their roles in building a telemetry solution.

Objective and scope.

The following elements are essential for enabling telemetry in a network.

- Data sources: Devices or components that generate operational data for telemetry.
- Data collectors: Systems or applications that receive and process telemetry data from network devices.
- Transport protocols: Mechanisms such as gRPC or TCP that carry telemetry data between sources and collectors.
- Data models: Schemas or structures (for example, YANG models) that define the format and semantics of telemetry data.
- Analytics engines: Tools or platforms that analyze, visualize, and act on collected telemetry data.

Table 8: Telemetry elements lookup table

Element	Role	Example	Notes
Data source	Generates telemetry data	Router, switch	Must support telemetry features
Data collector	Receives and stores telemetry data	Telemetry server	May include data processing capabilities
Transport protocol	Transports telemetry data	gRPC, TCP	Protocol selection affects performance
Data model	Defines data structure and semantics	YANG model	Standardizes data interpretation
Analytics engine	Analyzes and visualizes telemetry data	Network analytics platform	Enables actionable insights

Sensor path

Lists the YANG sensor path structure, node types, supported data models, and sensor path examples for telemetry streaming.

A sensor path specifies a YANG path or subset of data definitions in a YANG data model for telemetry streaming.

The sensor path specifies a YANG path or subset of data definitions in a YANG data model within a container. In a YANG model, the sensor path can end at any level in the container hierarchy. A YANG module defines a data model for the router, including the hierarchical organization and constraints on that data. YANG defines four node types. Each node has a name and either defines a value or contains child nodes. The node types for data modeling are:

- Leaf node: contains a single value of a specific type.
- Leaf-list node: contains a sequence of leaf nodes.
- List node: contains a sequence of leaf-list entries, each uniquely identified by one or more key leaves.
- Container node: contains a grouping of related nodes that have only child nodes, which can be any of the four node types.

Additional information:

- Only the following Sysadmin data models are supported for streaming telemetry:
  - Cisco-IOS-XR-sysadmin-controllers-ncs5500
  - Cisco-IOS-XR-sysadmin-entity-mib
  - Cisco-IOS-XR-sysadmin-entity-sensor-mib
  - Cisco-IOS-XR-sysadmin-envmon-ui
  - Cisco-IOS-XR-sysadmin-asic-errors-ael
  - Cisco-IOS-XR-sysadmin-show-media

To get started with using the data models, refer to the *Programmability Configuration Guide*.

Table 9: Sensor Paths

Feature	Sensor Path
CPU	`Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization`
Memory	`Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary`

Interface	`Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`
	`Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/data-rate`
	`openconfig-interfaces:interfaces/interface`

Node summary	`Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary`
Forwarding information base (FIB)	`Cisco-IOS-XR-fib-common-oper:fib-statistics/nodes/node/drops` `Cisco-IOS-XR-fib-common-oper:fib/nodes/node/protocols/protocol/vrfs/vrf/summary`

Feature	Sensor Path

MPLS Traffic engineering (MPLS-TE)	`Cisco-IOS-XR-mpls-te-oper:mpls-te/tunnels/summary`
	`Cisco-IOS-XR-ip-rsvp-oper:rsvp/interface-briefs/interface-brief`
	`Cisco-IOS-XR-mpls-te-oper:mpls-te/fast-reroute/protections/protection`
	`Cisco-IOS-XR-mpls-te-oper:mpls-te/signalling-counters/signalling-summary`
	`Cisco-IOS-XR-mpls-te-oper:mpls-te/p2p-p2mp-tunnel/tunnel-heads/tunnel-head`

MPLS Label distribution protocol (MPLS-LDP)	`Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/nodes/node/bindings-summary-all`
	`Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/global/active/default-vrf/summary`
	`Cisco-IOS-XR-mpls-ldp-oper:mpls-ldp/nodes/node/default-vrf/neighbors/neighbor`

Routing	`Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/statistics-global`
	`Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/neighbors/neighbor`
	`Cisco-IOS-XR-ip-rib-ipv4-oper:rib/rib-table-ids/rib-table-id/summary-protos/summary-proto`
	`Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/levels/level/adjacencies/adjacency`
	`Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/process-info`
	`Cisco-IOS-XR-ip-rib-ipv6-oper:ipv6-rib/rib-table-ids/rib-table-id/summary-protos/summary-proto`

Sensor path configuration and streaming behavior:

- Use specific sensor paths to avoid streaming unnecessary data. For example, to stream only the summary of MPLS-TE, use `sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te/autotunnel/mesh/summary` instead of `sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te`.
- Starting from Cisco IOS XR Software Release 7.2.1, the router streams telemetry data at predefined gather points in the data model even if the sensor-path configuration is to an individual leaf. Gather points are collection units; collection always happens at that level for operational data.
- For cadence-based subscriptions, data can be streamed at the leaf or container level under a gather point. If a subscription has multiple sensor paths that resolve to the same gather point and have the same cadence and encoding, data is pushed in a single collection stream for all the leaves. For example:

```
telemetry model-driven
 sensor-group intf-stats
  sensor-path
Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/bytes-sent

  sensor-path
Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/bytes-received

 !
 subscription intf-stats
  sensor-group-id intf-stats sample-interval 10000
 !
!
end
```

This subscription pushes one message with two leaves because the gather point `full-interface-stats` is the same for both sensor paths `bytes-sent` and `bytes-received`. This grouping of the leaves happens at

the subscription level. If these paths are configured under different subscriptions, data is streamed as different collections with separate messages, each including one leaf (`bytes-sent` or `bytes-received`).

- For event-driven subscriptions, streaming is always at the gather point in the model, even if a specific leaf or leaves are configured as sensor paths. There is configuration to restrict streaming to specific leaves for event-driven subscriptions. If this configuration is used, the sensor path of the configured leaf streams data even if there is a change in one of its adjacent leaves. This means that even if there is no change in value of the configured leaf, data can stream out to the collector. The collector must check if the leaf value changed before taking action on the streamed data.

```
telemetry model-driven
include select-leaves-on-events
```

  It is not recommended to configure sensor paths with the same gather point into different subscriptions.

- In the sensor path configuration, the schema node identifier can be configured with or without a leading slash.
- An MDT-capable device, such as a router, associates the sensor path to the nearest container path in the model. The router encodes and streams the container path within a single telemetry message. A receiver receives data about all the containers and leaf nodes at and below this container path. The router streams telemetry data, for one or more sensor paths, at the configured frequency (Cadence-driven telemetry on page 11), or when an event occurs (Event-driven telemetry on page 13), to one or more collectors through subscribed sessions.

Table 10: Sensor path configuration and streaming behavior

Configuration Aspect	Behavior	Release	Notes
Sensor path specificity	Use specific sensor paths to limit streamed data		Reduces unnecessary data streaming
Gather point streaming	Streaming occurs at gather points in the data model	7.2.1 and later	Applies to operational data
Cadence-based subscriptions	Multiple sensor paths with same gather point and cadence are grouped in one stream		Grouping occurs at subscription level
Event-driven subscriptions	Streaming is always at gather point; can restrict to specific leaves		Collector must check if leaf value changed
Schema node identifier	Can be configured with or without leading slash		

Subscription

Lists the attributes and behavior of telemetry subscriptions, which bind sensor paths and destinations for data collection.

A subscription binds one or more sensor paths and destinations for telemetry data collection. The collector uses the subscription to receive updates about the state of data on the router. Data for the subscribed paths streams until the session is terminated by the collector or the telemetry subscription configuration is removed.

A subscription binds sensor paths and destinations for telemetry data collection.

- A subscription can consist of one or more sensor paths.

- Data streams for the subscribed paths until the session is terminated or the subscription is removed.
- The collector uses the subscription to receive updates about the state of data on the router.

Table 11: Subscription attributes and behavior

Item	Attribute	Description	Notes
Subscription	sensor paths	One or more sensor paths can be associated with a subscription.	Each path defines a data source for telemetry.
Subscription	destination group	Specifies where telemetry data is sent.	Multiple destinations can be configured.
Subscription	sample interval	Defines how often data is collected and sent.	Configured in milliseconds.
Subscription	strict-timer	Enforces deterministic data collection at the configured interval.	On 32-bit platforms, configure only under the subscription; on 64-bit platforms, can be global or per subscription.
Subscription	session termination	Data streaming stops when the collector terminates the session or the subscription is removed.	Ensures control over telemetry data flow.

Table 12: Subscription and strict-timer configuration comparison

Attributes	Subscription-level strict-timer	Global strict-timer
Platform support	32-bit and 64-bit	64-bit only
Configuration scope	Per subscription	All subscriptions
Effect	Deterministic interval for one subscription	Deterministic interval for all subscriptions
Configuration mode	Under subscription configuration	Under telemetry model-driven configuration

Note

With a strict-timer configured for the sample interval, data collection starts exactly at the configured time interval, allowing deterministic behavior to stream data. On 32-bit platforms, strict-timer can be configured only under the subscription. 64-bit platforms support configuration at the global level in addition to the subscription level. Configuring at the global level affects all configured subscriptions.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#strict-timer
```

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription SUB1
Router(config-model-driven-subs)#sensor-group-id SGROUP1 sample-interval 10000
Router(config-model-driven-subs)#strict-timer
```

Subscriptions are configured to control telemetry data collection and delivery.

- Associate sensor groups, sample intervals, and destination groups with a subscription.
- Configure strict-timer for deterministic data streaming intervals.

Table 13: Subscription configuration example

Item	Command	Description	Notes
Configure telemetry model-driven	`telemetry model-driven`	Enters telemetry model-driven configuration mode.	
Create subscription	`subscription SUB1`	Creates a subscription named SUB1.	
Associate sensor group and sample interval	`sensor-group-id SGROUP1 sample-interval 10000`	Associates sensor group SGROUP1 and sets sample interval to 10000 ms.	
Configure strict-timer	`strict-timer`	Enables strict-timer for deterministic data collection intervals.	On 32-bit platforms, configure only under the subscription; on 64-bit platforms, can be global or per subscription.
Global strict-timer configuration	`strict-timer` (global)	Configures strict-timer globally for all subscriptions (supported on 64-bit platforms).	

Encoder

Lists the available encoder formats used to stream data from a router and their characteristics.

Encoders define the format used to stream data from a router. This reference outlines the supported encoding formats and their attributes.

You can encode streamed data from a router using one of these formats:

- GPB encoding: To use GPB encoding, you must provide metadata as compiled `.proto` files. A `.proto` file describes the GPB message format for streaming data. The `.proto` files are available at Cisco Network Telemetry Proto in Github.
  - Compact GPB encoding: Data is streamed in a compressed, non-self-descriptive format. The collector must use a `.proto` file for each sensor path to decode the data.
  - Self-describing GPB encoding: Data for each sensor path is streamed in a self-describing ASCII text format. The collector uses a single `telemetry.proto` file to decode any sensor path data. Self-describing GPB encoding is easier to manage because it requires only one `.proto` file, but the message size is larger.
- JSON encoding: Data is streamed as human-readable strings of keys and values.

Table 14: Encoder format lookup table

Item	Attribute 1	Attribute 2	Attribute 3
GPB encoding	Requires .proto files	Supports compact and self-describing formats	Used for streaming telemetry data
Compact GPB encoding	Compressed, non-self-descriptive	Collector needs .proto file per sensor path	Smaller message size
Self-describing GPB encoding	ASCII text, self-descriptive	Collector uses single telemetry.proto file	Larger message size, easier management
JSON encoding	Human-readable keys and values	No .proto files required	Easy to interpret

Table 15: Encoder format comparison table

Attributes	GPB encoding	JSON encoding
File requirement	.proto files required	No .proto files required
Format type	Binary or ASCII (self-describing)	Human-readable text
Collector integration	Requires decoding logic	Easy parsing
Message size	Compact or larger (self-describing)	Typically larger

> **Note**
>
> GPB encoding offers flexibility for telemetry streaming, while JSON encoding provides ease of use and readability.

Encoder formats serve to define how streamed data is structured and interpreted by collectors.

- GPB encoding provides flexibility for compact and self-describing telemetry data.
- JSON encoding offers human-readable output for easier integration and analysis.

Table 16: Encoder function lookup table

Item	Attribute 1	Attribute 2	Attribute 3
GPB encoding	Flexible telemetry streaming	Supports multiple sensor paths	Collector decodes with .proto files
Compact GPB encoding	Efficient bandwidth usage	Requires per-path decoding	Optimized for performance
Self-describing GPB encoding	Single file management	Easy collector integration	Higher message size
JSON encoding	Readable output	Simple parsing	Broad compatibility

Transport protocols for telemetry streaming

Lists the supported transport protocols and encoders used by routers to stream telemetry data in the telemetry push model.

This reference outlines the supported transport protocols and encoders for streaming telemetry data using the telemetry push model on routers.

In the telemetry push model, the router streams telemetry data using a transport protocol. The generated data is encapsulated into the desired format using encoders. Model-Driven Telemetry (MDT) data is streamed through these supported transport protocols:

- Google Protocol RPC (gRPC): used for both dial-in and dial-out modes.

  > **Note**
  >
  > gRPC protocol is not supported over Multiprotocol Label Switching (MPLS) including `explicit-null` label. Starting Cisco IOS XR Software Release `24.1.1`, this limitation is not applicable.

- Transmission Control Protocol (TCP): used for only dial-out mode.
- User Datagram Protocol (UDP): used for only dial-out mode. Because UDP is connectionless, the UDP destination is shown as `Active` irrespective of the state of the collector.

  UDP for Telemetry is not recommended for production networks as it doesn't support models that send messages larger than the UDP size limit of `65507` bytes.

If a message is dropped by the network before it reaches the collector, the protocol does not resend the data.

- Loopback1 is reserved and not supported for gRPC usage on IOS XR. You may use Loopback0 or Loopback2 and higher for gRPC and related configurations. However, from Cisco IOS XR Software Release 24.4.1, you can change the east-west interface using **linux networking vrf** *vrf name* **east-west** *interface name.*

> **Note**
>
> Telemetry data is streamed out of the router using an Extensible Manageability Services Deamon (emsd) process. The data of interest is subscribed through subscriptions and streamed through gRPC, TCP, or UDP sessions. However, a combination of gRPC, TCP, and UDP sessions with more than 150 active sessions leads to emsd crash or process restart.

Table 17: Supported transport protocols for telemetry streaming

Protocol	Mode	Notes	Platform/Release
gRPC	Dial-in, Dial-out	Not supported over MPLS (including `explicit-null` label) prior to IOS XR 24.1.1; not supported over subinterfaces on ASR9K prior to IOS XR 24.1.1; not supported on Loopback1.	All platforms; subinterface support on ASR9K from IOS XR 24.1.1
TCP	Dial-out	Standard support for dial-out mode.	All platforms
UDP	Dial-out	Connectionless; destination always shown as `Active`; not recommended for production; does not support messages >65507 bytes; no retransmission if dropped.	All platforms
Loopback interfaces	n/a	Loopback1 not supported for gRPC; use Loopback0 or Loopback2+; from IOS XR 24.4.1, east-west interface can be changed using linux networking vrf *vrf name* east-west *interface name*.	All platforms; interface change from IOS XR 24.4.1
emsd process	n/a	More than 150 active gRPC/TCP/UDP sessions can cause emsd crash or restart.	All platforms

gRPC Network Management Interface

Explains the gRPC Network Management Interface (gNMI) protocol that manages device configuration and streams telemetry data using gRPC.

gRPC Network Management Interface (gNMI) is a protocol that manages device configuration and streams telemetry data using gRPC.

- gNMI is a gRPC-based network management protocol used to modify, install, or delete configuration from network devices.
- It is used to view operational data, control, and generate telemetry streams from a target device to a data collection system.
- gNMI uses a single protocol to manage configurations and stream telemetry data from network devices.

gNMI subscription modes and configuration

Describes the modes and configuration options for gNMI streaming subscriptions, including cadence and event-driven settings.

gNMI defines three modes for streaming subscriptions that indicate how the router returns data:

- The `SAMPLE` mode is a cadence-based subscription supported for all operational models.
- The `ON_CHANGE` mode is an event-based subscription. Only the state leaf supports on_change events.
- The `TARGET_DEFINED` mode allows the target to determine the best type of subscription to be created on a per-leaf basis.

When a client creates a subscription specifying `TARGET_DEFINED`, the router determines the best subscription type per leaf. If the path refers to event-driven leaves, an `ON_CHANGE` subscription is created.

In Cisco IOS XR Release 7.2.1, `TARGET_DEFINED` mode is supported only for sensor paths of the OpenConfig model. Supported models include OC Interfaces, OC Telemetry, OC Shell Util, OC System NTP, and OC Platform.

An initial synchronization is established with all leaves. If multiple clients request the same subscription information, the initial synchronization is resent to all existing connections.

The **telemetry model-driven gnmi-target-defined** command determines the cadence for leaves using these parameters:

- cadence-factor: Multiplier factor for cadence of target-defined subscriptions. Range: 1 to 10. Default: 2.
- minimum-cadence: Minimum cadence for target-defined subscriptions in seconds. Range: 1 to 65535. Default: 30 seconds.

If cadence is specified in the gNMI request, it is used for the first collection. The cadence is calculated as:

```
Cadence = Maximum (Global minimum-cadence defined, (Collection time for one collection * cadence-factor))
```

If cadence is not specified, the default value of 30 seconds is used. This value can be modified using the following commands:

1. telemetry model-driven
2. gnmi-target-defined minimum-cadence 90
3. gnmi-target-defined cadence-factor 6

For more information about gNMI, refer to Github.

> **Note**
>
> Starting from Cisco IOS XR Software Release 24.4.1, at the start of the collection, target-defined subscriptions adjust their sample cadence interval (initially set as 30 seconds) based on the first few iterations of collection. The sample interval is adjusted to `first collection time * target-defined cadence-factor`.

Stem sentence for unordered list

- gNMI supports multiple subscription modes for telemetry data collection.

Stem sentence for ordered list.

1. Configure cadence and minimum-cadence parameters for target-defined subscriptions.

Stem sentence for item description list.

- cadence-factor: Multiplier for cadence interval.
- minimum-cadence: Minimum interval for data collection.

Stem sentence for comparison table.

Table 18: Comparison Table Title

Attributes	SAMPLE	ON_CHANGE
Subscription type	Cadence-based	Event-based
Supported models	All operational models	State leaf only

Stem sentence for contrast table.

Table 19: Contrast Table Title

TARGET_DEFINED	Native Model
Supported	Not supported

Stem sentence for lookup table.

Table 20: Look up Table Title

Items	Attribute 1	Attribute 2
SAMPLE	Cadence-based	All operational models
ON_CHANGE	Event-based	State leaf only

> **Note**
>
> TIP: Choose the appropriate note type from the attributes.

gNMI configuration example

Examples can be text only, graphics only, or a combination of text and graphics. Provide examples that show a typical instance of the concept and display all the key attributes.

```
telemetry model-driven
gnmi-target-defined minimum-cadence 90
gnmi-target-defined cadence-factor 6
!
!
```

Unsupported gNMI subscription mode

Native model is not supported for TARGET_DEFINED subscription mode in Cisco IOS XR Release 7.2.1.

gNMI subscription analogy

gNMI subscription modes are similar to choosing between scheduled updates, event notifications, or letting the system decide the best method for each item.

gRPC Network Operations Interface

Explains the gRPC Network Operations Interface (gNOI), which provides gRPC-based microservices for executing operational commands on network devices.

The gRPC Network Operations Interface (gNOI) is a network feature that enables gRPC-based microservices for operational command execution on network devices.

- gNOI uses gRPC as the transport protocol, and its configuration is the same as that of gRPC.
- Identifies relevant configuration, support, or operational details.
- Helps users understand when and how to apply the feature.

gNOI microservices and implementation details

The gNOI defines a set of gRPC-based microservices for executing operational commands on network devices. Extensible Manageability Services (EMS) gNOI is the Cisco IOS XR implementation of gNOI. gNOI uses gRPC as the transport protocol, and its configuration is the same as that of gRPC.

Refer to the for the list of gNOI RPCs.

Multiple gNOI microservices are available for operational command execution:

- Device management
- Certificate management
- File operations

Operational commands can be executed in sequence:

1. Initiate gNOI session
2. Send operational command
3. Receive response

Each gNOI microservice provides specific functionality:

- Device: Enables device-level operations
- Certificate: Manages device certificates
- File: Handles file operations on the device

Comparison of gNOI and gRPC attributes:

Table 21: gNOI vs gRPC Comparison

Attributes	gNOI	gRPC
Transport protocol	gRPC	gRPC
Microservice support	Yes	No

Contrast between gNOI and other operational interfaces:

Table 22: gNOI and Other Operational Interfaces

gNOI	Other Interfaces
Microservice-based	Not microservice-based

Lookup table for gNOI microservices:

Table 23: gNOI Microservices Lookup

Microservice	Function	Supported Platform
Device	Device-level operations	Cisco IOS XR
Certificate	Certificate management	Cisco IOS XR

Note

TIP: Ensure gNOI configuration matches gRPC transport settings for proper operation.

gNOI operational command example

For example, a user can use gNOI to initiate a device reboot by sending a command through the Device microservice, which executes the operation on the network device.

Non-gNOI operational command example

Using legacy CLI commands for device operations does not leverage gNOI microservices and lacks the benefits of gRPC-based transport.

gNOI analogy

gNOI microservices are like specialized tools in a toolbox, each designed for a specific operational task on network devices.

TLS authentication

Explains how TLS encrypts data and authenticates connections for gRPC model-driven telemetry.

TLS authentication is a network feature that secures gRPC model-driven telemetry by encrypting data and validating connections.

- The gRPC protocol supports Transport Layer Security (TLS) for encrypting data.
- By default, model-driven telemetry uses TLS to dial out.
- When TLS is enabled, the server sends a certificate to authenticate with the collector.

TLS authentication for gRPC model-driven telemetry

TLS encrypts data and authenticates connections between servers and collectors in gRPC model-driven telemetry. The collector validates the certificate by verifying the certificate authority and generates session keys to encrypt the session.

To bypass TLS, use the **grpc no-tls** command.

Unordered list: Key actions for TLS authentication in dial-out and dial-in scenarios.

- Verify that the gRPC protocol is configured for dial-out mode, as it supports TLS by default.
- Copy the TLS certificate to the `/misc/config/grpc/dialout/` path on the router. This certificate authenticates the server with the collector.
- In the certificate, set the `Common Name` (CN) to match the command, which defaults to the IP address of the destination if not specified.

```
protocol grpc tls-hostname
```

- Disable TLS 1.0: You can disable TLS version 1.0 using the **tlsv1-disable** command.

```
grpc
tlsv1-disable
```

- Use the certificate generated by EMS located in `/misc/config/grpc/[ems.pem|ca.cert|ems.key]`. To use your own certificates, replace them with the same names in the grpc directory and restart the process.

Ordered list: Sequence for TLS certificate setup in dial-out scenario.

1. Configure gRPC protocol for dial-out mode.
2. Copy the TLS certificate to the router's dialout directory.
3. Set the Common Name in the certificate to match the destination command or IP address.
4. Disable TLS 1.0 if required.

Item description list: TLS certificate files and their purpose.

- ems.pem: EMS-generated certificate for authentication.
- ca.cert: Certificate authority file for validation.
- ems.key: Private key for secure communication.

Comparison table: TLS authentication attributes for dial-out and dial-in scenarios.

Table 24: TLS Authentication Comparison

Attributes	Dial-out	Dial-in
Certificate location	/misc/config/grpc/dialout/	/misc/config/grpc/[ems.pem|ca.cert|ems.key]
Default Common Name	IP address of destination	EMS-generated or user-defined

Contrast table: TLS version support and security.

Table 25: TLS Version Support

TLS 1.0 enabled	TLS 1.0 disabled
Potential security threat	Improved security

Stem sentence for lookup table.

Table 26: Look up Table Title

Items	Attribute 1	Attribute 2
Dialout.pem	Certificate file for dial-out	Located in /misc/config/grpc/dialout/
ems.pem	EMS-generated certificate	Located in /misc/config/grpc/

Note

TLS provides secure communication between servers and clients, but TLS version 1.0 may pose a security threat. You can disable TLS version 1.0 using the **grpc tlsv1-disable** command.

This output shows the certificate that gRPC uses to establish a dial-out session:

```
Router#run
[node:]$ls -l /misc/config/grpc/dialout/
total 4
-rw-r--r-- 1 root root 4017 dialout.pem
```

Telemetry session establishment methods

Lists the available methods for establishing a telemetry session, including dial-out and dial-in modes.

This reference outlines the methods for establishing a telemetry session and highlights their similarities.

You can initiate a telemetry session using dial-out mode or dial-in mode. Both modes use the same data model and stream the same data.

- Dial-out mode: The device initiates the connection to the collector.
- Dial-in mode: The collector initiates the connection to the device.

Table 27: Telemetry session establishment methods comparison

Method	Initiator	Data Model	Data Streamed
Dial-out	Device	Same	Same
Dial-in	Collector	Same	Same

Dial-out mode

Explains dial-out mode, where the router initiates a subscription-based telemetry session by dialing out to the receiver.

Dial-out mode is a configuration mode that enables the router to initiate a subscription-based telemetry session by connecting to the receiver.

- Because the router starts the connection, you do not need to manage ports for inbound traffic.
- A simple protocol requires only access to the socket on the collector, while a secure protocol provides authentication and encryption for the session.
- This allows you to secure your collector and use a more advanced communication method with the router.

Dial-out mode operation and protocol options

Dial-out mode enables the router to initiate a connection to the receiver, establishing a subscription-based telemetry session. The router starts the connection, eliminating the need to manage ports for inbound traffic. This default mode lets you choose between protocols that offer simplicity, such as TCP, or security, such as gRPC. Simple protocols require only access to the socket on the collector, while secure protocols provide authentication and encryption for the session. This secures your collector and enables advanced communication with the router. If the connection between the router and the destination is lost, the router re-establishes the connection and continues to push data. Data transmitted during the reconnection period is lost.

For information about creating a dial-out session, refer to Establish a Model-Driven Telemetry Session from a Router to a Collector on page 34.

Dial-in mode

Explains dial-in mode, where a collector initiates a connection to the router and dynamically subscribes to sensor paths for secure data collection.

Dial-in mode is a configuration mode that enables a collector to initiate a connection to the router and dynamically subscribe to sensor paths for data collection.

- Because the collector establishes the session, destinations do not need to be created in the router configuration.
- The protocol used to establish the session, gRPC, provides advanced security capabilities to authenticate and encrypt the session.
- Only gRPC supports dial-in sessions.

Dial-in mode operation and session behavior

Dial-in mode allows a collector to initiate a connection to the router and subscribe dynamically to one or more sensor paths specified in a subscription. The router accepts connections from the collector, enabling a single channel of communication. Because the collector establishes the session, destinations do not need to be created in the router configuration. The protocol used to establish the session, gRPC, provides advanced security capabilities to authenticate

and encrypt the session. If the connection between the router and the collector is lost, the session is cancelled, and the collector must reconnect to the router to restart data streaming. Only gRPC supports dial-in sessions.

To explore dial-in mode and create a dial-in session, refer to Model-driven telemetry sessions on page 46.

Dial-in mode offers several operational characteristics:

- The collector initiates the session, eliminating the need for destination configuration on the router.
- gRPC protocol ensures secure authentication and encryption for the session.
- Session cancellation occurs if the connection is lost, requiring the collector to reconnect for data streaming.

Dial-in mode session steps:

1. The collector initiates a connection to the router.
2. The router accepts the connection and enables communication.
3. The collector subscribes to sensor paths for data collection.

Dial-in mode attributes and descriptions:

- Collector-initiated session: The collector starts the connection.
- Dynamic subscription: Sensor paths are subscribed to during the session.
- gRPC protocol: Provides authentication and encryption.

Comparison of dial-in mode and other modes:

Table 28: Dial-in mode vs. other modes

Attributes	Dial-in mode	Other modes
Session initiation	Collector initiates	Router initiates
Protocol	gRPC	Other protocols

Contrast between dial-in mode and router-initiated mode:

Table 29: Dial-in vs. router-initiated mode

Dial-in mode	Router-initiated mode
Collector initiates session	Router initiates session

Lookup table for dial-in mode attributes:

Table 30: Dial-in mode attribute lookup

Items	Attribute 1	Attribute 2
Dial-in session	Collector-initiated	gRPC protocol
Router-initiated session	Router-initiated	Other protocols

Note

TIP: Only gRPC supports dial-in sessions, and session security is ensured through authentication and encryption.

Dial-in mode session visual representation:

Dial-in mode example

For example, a collector initiates a gRPC connection to a router, subscribes to sensor paths, and receives streaming telemetry data without configuring destinations on the router.

Dial-in mode counter-example

In router-initiated mode, the router establishes the session and requires destination configuration, which differs from dial-in mode.

Dial-in mode analogy

Dial-in mode is like a guest calling a host to request information, rather than the host reaching out to the guest.

Identify the telemetry session suitable for your network

Lists the available telemetry session modes and their suitability based on transport protocols, encoding formats, and data streaming requirements.

Identify the appropriate telemetry session mode for your network based on transport protocols, encoding formats, and data streaming needs.

The transport protocols and encoding formats in your network help determine which telemetry session mode is suitable for your needs. Encoding efficiency depends on the space data occupies on the wire, memory utilization, and the amount of data you plan to stream from the router.

- Use TCP dial-out mode to stream telemetry data with a simple setup involving a single router and collector. This mode is easy to configure and does not require extensive protocol knowledge. It eliminates the need to manage ports for inbound connections.
- Use gRPC dial-out mode if your setup requires scaling to many devices or encrypting your data. This mode also removes the need to manage ports for inbound connections.
- Use gRPC dial-in mode if you already use gRPC in your network and want dynamic sessions without streaming data to fixed destinations. This mode is convenient for centralized configuration and operational data requests.

Table 31: Telemetry session modes and attributes

Session Mode	Transport Protocol	Encoding Format	Use Case
TCP dial-out	TCP	GPB/JSON	Simple setup, single router and collector, easy configuration, no inbound port management
gRPC dial-out	gRPC	GPB/JSON	Scalable to many devices, supports encryption, no inbound port management

Session Mode	Transport Protocol	Encoding Format	Use Case
gRPC dial-in	gRPC	GPB/JSON	Dynamic sessions, centralized configuration, operational data requests, no fixed streaming destinations

4 Establish a Model-Driven Telemetry Session from a Router to a Collector

Topics:

- Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure

Establish a Model-Driven Telemetry Session from a Router to a Collector explains the key information for this topic.

Streaming telemetry is a new paradigm in monitoring the health of a network. It provides a mechanism to efficiently stream configuration and operational data of interest from Cisco IOS XR routers. This streamed data is transmitted in a structured format to remote management stations for monitoring and troubleshooting purposes.

With telemetry data, you create a data lake. Analyzing this data, you proactively monitor your network, monitor utilization of CPU and memory, identify patterns, troubleshoot your network in a predictive manner, and devise strategies to create a resilient network using automation.

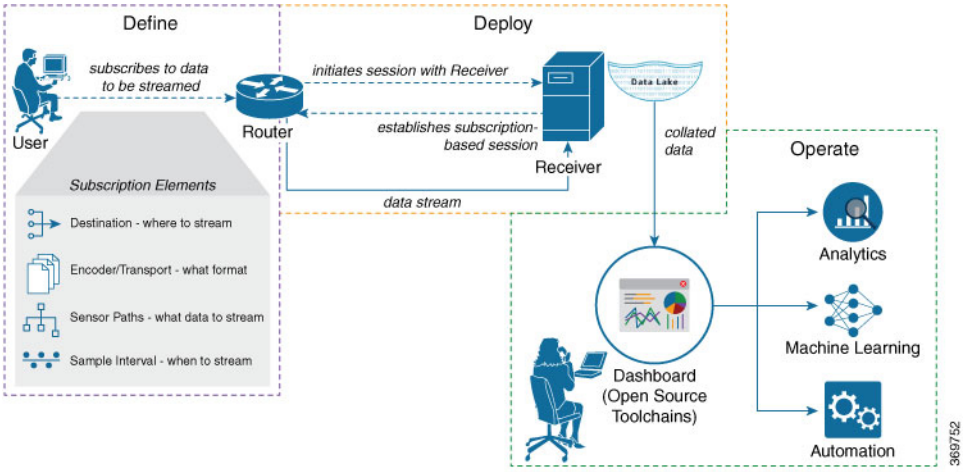
Telemetry works on a subscription model where you subscribe to the data of interest in the form of sensor paths. The sensor paths describes OpenConfig data models or native Cisco data models. You can access the OpenConfig and Native data models for telemetry from Github, a software development platform that provides hosting services for version control. You choose who initiates the subscription by establishing a telemetry session between the router and the receiver. The session is established using either a dial-out mode or a dial-in mode, described in the Scale-Up Your Network Monitoring Strategy Using Telemetry article.

> **Note**
>
> Watch this video to discover the power of real-time network management using model-driven telemetry.

This article describes the dial-out mode where the router dials out to the receiver to establish a telemetry session. In this mode, destinations and sensor-paths are configured and bound together into one or more subscriptions. The router continually attempts to establish a session with each destination in the subscription, and streams data to the receiver. The dial-out mode of subscriptions is persistent. Even when a session terminates, the router continually attempts to re-establish a new session with the receiver at regular intervals.

The following image shows a high-level overview of the dial-out mode:

Figure 8: Dial-Out Mode



This article describes, with a use case that illustrates the monitoring of CPU utilization, how streaming telemetry data helps you gain better visibility of your network, and make informed decisions to stabilize your network.

Note

You can programmatically configure a dial-out telemetry session using `openconfig-telemetry.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide*.

Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure

Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure explains the key information for this topic.

The use case illustrates how, with the dial-out mode, you can use telemetry data to proactively monitor CPU utilization. Monitoring CPU utilization ensures efficient storage capabilities in your network. This use case describes the tools used in the open-sourced collection stack to store and analyse telemetry data.

> **Note**
>
> Watch this video to see how you configure model-driven telemetry to take advantage of data models, open source collectors, encodings and integrate into monitoring tools.

Telemetry involves the following workflow:

- **Define:** You define a subscription to stream data from the router to the receiver. To define a subscription, you create a destination-group and a sensor-group.
- **Deploy:** The router establishes a subscription-based telemetry session and streams data to the receiver. You verify subscription deployment on the router.
- **Operate:** You consume and analyse telemetry data using open-source tools, and take necessary actions based on the analysis.

Before you begin

Make sure you have L3 connectivity between the router and the receiver.

Define a Subscription to Stream Data from Router to Receiver

Create a subscription to define the data of interest to be streamed from the router to the destination.

Procedure

1. Create one or more destinations to collect telemetry data from a router. Define a destination-group to contain the details about the destinations. Include the destination address (IPv4 or ipv6), port, transport, and encoding format in the destination-group.

   #### Example

   Create a destination-group using data model

   This example uses the native data model `Cisco-IOS-XR-um-telemetry-model-driven-cfg.yang`.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-config>
    <source>
      <candidate/>
    </source>
    <filter>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
```

```
        <destination-groups>
          <destination-group>
            <destination-id>CPU-Health</destination-id>
            <ipv4-destinations>
              <ipv4-destination>
                <ipv4-address>172.0.0.0</ipv4-address>
                <destination-port>57500</destination-port>
                <encoding>self-describing-gpb</encoding>
                <protocol>
                  <protocol>tcp</protocol>
                </protocol>
              </ipv4-destination>
            </ipv4-destinations>
          </destination-group>
        </destination-groups>
      </telemetry-model-driven>
    </filter>
  </get-config>
</rpc>
```

Create a destination group using CLI

```
##Configuration with tls-hostname##
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group CPU-Health
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 57500
Router(config-model-driven-dest-addr)#encoding self-describing-gpb
Router(config-model-driven-dest-addr)#protocol tcp
Router(config-model-driven-dest-addr)#commit
```

- `CPU-Health` is the name of the destination-group
- `172.0.0.0` is the IP address of the destination where data is to be streamed

> **Note**
>
> To avoid hard-coding IP address, the router can chose any of the configured IPv4 or ipv6 address using domain name service. If an established connections fails, the router connects to another resolved IP address, and streams data to that IP address.

- `57500` is the port number of the destination
- `self-describing-gpb` is the format in which data is encoded and streamed to the destination
- `tcp` is the protocol through which data is transported to the destination.

```
domain ipv4 host abcd 172.x.x.1 172.x.x.2
```

```
domain ipv6 host abcd fd00:xx:xx:xx:1::1 fd00:xx:xx:xx:1::3
```

2. Specify the subset of the data that you want to stream from the router using sensor paths. The sensor path represents the path in the hierarchy of a YANG data model. Create a sensor-group to contain the sensor paths.

Example

Create a sensor-group for CPU utilization using data model

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">

        <sensor-groups>
          <sensor-group>
            <sensor-group-identifier>Monitor-CPU</sensor-group-identifier>
            <sensor-paths>
              <sensor-path>

<telemetry-sensor-path>Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization</telemetry-sensor-path>

              </sensor-path>
            </sensor-paths>
          </sensor-group>
        </sensor-groups>
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

Create a sensor-group for CPU utilization using CLI

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group Monitor-CPU
Router(config-model-driven-snsr-grp)# sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
Router(config-model-driven-snsr-grp)# commit
```

- `Monitor-CPU` is the name of the sensor-group
- `Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization` is the sensor path from where data is streamed.

3. Subscribe to telemetry data that is streamed from a router. A subscription binds the destination-group with the sensor-group and sets the streaming method. The streaming method can be cadence-driven or event-driven telemetry.

Example

> **Note**
>
> The configuration for event-driven telemetry is similar to cadence-driven telemetry, with only the sample interval as the differentiator. Configuring the sample interval value to `0`, zero, sets the subscription for event-driven telemetry, while configuring the interval to any non-zero value sets the subscription for cadence-driven telemetry.

Create a subscription using data model

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">

        <subscriptions>
          <subscription>
           <subscription-identifier>CPU-Utilization</subscription-identifier>

            <sensor-profiles>
              <sensor-profile>
                <sensorgroupid>Monitor-CPU</sensorgroupid>
                <sample-interval>30000</sample-interval>
              </sensor-profile>
            </sensor-profiles>
            <destination-profiles>
              <destination-profile>
                <destination-id>CPU-Health</destination-id>
              </destination-profile>
            </destination-profiles>
           <source-interface>Interface1</source-interface>
          </subscription>
        </subscriptions>
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

Create a subscription using CLI

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription CPU-Utilization
Router(config-model-driven-subs)#sensor-group-id Monitor-CPU sample-interval
 30000
Router(config-model-driven-subs)#destination-id CPU-Health
Router(config-model-driven-subs)#source-interface Interface1
Router(config-model-driven-subs)#commit
```

where -

- `CPU-Utilization` is the name of the subscription
- `Monitor-CPU` is the name of the sensor-group

- `CPU-Health` is the name of the destination-group
- `Interface1` is the source interface that is used for establishing the telemetry session. If both the VRF and source interface are configured, the source interface must be in the same VRF as the one specified in the destination group.
- `30000` is the sample interval in milliseconds. The sample interval is the time interval between two streams of data. In this example, the sample interval is 30000 milliseconds or 30 seconds.

Verify Deployment of the Subscription

The router dials out to the receiver to establish a session with each destination in the subscription. After the session is established, the router streams data to the receiver to create a data lake.

You can verify the deployment of the subscription on the router.

Procedure

1. View the model-driven telemetry configuration on the router.

   Example

```
Router#show running-config telemetry model-driven
telemetry model-driven
destination-group CPU-Health
address-family ipv4 172.0.0.0 port 57500
encoding self-describing-gpb
protocol tcp
!
sensor-group Monitor-CPU
sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
!
subscription CPU-Utilization
sensor-group-id Monitor-CPU sample-interval 30000
destination-id CPU-Health
source-interface GigabitEthernet0/0/0/0
 !
!
```

2. Verify the state of the subscription. An `Active` state indicates that the router is ready to stream data to the receiver based on the subscription.

   Example

```
Router# show telemetry model-driven subscription CPU-Utilization

Subscription:  CPU-Utilization
-------------
  State:       NA
  Source Interface:       GigabitEthernet0_0_0_0( 0x0)
  Sensor groups:
  Id: Monitor-CPU
    Sample Interval:      30000 ms
    Sensor Path:
```

```
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
    Sensor Path State:    Resolved

  Destination Groups:
  Group Id: CPU-Health
    Destination IP:       172.0.0.0
    Destination Port:     57500
    Encoding:             self-describing-gpb
    Transport:            tcp
    State:                NA
    No TLS

  Collection Groups:
  ------------------
  No active collection groups
```

The router streams data to the receiver using the subscription-based telemetry session and creates a data lake in the receiver.

3. Check for the error.

Example

```
Router#show tech-support telemetry model-driven
Thu Nov 28 12:47:53.164 UTC
++ Show tech start time: 2024-Nov-28.124753.UTC ++
Thu Nov 28 12:47:54 UTC 2024 Waiting for gathering to complete
..
Thu Nov 28 12:48:00 UTC 2024 Compressing show tech output
Show tech output available at 0/RP0/CPU0 :
/harddisk:/showtech/showtech-telemetry_model_driven-2024-Nov-28.124753.UTC.tgz
++ Show tech end time: 2024-Nov-28.124800.UTC ++
```

Operate on Telemetry Data for In-depth Analysis of the Network

You can start consuming and analyzing telemetry data from the data lake using an open-sourced collection stack. This use case uses the following tools from the collection stack:

- Pipeline is a lightweight tool used to collect data. You can download Network Telemetry Pipeline from GitHub. You define how you want the collector to interact with routers and where you want to send the processed data using `pipeline.conf` file.
- Telegraph (plugin-driven server agent) and InfluxDB (a time series database (TSDB)) stores telemetry data, which is retrieved by visualization tools. You can download InfluxDB from GitHub. You define what data that you want to include into your TSDB using the `metrics.json` file.
- Grafana is a visualization tool that displays graphs and counters for data streamed from the router.

In summary, Pipeline accepts TCP and gRPC telemetry streams, converts data and pushes data to the InfluxDB database. Grafana uses the data from InfluxDB database to build dashboards and graphs. Pipeline and InfluxDB may run on the same server or on different servers.

Consider that the router is streaming data of approximately 350 counters every 5 seconds, and Telegraf requests information from the Pipeline at 1-second intervals. The CPU usage is analyzed in three stages using:

- a single router to get initial values
- two routers to find the difference in values and understand the pattern.

- five routers to arrive at a proof-based conclusion.

This helps you make informed business decisions about deploying the infrastructure; in this case, the CPU.

Procedure

1. Start Pipeline, and enter your router credentials.

   Note

   The IP address and port that you specify in the destination-group must match the IP address and port on which Pipeline is listening.

   Example

   ```
   $ bin/pipeline -config pipeline.conf

   Startup pipeline
   Load config from [pipeline.conf], logging in [pipeline.log]

   CRYPT Client [grpc_in_mymdtrouter], [http://172.0.0.0:5432]
    Enter username: <username>
    Enter password: <password>
   Wait for ^C to shutdown
   ```

2. In the Telegraph configuration file, add the following values to read the metrics about CPU usage.

   Example

   ```
   [[inputs.cpu]]
     ## Whether to report per-cpu stats or not
     percpu = true
     ## Whether to report total system cpu stats or not
     totalcpu = true
     ## If true, collect raw CPU time metrics.
     collect_cpu_time = false
     ## If true, compute and report the sum of all non-idle CPU states.
     report_active = false
   ```

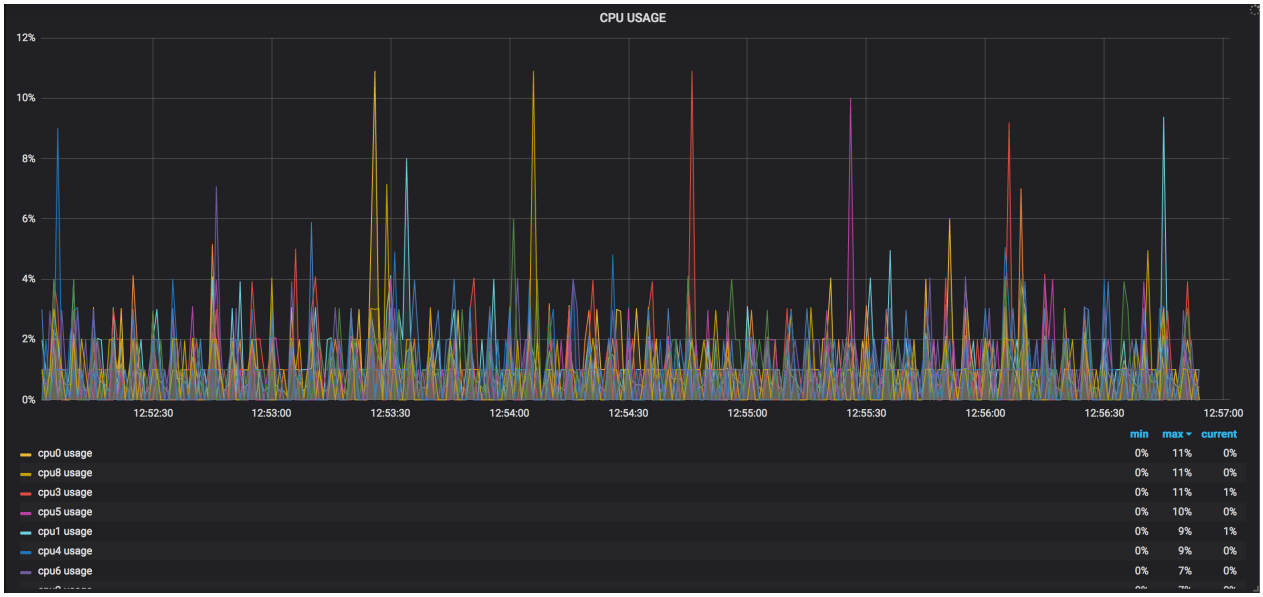
3. Use Grafana to create a dashboard and visualize data about CPU usage.

   One router

   The router pushes the counters every five seconds.

   All CPU cores are loaded equally, and there are spikes up to approximately 10 or 11 percent.

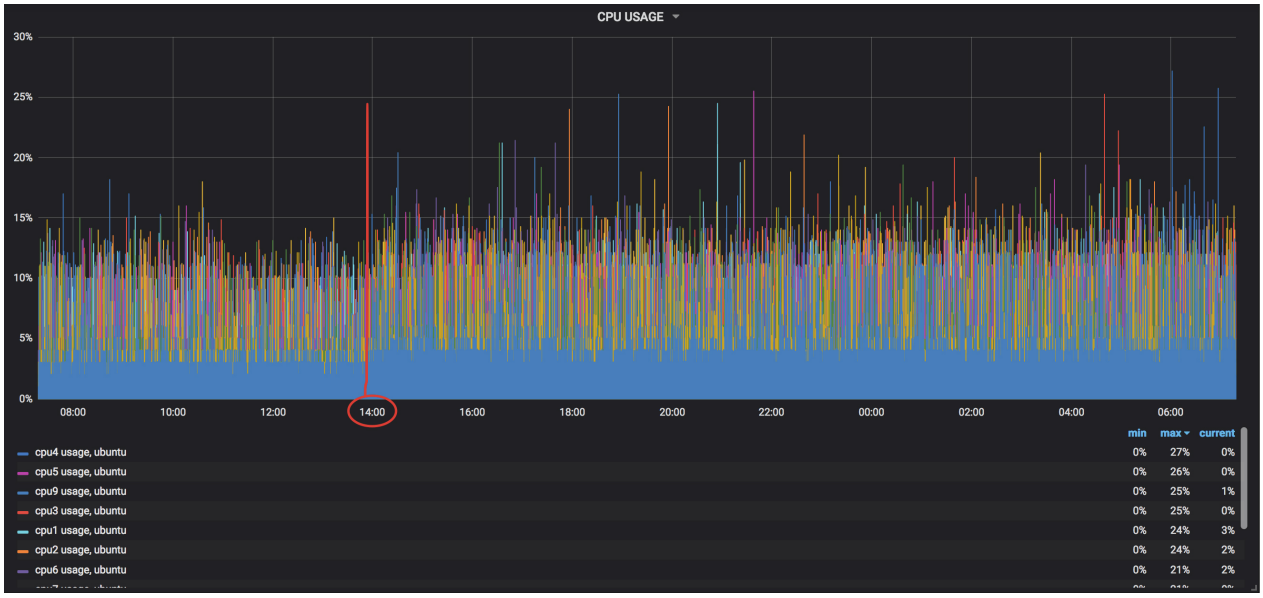
   Figure 9: CPU Usage Graph with a Single Router



Two routers

The second router is added at `14:00` in the timeline, and shows an increase in the spikes to around 25 percent with the midpoint value at 15 percent.

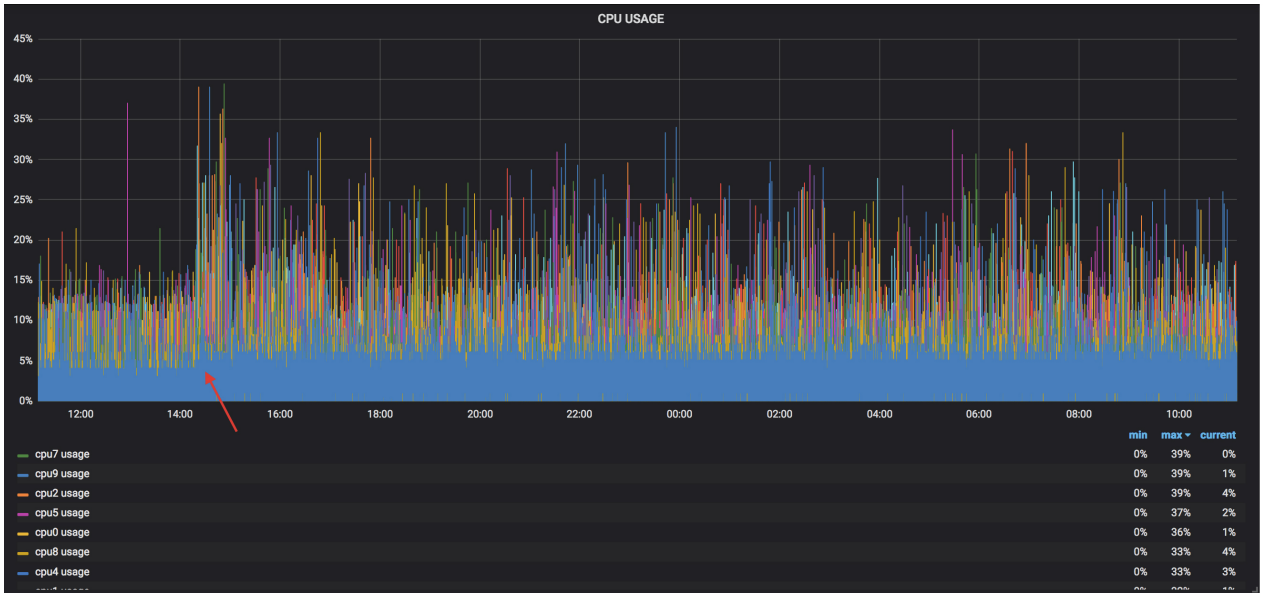
Figure 10: CPU Usage Graph with Two Routers



Five routers

With five routers, the spikes peak up to approximately 40 percent with the midpoint in the range of 22 to 25 percent.

Figure 11: CPU Usage Graph with Five Routers



369755

In conclusion, telemetry data shows that the processes are balanced almost equally across the CPU cores. There is no linear increase on a subset of cores. This analysis helps in planning the CPU utilization based on the number of counters that you stream.

5 Model-driven telemetry sessions

Topics:

- Monitor Network Parameters Using Telemetry Data for Proactive Analysis

Explains how model-driven telemetry sessions enable efficient, real-time streaming of network data from Cisco IOS XR routers to remote collectors using a subscription-based model.

A model-driven telemetry session is a network monitoring mechanism that

- streams configuration and operational data from Cisco IOS XR routers in real time,
- uses a subscription-based model to deliver structured data to remote collectors, and
- enables proactive monitoring, troubleshooting, and automation for resilient network operations.

How model-driven telemetry sessions work and their benefits

Describes how model-driven telemetry sessions stream configuration and operational data from Cisco IOS XR routers in real time, using a subscription-based model to deliver structured data to remote collectors for proactive monitoring, troubleshooting, and automation.

Streaming telemetry introduces a new paradigm for monitoring network health by providing a mechanism to efficiently stream configuration and operational data of interest from Cisco IOS XR routers. This streamed data is transmitted in a structured format to remote management stations for monitoring and troubleshooting purposes.

With telemetry data, you can create a data lake. By analyzing this data, you can proactively monitor your network, track CPU and memory utilization, identify patterns, troubleshoot your network in a predictive manner, and devise strategies to create a resilient network using automation.

Telemetry operates on a subscription model, where you subscribe to data of interest using sensor paths. Sensor paths describe OpenConfig data models or native Cisco data models. You can access the OpenConfig and Native data models for telemetry from GitHub, a software development platform that provides hosting services for version control.

You can choose who initiates the subscription by establishing a telemetry session between the router and the receiver. The session is established using either a dial-out mode or dial-in mode, as described in the Scale-Up Your Network Monitoring Strategy Using Telemetry article.

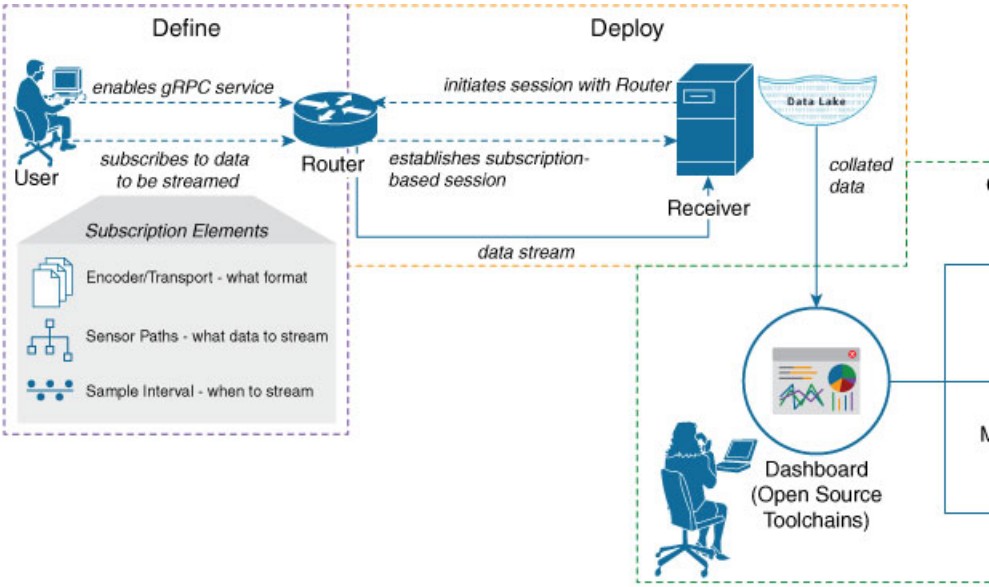
> **Note**
>
> Watch this video to discover the power of real-time network management using model-driven telemetry.

This article describes the dial-in mode, where a receiver dials in to the router to establish a telemetry session. In this mode, the receiver subscribes dynamically

to one or more sensor paths specified in a subscription. The router streams telemetry data through the same session established by the receiver. The dial-in mode of subscriptions is dynamic and terminates when the receiver cancels the subscription or when the session ends.

The following image shows a high-level overview of the dial-in mode:

Figure 12: Dial-In Mode



This article also provides a use case that illustrates simultaneous monitoring of various network parameters, showing how streaming telemetry data helps you gain better visibility of the network and make informed decisions to stabilize it.

Yang data model

You can programmatically configure a dial-in telemetry session using the openconfig-telemetry.yang OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide*.

Example of model-driven telemetry session

For instance, a network operator can use model-driven telemetry sessions to monitor CPU and memory utilization across multiple routers in real time, enabling proactive troubleshooting and automation strategies for network resilience.

Monitor Network Parameters Using Telemetry Data for Proactive Analysis

Describes how to monitor network parameters using telemetry data for proactive analysis, including prerequisites, workflow, and step-by-step configuration and verification.

The use case illustrates how, with the dial-in mode, you can use telemetry data to stream various parameters about your network. You use this data for predictive analysis where you monitor patterns, and proactively troubleshoot issues. This use case describes the tools used in the open-sourced collection stack to store and analyse telemetry data.

> **Note**
>
> Watch this video to see how you configure model-driven telemetry to take advantage of data models, open source collectors, encodings and integrate into monitoring tools.

Telemetry involves the following workflow:

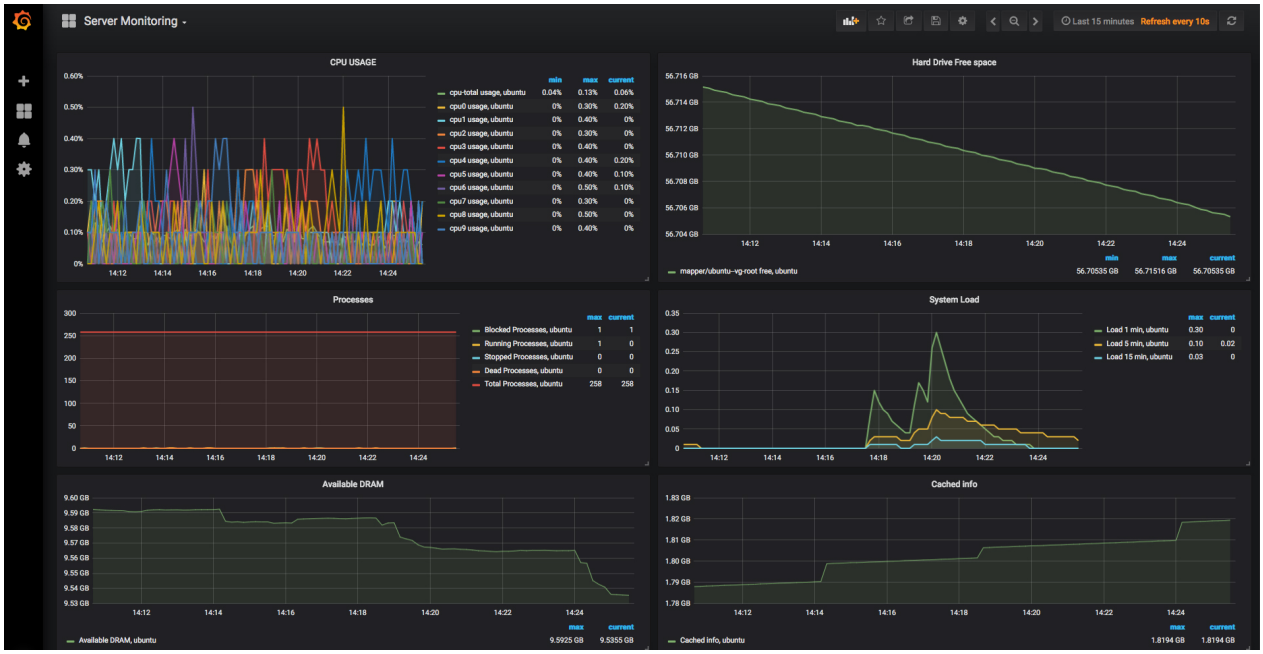
- Define: You define a subscription to stream data from the router to the receiver. To define a subscription, you create a sensor-group.
- Deploy: The receiver initiates a session with the router and establishes a subscription-based telemetry session. The router streams data to the receiver. You verify subscription deployment on the router.
- Operate: You consume and analyse telemetry data using open-source tools, and take necessary actions based on the analysis.

Figure 13: Visual Analysis of Network Health using Telemetry Data



369765

Figure 14: Visual Analysis of System Monitoring using Telemetry Data



369766

Before you begin

Ensure you meet these dependancies:

- Make sure you have L3 connectivity between the router and the receiver.
- Enable gRPC server on the router to accept incoming connections from the receiver.

```
Router#configure
Router(config)#grpc
Router(config-grpc)#port <port-number>
Router(config-grpc)#commit
```

The port-number ranges from 57344 to 57999. If a port number is unavailable, an error is displayed.

Note

Starting from Cisco IOS XR Software Release 7.4.1, Cisco IOS XR, release 7.4.1 introduces enhancements to the following gRPC in-band functionalities:

- Multi VRF support (other than default VRF). You must enable gRPC using the following configuration:

```
Router(config)#grpc
Router(config)#vrf <vrf-name>
Router(config-vrf)#port <port-number>
Router(config-vrf)#no-tls
```

  For more information, see the *Use gRPC Protocol to Define Network Operations with Data Models* chapter in the *Programmability Configuration Guide for Cisco NCS 6000 Series Routers*.

- Stream telemetry data out of physical sub-interface.
- Stream telemetry data out of bundle interface and sub-interface.

This enhancement to in-band telemetry is supported on 20-Port 100-Gbps (2T) line cards.

Procedure

1. Define a subscription to stream data from the router to the receiver. Specify the subset of the data that you want to stream from the router using sensor paths. The sensor path represents the path in the hierarchy of a YANG data model. This example uses the native data model `Cisco-IOS-XR-um-telemetry-model-driven-cfg.yang`. Create a sensor-group to contain the sensor paths.

   Example

```
sensor-group health
  sensor-path Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization

  sensor-path Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary
  sensor-path Cisco-IOS-XR-shellutil-oper:system-time/uptime
 !
sensor-group interfaces
  sensor-path
Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters

  sensor-path Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-summary
 !
sensor-group optics
  sensor-path
Cisco-IOS-XR-controller-optics-oper:optics-oper/optics-ports/optics-port/optics-info

 !
sensor-group routing
  sensor-path
Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/levels/level/adjacencies/adjacency

  sensor-path
```

```
Cisco-IOS-XR-clns-isis-oper:isis/instances/instance/statistics-global
  sensor-path
Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/protocol/isis/as/information

  sensor-path
Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/process-info

 !
sensor-group mpls-te
  sensor-path Cisco-IOS-XR-mpls-te-oper:mpls-te/tunnels/summary
  sensor-path Cisco-IOS-XR-ip-rsvp-oper:rsvp/interface-briefs/interface-brief

  sensor-path
Cisco-IOS-XR-ip-rsvp-oper:rsvp/counters/interface-messages/interface-message

 !
```

2. Subscribe to telemetry data that is streamed from a router. A subscription binds the sensor-group and sets the streaming method. The streaming method can be cadence-driven or event-driven. Separating the sensor-paths into different subscriptions enhances the efficiency of the router to retrieve operational data at scale.

   Example

   > **Note**
   >
   > The configuration for event-driven telemetry is similar to cadence-driven telemetry, with only the sample interval as the differentiator. Configuring the sample interval value to `0` (zero) sets the subscription for event-driven telemetry, while configuring the interval to any non-zero value sets the subscription for cadence-driven telemetry.

```
subscription health
  sensor-group-id health strict-timer
  sensor-group-id health sample-interval 30000
 !
subscription interfaces
  sensor-group-id interfaces strict-timer
  sensor-group-id interfaces sample-interval 30000
 !
subscription optics
  sensor-group-id optics strict-timer
  sensor-group-id optics sample-interval 30000
 !
subscription routing
  sensor-group-id routing strict-timer
  sensor-group-id routing sample-interval 30000
 !
subscription mpls-te
  sensor-group-id mpls-te strict-timer
  sensor-group-id mpls-te sample-interval 30000
 !
```

3. Verify deployment of the subscription. The receiver dials into the router to establish a dynamic session based on the subscription. After the session is established, the router streams data to the receiver to create a data lake. Verify the

state of the subscription. An `Active` state indicates that the router is ready to stream data to the receiver based on the subscription.

Example

```
Router#show telemetry model-driven subscription
Thu Jan 16 09:48:14.293 UTC
Subscription:  health                  State: Active
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  health                          30000               Resolved

Subscription:  optics                  State: NA
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  optics                          30000               Resolved

Subscription:  mpls-te                 State: NA
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  mpls-te                         30000               Resolved

Subscription:  routing                 State: NA
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  routing                         30000               Resolved

Subscription:  interfaces              State: NA
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  interfaces                      30000               Resolved

Subscription:  CPU-Utilization         State: NA
-------------
  Sensor groups:
  Id                              Interval(ms)        State
  Monitor-CPU                     30000               Resolved

  Destination Groups:
  Id                Encoding            Transport   State   Port    Vrf
 IP
  CPU-Health        self-describing-gpb tcp         NA      57500
 172.0.0.0
    No TLS
```

The router streams data to the receiver using the subscription-based telemetry session and creates a data lake in the receiver.

4. Operate on telemetry data for in-depth analysis of the network. Start Pipeline from the shell, and enter your router credentials. The streamed telemetry data is stored in InfluxDB.

Example

```
$ bin/pipeline -config pipeline.conf

Startup pipeline
Load config from [pipeline.conf], logging in [pipeline.log]

CRYPT Client [grpc_in_mymdtrouter], [http://172.0.0.0:5432]
 Enter username: <username>
 Enter password: <password>
Wait for ^C to shutdown
```

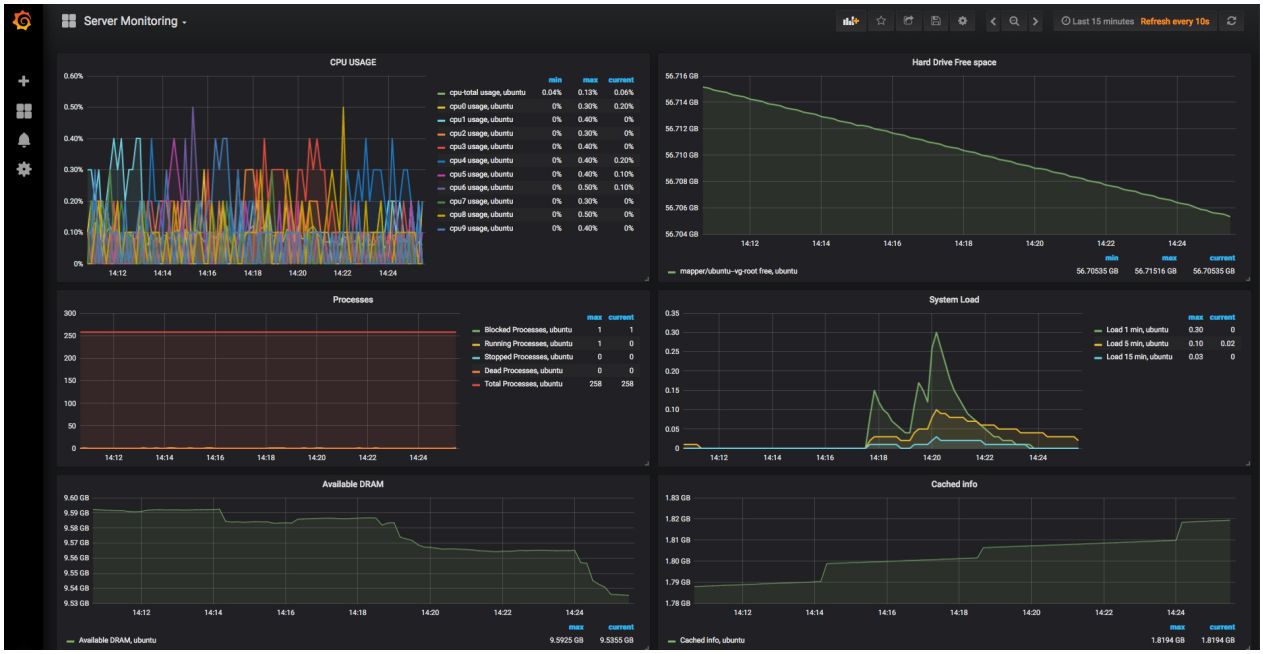
The streamed telemetry data is stored in InfluxDB.

5. Use Grafana to create a dashboard and visualize the streamed data.

Figure 15: Visual Analysis of Network Health using Telemetry Data



Figure 16: Visual Analysis of System Monitoring using Telemetry Data



In conclusion, telemetry data shows that various parameters of the network can be monitored simultaneously. This data is streamed in near real-time without affecting the performance of the network. With this data, you gain better visibility into your network.

Once completed, the router streams telemetry data to the receiver, creating a data lake for proactive analysis. You can visualize and analyze network health and performance using open-source tools such as Pipeline, InfluxDB, and Grafana.

What to do next

Review the dashboards and analysis results in Grafana and InfluxDB to proactively monitor and troubleshoot network issues.

6 Enhancements to streaming telemetry

Topics:

- Stream telemetry data about PBR decapsulation statistics
- gNMI internal path sharing

Explains enhancements to streaming telemetry that improve data collection and monitoring capabilities in network environments.

Enhancements to streaming telemetry is a network feature that improves methods for collecting, monitoring, and analyzing network information.

- Provides advanced techniques for data collection and analysis.
- Identifies relevant configuration, support, or operational details.
- Helps users understand when and how to apply the feature.

Streaming telemetry enhancement overview

Describes how streaming telemetry enhancements improve network data collection and monitoring.

Key benefits of streaming telemetry enhancements include:

- Improved visibility into network operations.
- More efficient data analysis and reporting.
- Support for advanced monitoring use cases.

Steps to leverage streaming telemetry enhancements:

1. Enable streaming telemetry on supported devices.
2. Configure relevant data sources and destinations.
3. Monitor and analyze collected telemetry data.

Common attributes and descriptions:

- Data source: Specifies the origin of telemetry information.
- Destination: Defines where telemetry data is sent.
- Format: Indicates the structure of telemetry messages.

Comparison of telemetry enhancement features:

Table 32: Streaming Telemetry Feature Comparison

Attributes	Standard Telemetry	Enhanced Telemetry
Data granularity	Basic	Advanced
Configuration flexibility	Limited	Extensive

Contrast between telemetry types:

Table 33: Telemetry Type Contrast

Push Model	Pull Model
Data sent proactively	Data requested on demand

Lookup table for telemetry enhancement features:

Table 34: Telemetry Enhancement Lookup Table

Items	Attribute 1	Attribute 2
Streaming Telemetry	Real-time data	Configurable output
SNMP Polling	Periodic data	Fixed output

Note

TIP: Review device compatibility and software requirements before enabling streaming telemetry enhancements.

Visual representation of streaming telemetry enhancements:

Streaming telemetry enhancement example

For example, enabling enhanced streaming telemetry on a router allows real-time monitoring of interface statistics and operational status.

Streaming telemetry enhancement counter-example

Using legacy SNMP polling does not provide the same real-time visibility or flexibility as enhanced streaming telemetry.

Streaming telemetry enhancement analogy

Enhanced streaming telemetry is like upgrading from periodic snapshots to a live video feed for network monitoring.

Stream telemetry data about PBR decapsulation statistics

Configure streaming of telemetry data about Policy-Based Routing (PBR) decapsulation statistics for GRE and GUE encapsulation protocols using the Cisco-IOS-XR-infra-policymgr-oper.yang data model.

Stream telemetry data about PBR decapsulation statistics for GRE and GUE encapsulation protocols to monitor packet delivery and decapsulation performance using the Cisco-IOS-XR-infra-policymgr-oper.yang data model.

- Monitor encapsulated packet statistics and identify mismatches in decapsulation counts.

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 addresses 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 the show policy-map type pbr [vrf vrf-name] address-family ipv4/ipv6 statistics command. For more information on PBR-based decapsulation, refer to *Interface and Hardware Component Configuration Guide for Cisco IOS XR, CRS, ASR 9000, NCS 6000, NCS 5000, NCS 5500, Cisco 8000, NCS 540, and NCS 560 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.

- Statistics are collected periodically and can be displayed on demand or streamed to a collector.

Procedure

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
```

Verify the configuration to ensure the correct PBR setup for the VRF.

- Confirm class-map and policy-map configuration for GRE traffic.

The running configuration displays the PBR setup for the VRF.

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, refer to the *Operate on Telemetry Data for In-depth Analysis of the Network* section in the Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure on page 36 chapter.

- Use the mdt_exec tool to stream telemetry data from the sensor path.

```
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
-------------------------------------
```

VRF statistics are displayed and telemetry data is streamed to the collector.

Telemetry data about PBR decapsulation statistics is streamed and available for monitoring and analysis at the collector.

gNMI internal path sharing

Explains how gNMI subscriptions map to Telemetry Agent terminology and how the device shares eligible telemetry collection work.

Internal Path sharing in gNMI Subscribe Remote Procedure Calls (RPCs) is a capability that

- enables a router to optimize resource usage by fetching data for common Internal Paths only once per sample interval when multiple gNMI `Subscribe RPCs` request overlapping telemetry data, and
- in steady state, ensures that each client (collector) receives an identical snapshot of the requested data, except for any encoding differences required by their specific subscriptions.

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
gNMI Internal Path sharing	Release 26.3.1	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 [ASIC: Q100, Q200, P100]) The router now automatically detects duplicate Internal Paths that appear across different gNMI subscription lists and shares collection for those paths. This optimization reduces redundant processing and provides consistent data snapshots to collectors. No user configuration is necessary for this functionality. This feature introduces these changes: **CLI:** • **show telemetry subscriptions [id <id>] [detail | internal]** • **show telemetry destination endpoints [id <id>] [detail | internal]** • **show telemetry collections [id <id>] [detail | internal]**

Internal Path sharing eligibility

Lists the conditions required for Internal Path sharing across gNMI Subscribe calls and provides the collection-instance indicator used to verify active sharing.

Internal Path sharing occurs when a device shares data collection from an Internal Path once for multiple gNMI Subscribe Remote Procedure Calls (RPCs), increasing resource efficiency.

For an Internal Path to be shared, the eligibility criteria are:

- The `SubscriptionList` mode is `STREAM`.
- The data producer for the Internal Path is EDM.
- The `SubscriptionMode` is `SAMPLE` or the `SAMPLE` collection instance of `TARGET_DEFINED`.
- For a given Internal Path across subscriptions, the `SubscriptionMode` must match across `Subscribe` calls.
- The `sample_interval` must match.
- The `Encoding` must match.
- The key-value filters must match.

- The `Path Target` must match.
- The `updates_only` must match.

View telemetry subscription, destination, and collection state

Verify the state and behavior of telemetry subscriptions, destinations, and collections for the optimized sharing design.

Use these commands to ensure telemetry subscriptions, destinations, and collections are functioning correctly by inspecting their configuration and internal state.

By default, each command shows a concise configuration view. The **detail** and **internal** options provide additional information, with **internal** focused on implementation behavior.

Before you begin

Follow these steps to verify telemetry subscription, destination, and collection state.

Procedure

1. Use the **show telemetry subscriptions detail** command to verify subscription state, sensor groups, destinations, and associated collections.

```
Router#show telemetry subscriptions detail
Mon Sep  7 03:59:33.821 UTC
Subscription: GNMI__13787445275569793446
Subscription ID: 6
-------------
  State: Active
  Collection Instances: 1
  Updates Only: False

  Sensor Groups:
  --------------
  ID: GNMI__13787445275569793446_0
    Submode:              SAMPLE
    Heartbeat Interval:   0 ms
    Sample Interval:      15000 ms
    Sensor Path:          Cisco-IOS-XR-shellutil-oper:system-time/uptime
    Sensor Path State:    Resolved
    Error:
    Filter Keys:

  Destinations:
  -------------
  99.99.1.2:53402

    Endpoint ID:                  2007
    Subscription:                 GNMI__13787445275569793446
    VRF:
    VRF ID:
    DSCP:                         Default
    Transport:                    gRPC
    Encoding:                     gNMI JSON IETF
    Compression:                  Gzip
    UDP MTU:                      0 (bytes)
    TLS:                          Enabled
    TLS Hostname:
    Initial Updates:              1/1
    State:                        Active
    Total Messages Sent:          8
```

```
    Total Bytes Sent:             770
    Pending Queue Memory:         0 (bytes)
    Pending Queue Messages:       0
    Pending On-change Messages:   0
    Processed On-change Messages: 0
    Last Message Send Time:       2026-09-07 03:59:22.434883 +0000
    Last gNMI Sync-response Time: 2026-09-07 03:57:52.434883 +0000

  99.99.1.2:53418

    Endpoint ID:                  2008
    Subscription:                 GNMI__13787445275569793446
    VRF:
    VRF ID:
    DSCP:                         Default
    Transport:                    gRPC
    Encoding:                     gNMI JSON IETF
    Compression:                  Gzip
    UDP MTU:                      0 (bytes)
    TLS:                          Enabled
    TLS Hostname:
    Initial Updates:              1/1
    State:                        Active
    Total Messages Sent:          8
    Total Bytes Sent:             770
    Pending Queue Memory:         0 (bytes)
    Pending Queue Messages:       0
    Pending On-change Messages:   0
    Processed On-change Messages: 0
    Last Message Send Time:       2026-09-07 03:59:22.435337 +0000
    Last gNMI Sync-response Time: 2026-09-07 03:57:53.822337 +0000

  Collection Instances:
  ---------------------
  ID: 9
    Sensor Path:
Cisco-IOS-XR-shellutil-oper:system-time/uptime
    Submode:                      SAMPLE
    Sample Interval:              15000 ms
    Completed Collections:        8
    Collection Elapsed Time:      Min: 3 ms Avg: 4 ms Max: 7 ms
    Data Source Elapsed Time:     Min: 3 ms Avg: 3 ms Max: 7 ms
    Cadence Drift Elapsed Time:   Min: 0 ms Avg: 0 ms Max: 0 ms
    Scheduler Elapsed Time:       Min: 0 ms Avg: 0 ms Max: 0 ms
    Last Max Collection Time:     2026-09-07 03:57:53.822883 +0000
    Last Max Data Source Time:    2026-09-07 03:57:53.822883 +0000
    Last Collection Start Time:   2026-09-07 03:59:22.429883 +0000
    Last Collection End Time:     2026-09-07 03:59:22.435883 +0000
    Missed Collections:           0
    Last Missed Collection Time:
    Collection Splits:            0
    Last Collection Split Time:
    Subscription IDs:             6
    Creation Time:                2026-09-07 03:57:52.432883 +0000

    Destinations:
    -------------
    Endpoint ID: 2007
      Matching Leaves:
      Path Target:
      Encoding:                                gNMI JSON IETF
      Join Time:                               2026-09-07 03:57:52.433175
```

```
+0000
      Initial Update Completion Time:           2026-09-07 03:57:52.436175
+0000
      Completed Collections:                    7
      Collection Elapsed Time:                  Min: 3 ms Avg: 4 ms Max: 6
ms
      Last Collection Start Time:               2026-09-07 03:59:22.430175
+0000
      Last Collection End Time:                 2026-09-07 03:59:22.436175
+0000
      Data Source Items:                        7
      Data Source Filtered Items:               0
      Encoded Items:                            7
      Last Message Send Time:                   2026-09-07 03:59:22.436175
+0000
      Sent Messages:                            7
      Sent Bytes:                               770
      Dropped Messages:                         0
      Dropped Bytes:                            0
      Slow Endpoint:                            N
      Last Slow Endpoint Time:
      Last Slow Endpoint Clear Time:
      Slow Endpoint Clears:                     0
      Scheduler Retries:                        0
      Consecutive In-time Collections:          7
      Slow Endpoint Collections:                0
      Endpoint Slowness Occurrences:            0

    Endpoint ID: 2008
      Matching Leaves:
      Path Target:
      Encoding:                                 gNMI JSON IETF
      Join Time:                                2026-09-07 03:57:52.433199
+0000
      Initial Update Completion Time:           2026-09-07 03:57:53.823199
+0000
      Completed Collections:                    7
      Collection Elapsed Time:                  Min: 3 ms Avg: 4 ms Max: 7
ms
      Last Collection Start Time:               2026-09-07 03:59:22.430199
+0000
      Last Collection End Time:                 2026-09-07 03:59:22.436199
+0000
      Data Source Items:                        7
      Data Source Filtered Items:               0
      Encoded Items:                            7
      Last Message Send Time:                   2026-09-07 03:59:22.436199
+0000
      Sent Messages:                            7
      Sent Bytes:                               770
      Dropped Messages:                         0
      Dropped Bytes:                            0
      Slow Endpoint:                            N
      Last Slow Endpoint Time:
      Last Slow Endpoint Clear Time:
      Slow Endpoint Clears:                     0
      Scheduler Retries:                        0
      Consecutive In-time Collections:          7
      Slow Endpoint Collections:                0
      Endpoint Slowness Occurrences:            0
```

- If subscriptions display the same Collection Instance ID for an Internal Path, the device is sharing that Internal Path across the corresponding `Subscribe` RPCs.
- If the Collection Instance IDs differ, the output does not indicate shared collection for that Internal Path.

2. Use the **show telemetry destination endpoints detail** command to view the destination configuration and destination-specific behavior.

```
Router#show telemetry destination endpoints detail
Fri Jul 10 08:38:17.857 UTC
Destinations:
-------------
192.168.122.1:33284
  Endpoint ID:                  2002
  Subscription:                 GNMI__2005423308499225058
  VRF:
  VRF ID:
  DSCP:                         Default
  Transport:                    gRPC
  Encoding:                     gNMI JSON IETF
  Compression:                  Gzip
  UDP MTU:                      0 (bytes)
  TLS:                          Enabled
  TLS Hostname:
  Initial Updates:              6
  State:                        Active
  Total Messages Sent:          103
  Total Bytes Sent:             37580
  Pending Queue Memory:         0 (bytes)
  Pending Queue Messages:       0
  Pending On-change Messages:   0
  Processed On-change Messages: 0
  Last Message Send Time:       2026-07-10 08:38:12.672323 +0000
  Last gNMI Sync-response Time: 2026-07-10 08:38:12.672323 +0000

192.168.122.1:33298
  Endpoint ID:                  2003
  Subscription:                 GNMI__2005423308499225058_0
  VRF:
  VRF ID:
  DSCP:                         Default
  Transport:                    gRPC
  Encoding:                     gNMI JSON IETF
  Compression:                  Gzip
  UDP MTU:                      0 (bytes)
  TLS:                          Enabled
  TLS Hostname:
  Initial Updates:              6
  State:                        Active
  Total Messages Sent:          103
  Total Bytes Sent:             37580
  Pending Queue Memory:         0 (bytes)
  Pending Queue Messages:       0
  Pending On-change Messages:   0
  Processed On-change Messages: 0
  Last Message Send Time:       2026-07-10 08:38:14.189349 +0000
  Last gNMI Sync-response Time: 2026-07-10 08:38:14.189349 +0000
```

3. Use the **show telemetry collections detail** command to collect processing details, associated destinations, collection times, and sent packets or bytes.

```
Router#show telemetry collections detail
Fri Jul 10 08:38:17.778 UTC
Collection Instances:
---------------------
ID: 7
  Sensor Path:
openconfig-interfaces:interfaces/interface/state
  Submode:                      SAMPLE
  Sample Interval:              30000 ms
  Heartbeat Interval:           0 ms
  Completed Collections:        2
  Collection Elapsed Time:      Min: 3 ms Avg: 5 ms Max: 7 ms
  Data Source Elapsed Time:     Min: 3 ms Avg: 3 ms Max: 3 ms
  Last Max Collection Time:     2026-07-10 08:38:13.177993 +0000
  Last Collection Start Time:   2026-07-10 08:38:13.174993 +0000
  Last Collection End Time:     2026-07-10 08:38:13.177993 +0000
  Missed Collections:           0
  Last Missed Collection Time:
  Collection Splits:            0
  Last Collection Split Time:
  Subscription IDs:             2, 3
  Creation Time:                2026-07-10 08:38:11.657993 +0000

  Destinations:
  -------------
  Endpoint ID: 2002
    Matching Leaves:
    Path Target:
    Encoding:                                 gNMI JSON IETF
    Join Time:                               2026-07-10 08:38:11.658646 +0000

    Initial Update Completion Time:          2026-07-10 08:38:11.665646 +0000

    Last Message Send Time:                  2026-07-10 08:38:11.665646 +0000

    Sent Messages:                            34
    Sent Bytes:                               18293
    Dropped Messages:                         0
    Dropped Bytes:                            0
    Slow Endpoint:                            N
    Last Slow Endpoint Time:
    Last Slow Endpoint Clear Time:
    Slow Endpoint Clears:                     0
    Scheduler Retries:                        0
    Consecutive In-time Collections:          1
    Slow Endpoint Collections:                0
    Endpoint Slowness Occurrences:            0

  Endpoint ID: 2003
    Matching Leaves:
    Path Target:
    Encoding:                                 gNMI JSON IETF
    Join Time:                               2026-07-10 08:38:11.658666 +0000

    Initial Update Completion Time:          2026-07-10 08:38:13.178666 +0000

    Last Message Send Time:                  2026-07-10 08:38:13.178666 +0000

    Sent Messages:                            34
    Sent Bytes:                               18293
```

```
    Dropped Messages:                          0
    Dropped Bytes:                             0
    Slow Endpoint:                             N
    Last Slow Endpoint Time:
    Last Slow Endpoint Clear Time:
    Slow Endpoint Clears:                      0
    Scheduler Retries:                         0
    Consecutive In-time Collections:           1
    Slow Endpoint Collections:                 0
    Endpoint Slowness Occurrences:             0
```

You can review whether the optimized sharing design is behaving as expected for subscriptions, destinations, and collections.

The relevant telemetry objects are visible in brief, detail, or internal form for troubleshooting and validation.