



Core Components of Model-driven Telemetry Streaming

The core components used in streaming model-driven telemetry data are:

- [Session, on page 1](#)
- [Sensor Path, on page 2](#)
- [Subscription, on page 2](#)
- [Transport and Encoding, on page 3](#)

Session

A telemetry session can be initiated using:

Dial-in Mode

In a dial-in mode, an MDT receiver dials in to the router, and subscribes dynamically to one or more sensor paths or subscriptions. The router acts as the server and the receiver is the client. The router streams telemetry data through the same session. The dial-in mode of subscriptions is dynamic. This dynamic subscription terminates when the receiver cancels the subscription or when the session terminates.

There are two methods to request sensor-paths in a dynamic subscription:

- **OpenConfig RPC model:** The `subscribe` RPC defined in the model is used to specify sensor-paths and frequency. In this method, the subscription is not associated with an existing configured subscription. A subsequent `cancel` RPC defined in the model removes an existing dynamic subscription.

Cisco IOS XR supports Google network management interface (gNMI) protocol. gNMI is a unified management protocol for streaming telemetry data using gRPC framework. This framework and protocol do not need explicit configuration, but simplifies telemetry configuration on the router by starting only the gRPC server.

The RPC is defined in the [gNMI proto file](#). The gNMI specification is published at [Github](#).

- **IOS XR MDT RPC:** IOS XR defines RPCs to subscribe and to cancel one or more configured subscriptions. The sensor-paths and frequency are part of the telemetry configuration on the router. A subscription is identified by its configured subscription name in the RPCs.

Dial-out Mode

In a dial-out mode, the router dials out to the receiver. This is the default mode of operation. The router acts as a client and receiver acts as a server. In this mode, sensor-paths and destinations 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. When a session terminates, the router continually attempts to re-establish a new session with the receiver every 30 seconds.

Sensor Path

The sensor path describes a YANG path or a subset of data definitions in a YANG model with a container. In a YANG model, the sensor path can be specified to end at any level in the container hierarchy.

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-based streaming) or when the sensor-path content changes (event-based streaming), to one or more receivers through subscribed sessions.

Subscription

A subscription binds one or more sensor paths and destinations. An MDT-capable device streams data for each sensor path at the configured frequency (cadence-based streaming) or when the sensor-path content changes (event-based streaming) to the destination.

The following example shows subscription `SUB1` that associates a sensor-group, sample interval and destination group.

```
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
```



Note

With a `strict-timer` configured for the sample interval, the data collection starts exactly at the configured time interval allowing a more deterministic behavior to stream data.

In 32-bit platforms, `strict-timer` can be configured only under the subscription. Whereas, 64-bit platforms support configuration at global level in addition to the subscription level. However, configuring at the global level will affect all configured subscriptions.

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

Transport and Encoding

The router streams telemetry data using a transport mechanism. The generated data is encapsulated into the desired format using encoders.

Model-Driven Telemetry (MDT) data is streamed through these supported transport mechanisms:

- **Google Protocol RPC (gRPC):** used for both dial-in and dial-out modes.
- **Transmission Control Protocol (TCP):** used for only dial-out mode.
- **User Datagram Protocol (UDP):** used for only dial-out mode.

The data to be streamed can be encoded into Google Protocol Buffers (GPB) or JavaScript Object Notation (JSON) encoding. In GPB, the encoding can either be compact GPB (for optimising the network bandwidth usage) or self-describing GPB. The encodings supported are:

- **GPB encoding:** configuring for GPB encoding requires metadata in the form of compiled .proto files. A .proto file describes the GPB message format, which is used to stream data. The .proto files are available in the Github repository.
 - **Compact GPB encoding:** data is streamed in compressed and non self-describing format. A .proto file corresponding to each sensor-path must be used by the receiver to decode the streamed data.
 - **Key-value (KV-GPB) encoding:** data of each sensor path streamed is in a self-describing formatted ASCII text. A single .proto file `telemetry.proto` is used by the receiver to decode any sensor path data. Because the key names are included in the streamed data, the data on the wire is much larger as compared to compact GPB encoding.
- **JSON encoding**



Note Telemetry data is streamed out of the router using an Extensible Manageability Services Daemon (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.
