SNMP Migration to Telemetry on IOS XR

Contents

Introduction

This article introduces Simple Network Management Protocol (SNMP) components, and provides a correlation between current implementations based on SNMP monitoring into Model Driven Telemetry (MDT) approach.

SNMP

SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language that is used for monitoring and managing devices in a network

Components of SNMP

The SNMP framework has the following components, which are described in the following sections:

- SNMP Manager
- SNMP Agent
- SNMP MIB

SNMP Manager

The SNMP manager is a system that controls and monitors the activities of network hosts using SNMP. The most common managing system is a network management system (NMS). The term NMS can be applied either to a dedicated device used for network management or to the applications used on such a device.

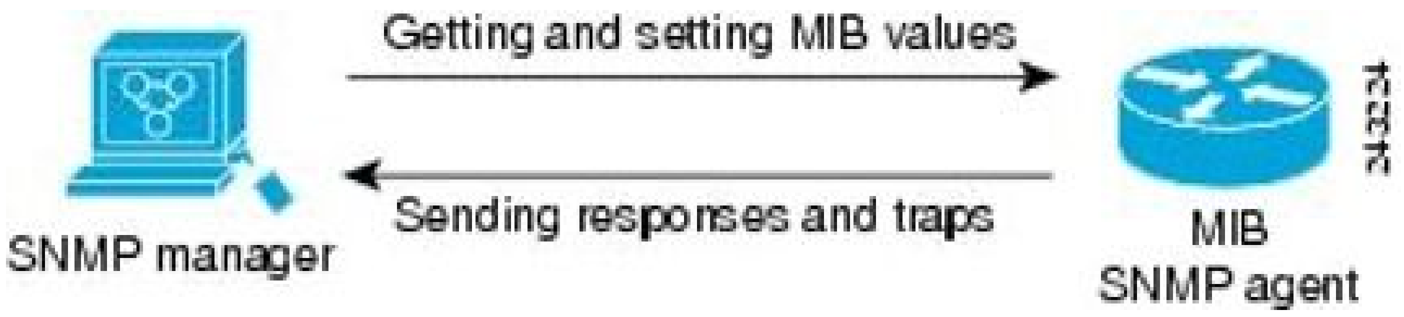
SNMP Agent

The SNMP agent is the software component within a managed device that maintains the data for the device and reports this data, as needed, to managing systems. The agent resides on the routing device (router, access server, or switch).

SNMP MIB

An SNMP agent contains MIB variables, whose values can be requested or changed by the SNMP manager through 'Get' or 'Set' operations. A manager can get a value from an agent or store a value in that agent. The agent gathers data from the SNMP MIB, the repository for information about device parameters and network data. The agent can also respond to manager requests to get or set data.

The figure below illustrates the communications between the SNMP manager and agent. A manager sends an agent requests to get and set the SNMP MIB values. The agent responds to these requests. Independent of this interaction, the agent can send the manager unsolicited notifications (traps or informs) to notify the manager about network conditions.



SNMP Operations

The SNMP applications perform the following operations to retrieve data, modify SNMP object variables, and send notifications:

- SNMP Get
- SNMP SET
- SNMP Notifications

SNMP Get

The SNMP GET operation is performed by an NMS to retrieve SNMP object variables. There are three types of GET operations:

- GET—Retrieves the exact object instance from the SNMP agent.
- GETNEXT—Retrieves the next object variable, which is a lexicographical successor to the specified variable.
- GETBULK—Retrieves a large amount of object variable data, without the need for repeated GETNEXT operations.

SNMP SET

The SNMP SET operation is performed by a NMS to modify the value of an object variable.

SNMP Notifications

A key feature of SNMP is its capability to generate unsolicited notifications from an SNMP agent.

Unsolicited (asynchronous) notifications can be generated as traps or inform requests (informs). Traps are messages alerting the Simple Network Management Protocol (SNMP) manager to a condition on the network. Informs are traps that include a request for confirmation of receipt from the SNMP manager. Notifications can indicate improper user authentication, restarts, the closing of a connection, loss of connection to a neighbor device, or other significant events.

Traps are less reliable than informs because the receiver does not send an acknowledgment when it receives a trap. The sender does not know if the trap was received. An SNMP manager that receives an inform acknowledges the message with an SNMP response Protocol Data Unit (PDU). If the sender never receives a response, the inform can be sent again. Thus, informs are more likely to reach their intended destination.

Traps are often preferred even though they are less reliable because informs consume more resources in the device and the network. Unlike a trap, which is discarded as soon as it is sent, an inform must be held in memory until a response is received or the request times out. Also, traps are sent only once, whereas an inform may be resent several times. The retries increase traffic and contribute to higher overhead on the network. Use of traps and informs requires a trade-off between reliability and resources.

MIBs and RFCs

Management Information Base (MIB) modules typically are defined in Request for Comments (RFC) documents submitted to the Internet Engineering Task Force (IETF), an international standards body. RFCs are written by individuals or groups for consideration by the Internet Society and the Internet community as a whole, usually with the intention of establishing a recommended Internet standard. Before being given RFC status, recommendations are published as Internet Draft (I-D) documents. RFCs that have become recommended standards are also labeled as standards documents (STDs). You can learn about the standards process and the activities of the IETF at the Internet Society website at http://www.isoc.org. You can read the full text of all RFCs, I-Ds, and STDs referenced in Cisco documentation at the IETF website at http://www.ietf.org.

The Cisco implementation of SNMP uses the definitions of MIB II variables described in RFC 1213 and definitions of SNMP traps described in RFC 1215.

Cisco provides its own private MIB extensions with every system. Cisco enterprise MIBs comply with the guidelines described in the relevant RFCs unless otherwise noted in the documentation. You can find the MIB module definition files and the list of MIBs supported on each Cisco platform on the Cisco MIB website on Cisco.com.

Versions of SNMP

Currently Cisco devices support the following versions of SNMP:

- SNMPv1—Simple Network Management Protocol: a full Internet standard, defined in RFC 1157. (RFC 1157 replaces the earlier versions that were published as RFC 1067 and RFC 1098.) Security is based on community strings.
- SNMPv2c—The community string-based Administrative Framework for SNMPv2. SNMPv2c (the "c" is for "community") is an experimental Internet protocol defined in RFC 1901, RFC 1905, and RFC 1906. SNMPv2c is an update of the protocol operations and data types of SNMPv2p (SNMPv2 Classic) and uses the community-based security model of SNMPv1.
- SNMPv3—Version 3 of SNMP. SNMPv3 is an interoperable standards-based protocol defined in RFCs 3413 to 3415. SNMPv3 provides secure access to devices by authenticating and encrypting packets over the network.

The security features provided in SNMPv3 are as follows:

- Message integrity—Ensuring that a packet has not been tampered with in transit.
- Authentication—Determining that the message is from a valid source.
- Encryption—Scrambling the contents of a packet to prevent it from being learned by an unauthorized source.

Both SNMPv1 and SNMPv2c use a community-based form of security. The community of SNMP managers are able to access the agent MIB is defined by a community string.

SNMPv2c support includes a bulk retrieval mechanism and detailed error message reporting to management stations. The bulk retrieval mechanism supports the retrieval of tables and large quantities of information, minimizing the number of round trips required. The SNMPv2c improved error handling support includes expanded error codes that distinguish different types of errors; these conditions are reported through a single error code in SNMPv1. The following three types of exceptions are also reported: no such object, no such instance, and end of MIB view.

SNMPv3 is a security model in which an authentication strategy is set up for a user and the group in which the user resides. A security level is the permitted level of security within a security model. A combination of a security model and a security level determines which security mechanism is employed when handling an SNMP packet.

Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. The table below lists the combinations of security models and levels and their meanings.

Model	**Level**	**Authentication**	**Encryption**	What **Happens**
v1	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.
v3	authNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.
v3	authPriv	MD5 or SHA	Data Encryption Standard (DES)	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard.

An SNMP agent should be implemented in order to use the version of SNMP supported by the management station. An agent can communicate with multiple managers.

SNMPv3 supports RFCs 1901 to 1908, 2104, 2206, 2213, 2214, and 2271 to 2275. For additional information about SNMPv3, see RFC 2570, Introduction to Version 3 of the Internet-standard Network Management Framework (this is not a standards document).

Yang models

Yang models represent a tree-structured abstraction of a specific feature or hardware characteristics of a system. In network elements, a Yang model could represent a routing protocol, internal physical sensors arrays. YANG language and terminology is described on RFC 6020 and next updated on RFC 7950. In high-level, a Yang model organizes the data representing the main structure into submodules and containers that are a list of sub-nodes related. Several node types are explained next.

A leaf node contains simple data like an integer or a string. It has exactly one value of a particular type and no child nodes.

```
leaf host-name {

          type string;

          description "Hostname for this system";
```

```
    }
```

A leaf-list is a sequence of leaf nodes with exactly one value of a particular type per leaf.

```
leaf-list domain-search {

         type string;

         description "List of domain names to search";

      }
```

A container node is used to group related nodes in a subtree. A container has only child nodes and no value. A container may contain any number of child nodes of any type (including leafs, lists, containers, and leaf-lists).

```
container system {

         container login {

             leaf message {

                 type string;

                 description

                     "Message given at start of login session";

             }

         }

     }
```

A list defines a sequence of list entries. Each entry is like a structure or a record instance and is uniquely identified by the values of its key leafs. A list can define multiple key leafs and may contain any number of child nodes of any type (including leafs, lists, containers etc.).

Finally, a sample model that binds all these note types together looks like the following example:

```
## Contents of "example-system.yang"
module example-system {
  yang-version 1.1;
  namespace "urn:example:system";
  prefix "sys";
  organization "Example Inc.";
  contact "joe@example.com";
  description "The module for entities implementing the Example system.";
  revision 2007-06-09 {
    description "Initial revision.";
  }
  container system {
    leaf host-name {
      type string;
      description "Hostname for this system.";
    }
    leaf-list domain-search {
      type string;
      description "List of domain names to search.";
    }
    container login {
      leaf message {
        type string;
        description "Message given at start of login session.";
      }
      list user {
        key "name";
        leaf name {
           type string;
        }
        leaf full-name {
          type string;
        }
        leaf class {
          type string;
        }
      }
    }
  }
}
```

However, the Yang language used on Yang Models does not indicate the organization of the data into containers/list/leafs. This is why a certain feature on a network element could be represented with diverse Yang models. This challenge has been addressed with the following Yang Models Types:

- OpenConfig Models
- Native Models

OpenConfig Models

OpenConfig Models were developed using agnostic vendor organization for the model representing a specific feature, the benefit of this approach is that a NMS could use these models for interacting with network elements on multi-vendor or even multi-platform environment.

As the name states, these models are open and are publicly available for inspecting on repositories like github on this link:

https://github.com/openconfig/public/tree/master/release/models

As an example, you can find an openconfig model for Border Gateway Protocol (BGP), another for Link Aggregation Control Protocol (LACP) and a different one for ISIS, with different specific model. In the case of BGP you can find a model for BGP errors, another one for BGP policy and so on. The models could be related, and some models can call another yang package. For example, openconfig-bgp-neighbor.yang belongs to openconfig-bgp.yang:

```
module openconfig-bgp {
   yang-version "1";

   ## namespace
   namespace "http://openconfig.net/yang/bgp";
   prefix "oc-bgp";


  ## import some basic inet types
  import openconfig-extensions { prefix oc-ext; }
  import openconfig-rib-bgp { prefix oc-bgprib; }

  ## Include the OpenConfig BGP submodules
  ## Common: defines the groupings that are common across more than
  ## one context (where contexts are neighbor, group, global)
  include openconfig-bgp-common;
  ## Multiprotocol: defines the groupings that are common across more
  ## than one context, and relate to Multiprotocol
  include openconfig-bgp-common-multiprotocol;
  ## Structure: defines groupings that are shared but are solely used for
  ## structural reasons.
  include openconfig-bgp-common-structure;
  ## Include peer-group/neighbor/global - these define the groupings
  ## that are specific to one context
  include openconfig-bgp-peer-group;
  include openconfig-bgp-neighbor;
  include openconfig-bgp-global;

<snip>
```

To sum up, OpenConfig Models are oriented for protocols common to all platforms, like IETF or RFC standardized features.

Native Models

In contrast, Native models are vendor-oriented models that cover in depth structures specific to a particular platform. For example, models that group sensors of physical values inside a network element like voltages, temperatures, ASIC counters, Fabric counters and so on. Since they depend on the platform, it is common to find models specific for NCS6K, ASR9K or Cisco 8000.

As OpenConfig Models, Native Models are also available in Github repository:

https://github.com/YangModels/yang/tree/master/vendor/cisco/xr

As these models tend to be much more specific and complete than OpenConfig models, they are tied to specific software version, and subject to change between software releases.

There are two main categories for Native Models:

- “Oper” models, used to retrieve information from an element.

For example, Cisco-IOS-XR-eigrp-oper.yang

- "Cfg" models, used to configure a network element

For example, Cisco-IOS-XR-eigrp-cfg.yang

In general terms, Model Driven Telemetry uses "oper" models to stream data from infrastructure and NMS like NSO uses "cfg" models to make changes in the configuration on network elements.

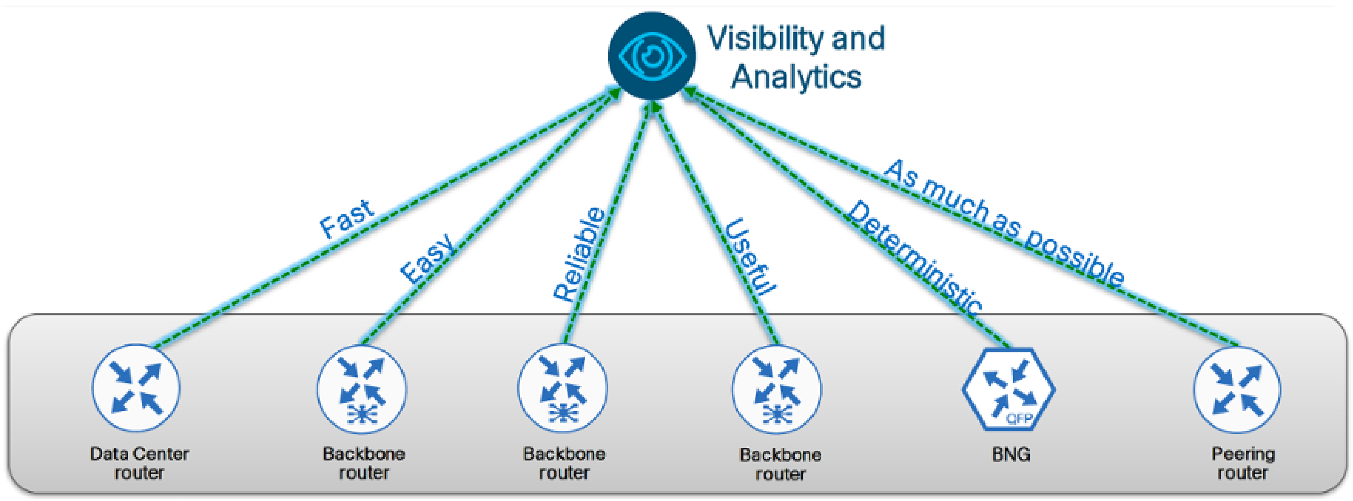
Native and OpenConfig Yang Models are present on XR software on /pkg/yang folder and can be listed to find out if any Yang Model is available on a platform. This example is for XRrv9k running cXR 6.4.2:

```
RP/0/RP0/CPU0:xrv9k1#run ls  /pkg/yang | grep isis

Tue Sep 22 14:21:27.471 CLST

Cisco-IOS-XR-clns-isis-cfg.yang

Cisco-IOS-XR-clns-isis-datatypes.yang

Cisco-IOS-XR-clns-isis-oper-sub1.yang

Cisco-IOS-XR-clns-isis-oper-sub2.yang

Cisco-IOS-XR-clns-isis-oper-sub3.yang

Cisco-IOS-XR-clns-isis-oper.yang

Cisco-IOS-XR-isis-act.yang

openconfig-isis-lsdb-types.yang

openconfig-isis-lsp.yang

openconfig-isis-policy.yang

openconfig-isis-routing.yang

openconfig-isis-types.yang

openconfig-isis.yang



RP/0/RP0/CPU0:xrv9k1#
```

Telemetry

Telemetry is a process that allows collecting information from different remote elements into a central location that aggregate visibility and analytics layer.

In networking environments, the data could be produced by every element in the network, routers, swiches between others and the information could be related to a very large set of specific protocols, performance counters or measures from physical sensors.



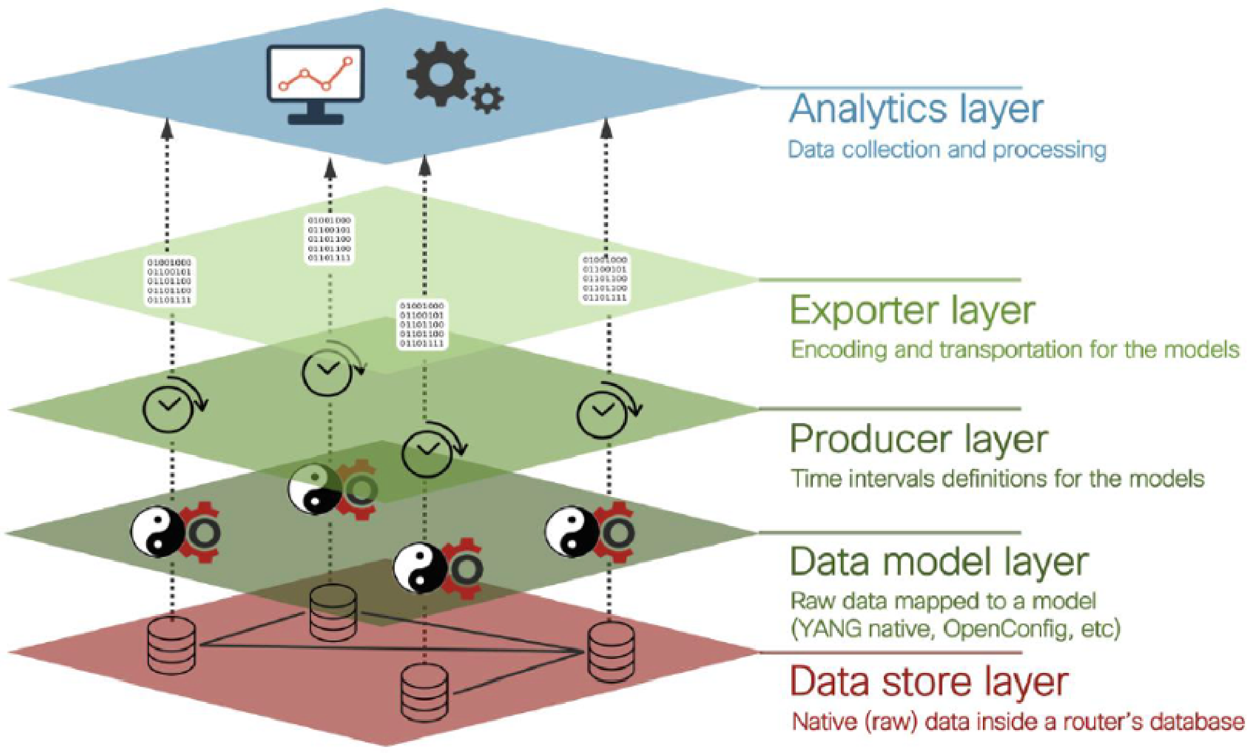
In general, Visibility and Analytics functions are located in central points in networks, streaming of telemetry information is made using networking transport mechanisms, so telemetry information should be fast as possible allowing to scale.

As opposed to SNMP legacy mechanisms, Telemetry uses a Push paradigm, where the network should be provisioned to stream its own data without being polled at regular intervals, which is the main characteristic of SNMP based monitoring. This provision it is often called subscription, and it is based on a set of variables to be monitored, the regular interval for the sampling interval for data collection, and the remote system to send this data across the network.

Model Driven Telemetry

MDT states for Model Driven Telemetry, and as the name says, it is based on Yang Models. Every aspect in network equipment could be represented with Yang Models, for example OSPF Neighbors table, RIB or Temperature Sensors for each component on modular systems.

Regarding MDT architecture, it can be divided in the following layers:


Analytics layer
Data collection and processing
Exporter layer
Encoding and transportation for the models
Producer layer
Time intervals definitions for the models
Data model layer
Raw data mapped to a model
(YANG native, OpenConfig, etc)
Data store layer
Native (raw) data inside a router's database

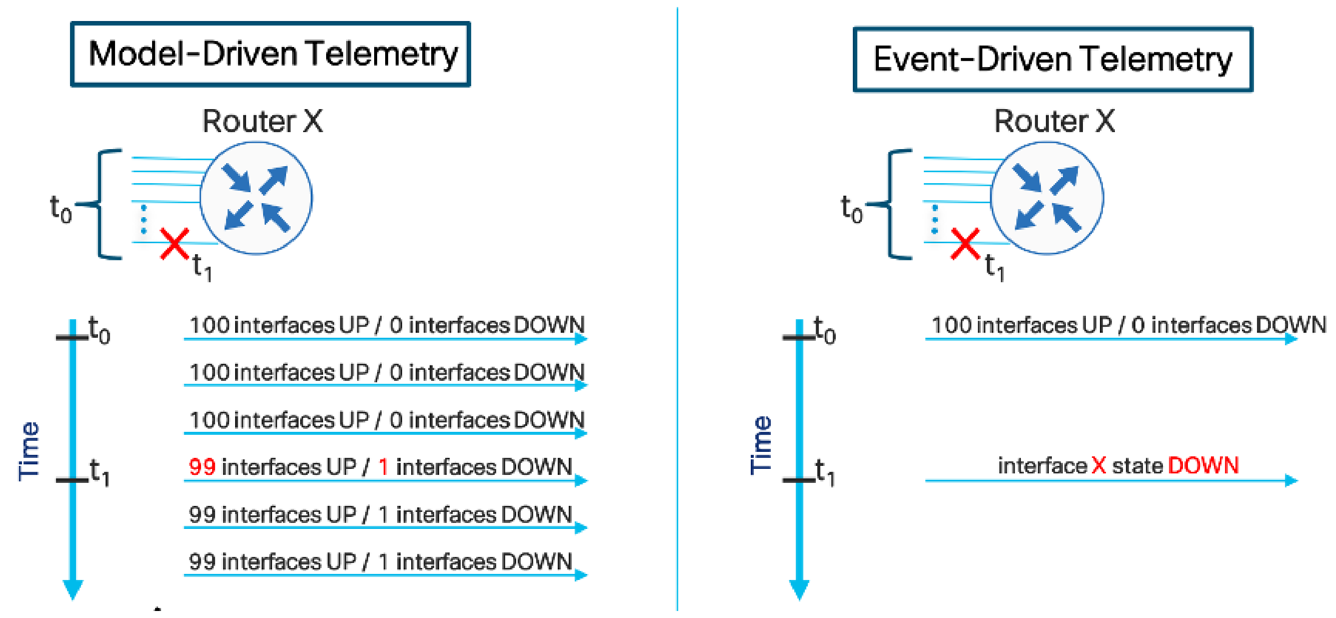
> **Note**: Regarding the producer layer, in model driven telemetry there is a sampling-interval definition that controls how often the device consult the internal database for raw data and organizes this data into the data-model layer.

The telemetry subscription also defines which models and with containers/path would produce data to be streamed into the Analytics Layer. This definition would impact in relevant information to business purposes. MDT definition of this sensor-path would be analogue to define OID to retrieve via SNMP, since both techinques produces structured data at defined sampling-rate.

Event Driven Telemetry

EDT stands for Event Driven Telemetry and is also based on Yang models for the structure. The main difference is that the trigger for the collection and data stream is not regular interval, but is a specific event, like threshold cross, link events, hardware failure, and so on.

A comparison from an event with Model Driven Telemetry and Event Driven Telemetry is presented next:

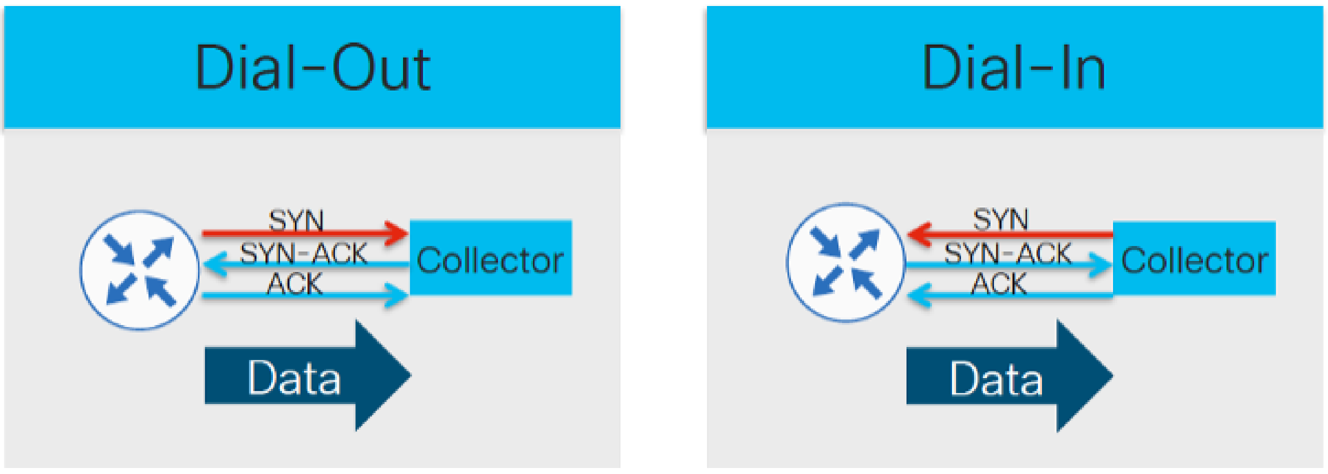

Model-Driven Telemetry
Router X
t0
t1
Time
t0
t1
100 interfaces UP / 0 interfaces DOWN
100 interfaces UP / 0 interfaces DOWN
100 interfaces UP / 0 interfaces DOWN
99 interfaces UP / 1 interfaces DOWN
99 interfaces UP / 1 interfaces DOWN
99 interfaces UP / 1 interfaces DOWN
Event-Driven Telemetry
Router X
t0
t1
Time
t0
t1
100 interfaces UP / 0 interfaces DOWN
interface X state DOWN

Tip: This figure shows redundant messages using MDT, but only messages representing changes by using EDT.

Transport

Telemetry should be reliable as possible, so makes sense using Transmission Control Protocol (TCP) based transport for using session-oriented sockets between the infrastructure and the Analytics layer, which should implement collectors for making the session.

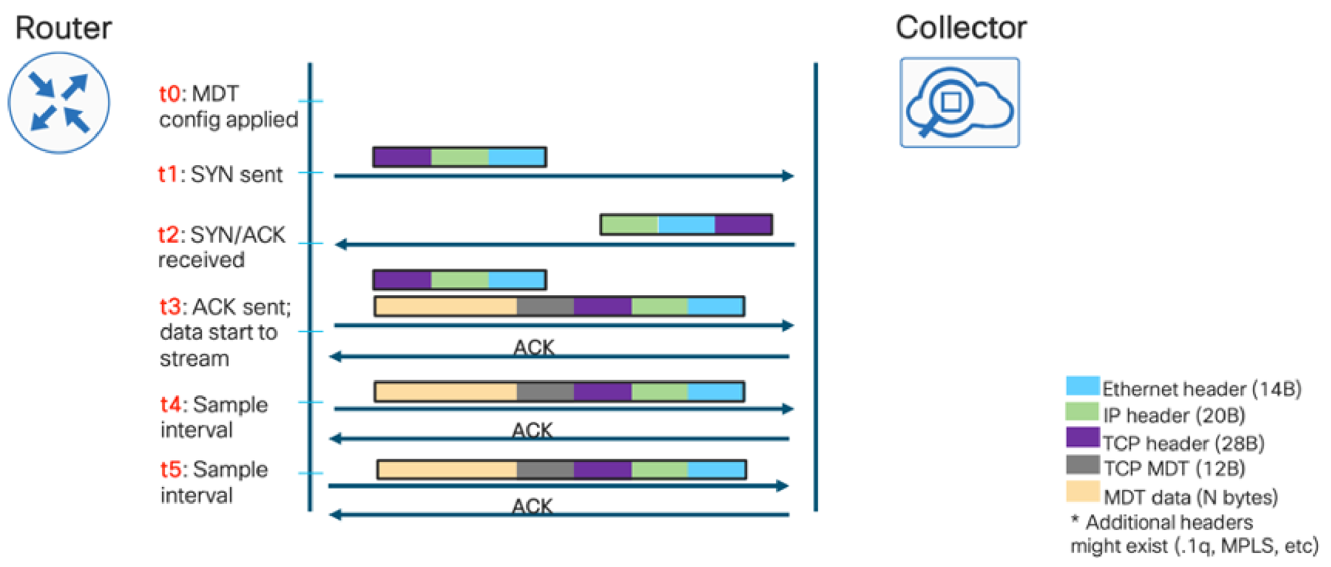
There are two main approaches when using telemetry, and they differ between each other in the 3-way-handshake initial flow.



Note: In Dial-Out mode, the setup of the session is started on the infrastructure side, which implies that the sensors of interest should be configured on the network elements. In constrast, Dial-In approach allow a lighter configuration on network elements since the collector should ask for specific sensor paths at setup phase.

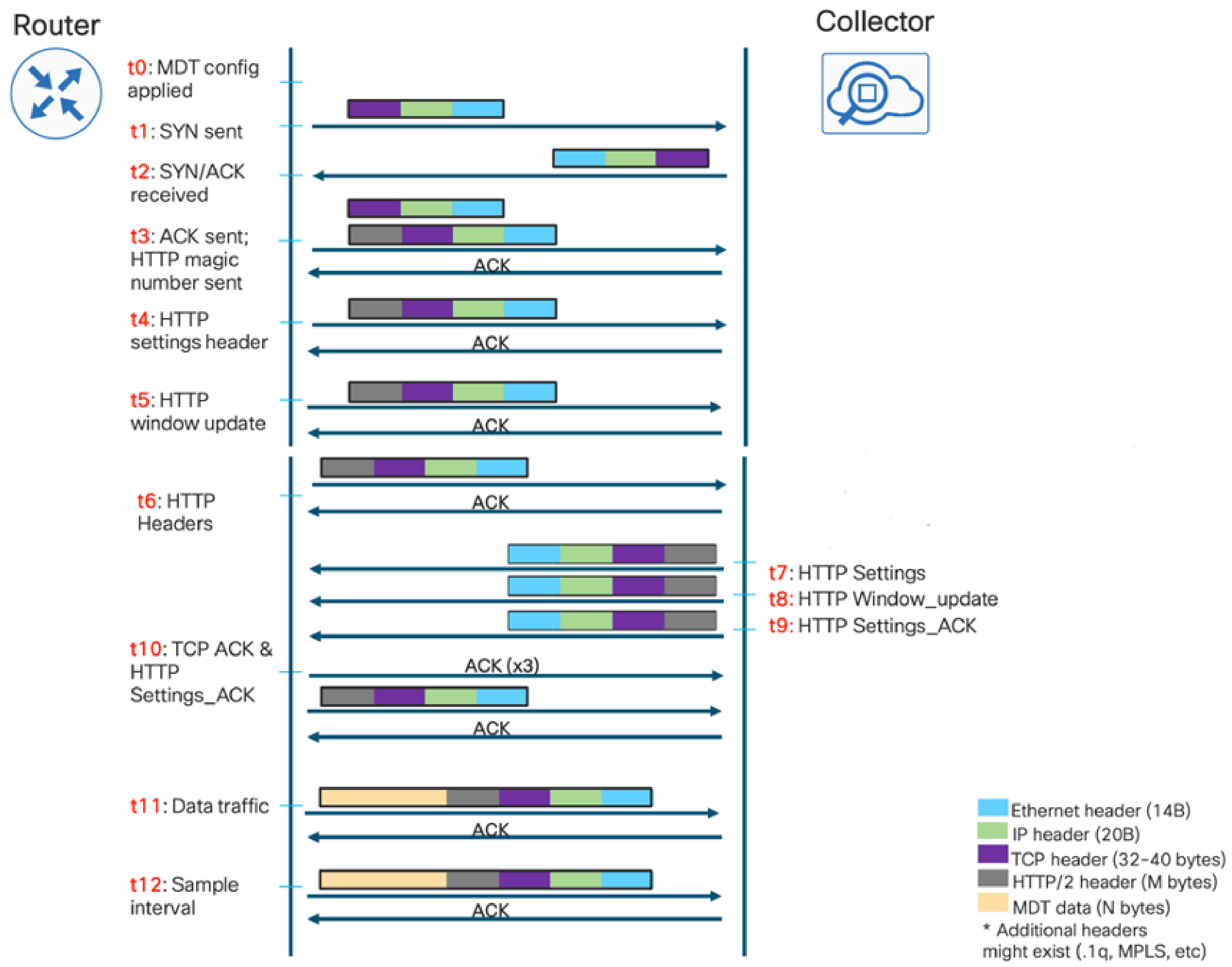
TCP

TCP is the simplest way of making a connection-oriented session between a network element and a telemetry collector, and the stream of data starts from Router to Collector who sent ACK back to the router for reliability purposes:



gRPC

Since Google Protocol RPC (gRPC) works over Hypertext Transfer Protocol/2 (HTTP/2), the session itself should form at setup, and allows speed control from the collector side natively:



gNMI/gNOI

gRPC Network Management Interface (gNMI) is gRPC network management protocol developed by Google. gNMI provides the mechanism to install, manipulate, and delete the configuration of network devices, and also to view operational data. The content provided through gNMI can be modeled using YANG.

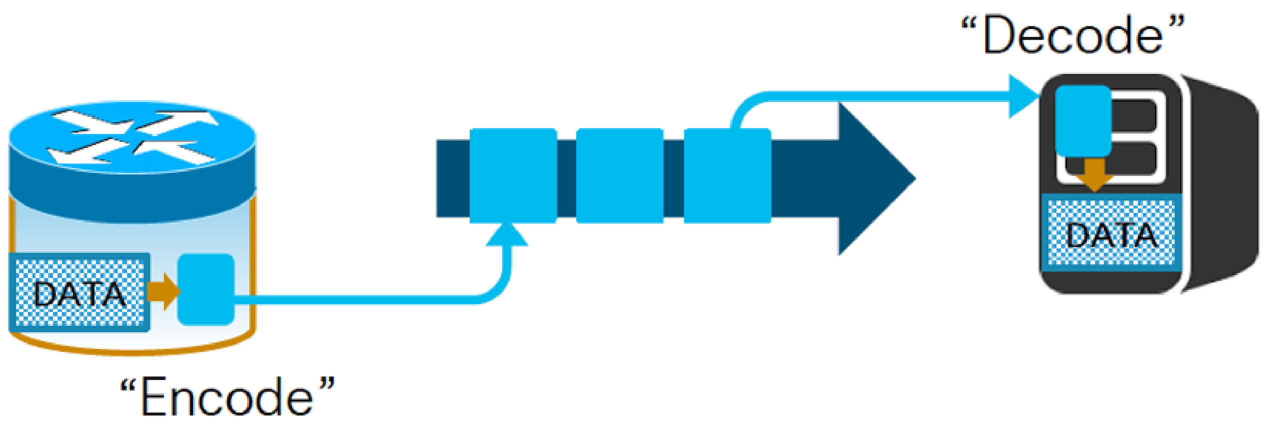
gNMI uses gRPC-HTTP/2 to setup a connection and provides bi-directional channel between network elements and a NMS that could also be a telemetry collector, but also provides and interface to manage the devices.

Between the operations supported by this protocol, we can find gNMI Get, gNMI Set that return the information requested, success or error messages.

gRPC Network Operations Interface (gNOI) is a collection of microservices that uses the same communication channel as gNMI but allows generic operations not related to configuration itself like ping, reboot, change SSL certificates, clearing, etc.

Encoding

Yang models define the structure of the data, its hierarchy, and the type of every leaf node on it. However, modeling does not indicate how this data should be serialized. This process governate the conversion from structured data into a stream of bytes to be sent over the TCP connection (raw TCP, gRPC, gNMI, etc).



> **Note**: This process should be implemented with an equivalent mechanism in the network element who should encode the data, and the collector should decode this data.

JSON

The first encoding mechanism is native JavaScript Object Notation (JSON) format, which is well-known but is human-oriented as it has every key represented as string which is inefficient in terms of message size. The major benefit of using JSON is that is easy to parse, and read as is text-based as the next example:

```
{
```

```
“node_id_str":"test-IOSXR ",
"subscription_id_str":" if_rate",
"encoding_path":"Cisco-IOS-XR-infra-statsdoper:infra-statistics/interfaces/interface/latest/datarate",
"collection_start_time":1510716302467,
"msg_timestamp":1510716302479,
"data_json":
[
 {
   "timestamp":1510716282334,
   "keys":{
           "interface-name":"Null0”
              },
   "content":{
           "input-data-rate":0,
           "input-packet-rate":0,
           "output-data-rate":0,
           "output-packet-rate":0,
       <>
{
  "timestamp": 1510716282344,
  "keys":{
          "interface-name":"GigabitEthernet0/0/0/0”
             },
 "content":{
          "input-data-rate":8,
          "input-packet-rate":1,
          "output-data-rate":2,
          "output-packet-rate":0,
          <>
 "collection_end_time":1510716302372
}
```

GPB-KV

Google Protocol Buffers-Key Value (GPB-KV) Encoding format it is also called self-describing GPB because it makes use of protocol buffers to make usage of messages that point to particular elements on Yang models. This implies that only one .proto file is needed to encode/decode purposes, and the keys itself from the data are in self-described strings.

```
node_id_str: “test-IOSXR"
subscription_id_str: ”if_rate"
encoding_path: "Cisco-IOS-XR-infra-statsd-oper:infrastatistics/interfaces/interface/latest/data-rate"
collection_id: 3
collection_start_time: 1485793813366
msg_timestamp: 1485793813366
data_gpbkv {
  timestamp: 1485793813374
  fields {
   name: "keys"
   fields {
    name: "interface-name" string_value: "Null0"
    }
   }
fields {
  name: "content"
  fields { name: "input-data-rate" 8: 0 }
  fields { name: "input-packet-rate" 8: 0 }
  fields { name: "output-data-rate" 8: 0 }
  fields { name: "output-packet-rate" 8: 0 }
  <>
data_gpbkv {
  timestamp: 1485793813389
  fields {
    name: "keys"
    fields { name: "interface-name" string_value: "GigabitEthernet0/0/0/0" }
    }
fields {
  name: "content"
  fields { name: "input-data-rate" 8: 8 }
  fields { name: "input-packet-rate" 8: 1 }
  fields { name: "output-data-rate" 8: 2 }
  fields { name: "output-packet-rate" 8: 0 }
  <>
}
...
collection_end_time: 1485793813405
```

GPB

Finally, Google Protocol Buffers (GPB), also called compact GPB, takes this approach one step futher and requires .proto files to map every key of the structure making it much more efficient in terms of message size since everything is sent as binary values. However, the drawback is the need of compiling every .proto file associated to every Yang model supported by infrastructure/collector.

```
node_id_str: ”test-IOSXR"
subscription_id_str: ”if_rate"
encoding_path: "Cisco-IOS-XR-infra-statsdoper:infrastatistics/interfaces/interface/latest/data-rate"
collection_id: 5
collection_start_time: 1485794640452
msg_timestamp: 1485794640452
data_gpb {
  row {
   timestamp: 1485794640459
   keys: "\n\005Null0"
   content: "\220\003\000\230\003\000\240\003\000\250\0 03\000\260\003\000\270\003\000\300\003\000\ 310
   }
  row {
   timestamp: 1485794640469
   keys: "\n\026GigabitEthernet0/0/0/0"
   content: "\220\003\010\230\003\001\240\003\002\250\0 03\000\260\003\000\270\003\000\300\003\000\ 310
   }
collection_end_time: 1485794640480
```

MDT Configuration in IOS XR

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

- Session
- Sensor Path
- Subscription

- Transport and Encoding

The session options can be Dial-in or Dial-out as we discussed previously. In order to build the configuration in IOS XR.

Dial-Out Mode

for Dial-Out mode, the router initiates a session to the destinations based on the subscription, and the process should include following steps:

- Create a Destination Group
- Create a Sensor Group
- Create a Subscription
- Validate Dial-out configuration

In order to create a destination group, you need to know the Internet Protocol Version 4 (IPv4) / Internet Protocol Version 6 (IPv6) address of the collector and the port that would service this application. Also, you need to specify the protocol and the encoding that should be agreed on the network device and the collector.

Finally, you may need to specify the Virtual Routing and Forwarding (VRF) used to communicate to the collector network address.

Next, an example of Dial-Out configuration is presented:

```
telemetry model-driven

 destination-group DG1

  vrf MGMT

  address-family ipv4 192.168.122.20 port 5432

   encoding self-describing-gpb

   protocol tcp

  !

 !
```

Encoding options are presented next:

<#root>

```
RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#encoding ?

  gpb                    GPB encoding

  json                   JSON encoding
  self-describing-gpb  Self describing GPB encoding
← Also known as GPB-KV


RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#encoding
```

The protocols options:

```
RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#protocol ?

  grpc  gRPC

  tcp   TCP

  udp   UDP

RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#protocol grpc ?

  gzip          gRPC gzip message compression

  no-tls        No TLS

  tls-hostname  TLS hostname

  <cr>

RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#protocol tcp ?

  <cr>

RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#protocol udp ?

  packetsize  UDP packet size

  <cr>

RP/0/RP0/CPU0:C8000-1(config-model-driven-dest-addr)#protocol udp
```

TCP protocol is straightforward and only need the port settings attached to the IPv4/IPv6 address. User Datagram Protocol (UDP) in contrast is connectionless, so the destination group status would always be active.

Compression in gRPC can be achieved by the usage of the optional ***gzip*** keyword. gRPC uses TLS by default, so a certificate should be installed locally on the router for this usage. This behaviour can be override by configuration of ***no-tls*** keyword. Finally, you can specify a different hostname for certificate purposes using ***tls-hostname*** keyword.

Next, sensor-group section should be added listing the sensor-paths of our interest. This section is straightforward but is important to know that the sensor-path itself allows for filtering to optimize several resources like Central Processing Unit (CPU) and bandwidth.

```
telemetry model-driven

 sensor-group SG1

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

  sensor-path Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[interface-na

 !

!
```

Note: The format needed for a sensor path is <model-name>:<container-path>

This document presents the mapping from SNMP based monitoring using OID representing “leaves” in this legacy approach into YANG models, represented with XPATHs that matches the same “leaves”.

The final configuration stage should be configuring a subscription, which ties the sensor group with a cadence for the telemetry streaming to a destination group.

```
telemetry model-driven

 subscription SU1

  sensor-group-id SG1 sample-interval 5000

  destination-id DG1

 !

!
```

This example uses a sampling interval of 5000 milliseconds (5 seconds) which is relative to the end of the previous collection. To change this behavior, you can change ***sample-interval*** keyword with ***strict-timer*** option.

For verification, you can use the following command that covers subscription status. This method allows covering sensor-group and destination-group information also.

```
RP/0/RP0/CPU0:C8000-1#sh telemetry model-driven subscription SU1

Wed Nov 18 15:38:01.397 UTC

Subscription:  SU1

-------------

  State:       ACTIVE

  Sensor groups:

  Id: SG1
  Sample Interval:      5000 ms
  Heartbeat Interval:   NA
  Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[
  Sensor Path State:    Resolved
  Sensor Path:          Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
  Sensor Path State:    Resolved


Destination Groups:
Group Id: DG1
  Destination IP:       192.168.122.10
  Destination Port:     5432
  Destination Vrf:      MGMT(0x60000001)
  Encoding:             self-describing-gpb
  Transport:            tcp
  State:                Active
  TLS :                 False
  Total bytes sent:     636284346
  Total packets sent:   4189
  Last Sent time:       2020-11-18 15:37:58.1700077650 +0000


Collection Groups:
------------------
  Id: 9
  Sample Interval:      5000 ms
  Heartbeat Interval:   NA
  Heartbeat always:     False
  Encoding:             self-describing-gpb
  Num of collection:    1407
  Collection time:      Min:     4 ms Max:    13 ms
  Total time:           Min:     8 ms Avg:    10 ms Max:    20 ms
```

```
    Total Deferred:       0
    Total Send Errors:    0
    Total Send Drops:     0
    Total Other Errors:   0
    No data Instances:    1407
    Last Collection Start:2020-11-18 15:37:57.1699545994 +0000
    Last Collection End:  2020-11-18 15:37:57.1699555589 +0000
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/

    Id: 10
    Sample Interval:      5000 ms
    Heartbeat Interval:   NA
    Heartbeat always:     False
    Encoding:             self-describing-gpb
    Num of collection:    1391
    Collection time:      Min:   178 ms Max:   473 ms
    Total time:           Min:   247 ms Avg:   283 ms Max:   559 ms
    Total Deferred:       0
    Total Send Errors:    0
    Total Send Drops:     0
    Total Other Errors:   0
    No data Instances:    0
    Last Collection Start:2020-11-18 15:37:58.1699805906 +0000
    Last Collection End:  2020-11-18 15:37:58.1700078415 +0000
    Sensor Path:          Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization

RP/0/RP0/CPU0:C8000-1#
```

Dial-In Mode

In Dial In mode, the collector initiates the connection to the network elements. Then, the collector should indicate the interest to build a subscription.

The configuration has the following steps:

- Enable gRPC service
- Setup Sensor groups
- Verification

To enable the gRPC service, the configuration is displayed next:

```
!

grpc

 vrf MGMT

 port 57400

 no-tls

 address-family dual

!
```

The options are straightforward, including the VRF, and the TCP port. By default, gRPC uses TLS but it can be disabled with ***no-tls*** keyword. Finally, the ***address-family dual*** option allows connection using IPv4 and IPv6.

Next, Dial-in requires the definition of sensor-groups locally, that would be used by the collector later to define a subscription.

```
telemetry model-driven

 sensor-group SG3

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

  sensor-path Cisco-IOS-XR-fib-common-oper:fib-statistics/nodes/node/drops

 !

!
```

At this, point the configuration for Dial-In mode, is complete, and the collector itself can make a subscription to the router using gRPC. For verification, you can make the same approach as in dial-out mode:

```
RP/0/RP0/CPU0:C8000-1#sh telemetry model-driven subscription anx-1605878175837
```

```
Fri Nov 20 13:58:37.894 UTC

Subscription:  anx-1605878175837

-------------

  State:       ACTIVE

  Sensor groups:

  Id: SG3

    Sample Interval:      15000 ms

    Heartbeat Interval:   NA

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

    Sensor Path State:    Resolved

    Sensor Path:          Cisco-IOS-XR-fib-common-oper:fib-statistics/nodes/node/drops

    Sensor Path State:    Resolved



  Destination Groups:

  Group Id: DialIn_1003

    Destination IP:       192.168.122.10

    Destination Port:     46974

    Compression:          gzip

    Encoding:             json

    Transport:            dialin

    State:                Active

    TLS :                 False

    Total bytes sent:     71000035

    Total packets sent:   509

    Last Sent time:       2020-11-20 13:58:32.1030932699 +0000



  Collection Groups:

  ------------------

    Id: 5

    Sample Interval:      15000 ms
```

```
Heartbeat Interval:   NA
Heartbeat always:     False
Encoding:             json
Num of collection:    170
Collection time:      Min:   273 ms Max:   640 ms
Total time:           Min:   276 ms Avg:   390 ms Max:   643 ms
Total Deferred:       0
Total Send Errors:    0
Total Send Drops:     0
Total Other Errors:   0
No data Instances:    0
Last Collection Start:2020-11-20 13:58:32.1030283276 +0000
Last Collection End:  2020-11-20 13:58:32.1030910008 +0000
Sensor Path:          Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization


Id: 6
Sample Interval:      15000 ms
Heartbeat Interval:   NA
Heartbeat always:     False
Encoding:             json
Num of collection:    169
Collection time:      Min:    15 ms Max:    33 ms
Total time:           Min:    17 ms Avg:    22 ms Max:    33 ms
Total Deferred:       0
Total Send Errors:    0
Total Send Drops:     0
Total Other Errors:   0
No data Instances:    0
Last Collection Start:2020-11-20 13:58:32.1030910330 +0000
Last Collection End:  2020-11-20 13:58:32.1030932787 +0000
Sensor Path:          Cisco-IOS-XR-fib-common-oper:fib-statistics/nodes/node/drops
```

```
RP/0/RP0/CPU0:C8000-1#
```

Tip: Note that no cadence, encoding, collector IP or transport is hardcoded on the router for dial-in mode.

SNMP Migration to MDT

In order to accomplish the migration from traditional SNMP into Telemetry model, the following aspects should be covered:

- MIB migration into XPATH
- Trap migration into Telemetry
- Security considerations

MIB migration into XPATH

For this purpose, we could categorize MIB using its own hierarchy which could be mapped (at least on high level) to a particular functionality.

BGP4-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to BGP peering sessions.

OID Name	OID Number	OID Description	XPATH
bgpPeerLastError	1.3.6.1.2.1.15.3.1.14	The last error code and subcode seen by this peer on this connection. If no error has occurred, this field is zero. Otherwise, the first byte of this two byte OCTET STRING contains the error code, and the second byte contains the	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/neighbor-missing-eor-table/neighbor/last-notify-error-code

		subcode.	
bgpPeerOutUpdates	1.3.6.1.2.1.15.3.1.11	The number of BGP UPDATE messages transmitted on this connection.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/update-messages-out
bgpPeerInUpdates	1.3.6.1.2.1.15.3.1.10	The number of BGP UPDATE messages received on this connection.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/update-messages-in
bgpPeerNegotiatedVersion	1.3.6.1.2.1.15.3.1.4	The negotiated version of BGP running between the two peers. This entry MUST be zero (0) unless the bgpPeerState is in the openconfirm or the established state. Note that legal values for this object are between 0 and 255.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/negotiated-protocol-version
bgpPeerState	1.3.6.1.2.1.15.3.1.2	The BGP peer connection state.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-state
bgpPeerRemoteAddr	1.3.6.1.2.1.15.3.1.7	The remote IP address of this entry's BGP peer.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-remote-

			address
bgpPeerLocalAddr	1.3.6.1.2.1.15.3.1.5	The local IP address of this entry's BGP connection.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-local-address
bgpPeerFsmEstablishedTime	1.3.6.1.2.1.15.3.1.16	This timer indicates how long (in seconds) this peer has been in the established state or how long since this peer was last in the established state. It is set to zero when a new peer is configured or when the router is booted.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-established-time
bgpPeerAdminStatus	1.3.6.1.2.1.15.3.1.3	The desired state of the BGP connection. A transition from 'stop' to 'start' will cause the BGP Manual Start Event to be generated. A transition from 'start' to 'stop' will cause the BGP Manual Stop Event to be generated. This parameter can be used to restart BGP peer	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-admin-status

		connections. Care should be used in providing write access to this object without adequate authentication.	

CISCO-BGP4-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to BGP session state and prefix interchanged.

OID Name	**OID Number**	**OID Description**	**XPATH**
cbgpPeer2RemoteAs	1.3.6.1.4.1.9.9.187.1.2.5.1.11	The remote autonomous system number received in the BGP OPEN message.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/sessions/session/remote-as
cbgpPeer2PrevState	1.3.6.1.4.1.9.9.187.1.2.5.1.29	The BGP peer connection previous state.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/previous-connection-state
cbgpPeer2State	1.3.6.1.4.1.9.9.187.1.2.5.1.3	The BGP peer connection state.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-state
cbgpPeer2LocalAddr	1.3.6.1.4.1.9.9.187.1.2.5.1.6	The local IP address of this entry's BGP connection.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/connection-local-address
cbgpPeer2AdvertisedPrefixes	1.3.6.1.4.1.9.9.187.1.2.8.1.6	This counter is incremented	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-

		when a route prefix, which belongs to an address family is advertised on this connection. It is initialized to zero when the connection is undergone a hard reset.	table/neighbor/af-data/prefixes-advertised
cbgpPeer2AcceptedPrefixes	1.3.6.1.4.1.9.9.187.1.2.8.1.1	Number of accepted route prefixes on this connection, which belong to an address family.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/af-data/prefixes-accepted
cbgpPeerPrefixLimit	1.3.6.1.4.1.9.9.187.1.2.1.1.3	Max number of route prefixes accepted on this connection	Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active/default-vrf/afs/af/neighbor-af-table/neighbor/af-data/max-prefix-limit
cbgpPeer2PrefixThreshold	1.3.6.1.4.1.9.9.187.1.2.8.1.4	Prefix threshold value (%) for an address family on this connection at which warning message stating the prefix count is crossed the threshold or corresponding SNMP notification is generated.	Cisco-IOS-XR-ipv4-bgp-oper:bgp/config-instances/config-instance/config-instance-default-vrf/entity-configurations/entity-configuration/af-dependent-config/max-prefix-warn-threshold

CISCO-CLASS-BASED-QOS-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to statistics in Quality of Service (QoS) classes/policies.

OID Name	**OID Number**	**OID Description**	**XPATH**
cbQosCMDropBitRate	1.3.6.1.4.1.9.9.166.1.15.1.1.18	The bit rate of the drops per class as the result of all features that can produce drops (e.g., police, random detect, etc.).	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/general-stats/total-drop-rate Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics/class-stats/general-stats/total-drop-rate
cbQosCMDropPkt64	1.3.6.1.4.1.9.9.166.1.15.1.1.14	The 64 bits counter of dropped pkts per class as the result of all features that can produce drops (e.g., police, random detect, etc.).	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/general-stats/total-drop-packets Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics/class-stats/general-stats/total-drop-packets
cbQosCMPrePolicyPkt64	1.3.6.1.4.1.9.9.166.1.15.1.1.3	The 64 bits count of inbound packets prior to executing any QoS policies.	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/general-stats/pre-policy-matched-packets Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics/class-

			stats/general-stats/pre-policy-matched-packets
cbQosCMName	1.3.6.1.4.1.9.9.166.1.7.1.1.1	Name of the Classmap.	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/class-name
cbQosCMPostPolicyByte64	1.3.6.1.4.1.9.9.166.1.15.1.1.10	The 64 bits count of outbound octets after executing QoS policies.	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/child-policy/class-stats/general-stats/transmit-bytes Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics/class-stats/child-policy/class-stats/general-stats/transmit-bytes
cbQosIfIndex	1.3.6.1.4.1.9.9.166.1.1.1.1.4	ifIndex for the interface to which this service is attached. This field makes sense only if the logical interface has a snmp ifIndex. For e.g. the value of this field is meaningless when the cbQosIfType is controlPlane.	Cisco-IOS-XR-infra-policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type/policy-maps
cbQosConfigIndex	1.3.6.1.4.1.9.9.166.1.5.1.1.2	An arbitrary	Cisco-IOS-XR-infra-

		(system-assigned) config (instance independent) index for each Object. Each objects having the same configuration share the same config index.	policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type/policy-maps
cbQosCMPrePolicyByte64	1.3.6.1.4.1.9.9.166.1.15.1.1.6	The 64 bits count of inbound octets prior to executing any QoS policies.	Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/input/service-policy-names/service-policy-instance/statistics/class-stats/child-policy/class-stats/general-stats/pre-policy-matched-bytes Cisco-IOS-XR-qos-ma-oper:qos/interface-table/interface/output/service-policy-names/service-policy-instance/statistics/class-stats/child-policy/class-stats/general-stats/pre-policy-matched-bytes

CISCO-ENHANCED-MEMPOOL-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to the memory usage.

OID Name	OID Number	OID Description	XPATH
cempMemPoolUsed	1.3.6.1.4.1.9.9.221.1.1.1.1.7	Indicates the number of bytes from the memory pool that are currently in use by applications on the physical entity.	Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary
cempMemPoolHCUsed	1.3.6.1.4.1.9.9.221.1.1.1.1.18	Indicates the number of bytes from the	Cisco-IOS-XR-nto-misc-oper:memory-

		memory pool that are currently in use by applications on the physical entity. This object is a 64-bit version of cempMemPoolUsed.	summary/nodes/node/detail/total-used
cempMemPoolHCFree	1.3.6.1.4.1.9.9.221.1.1.1.1.20	Indicates the number of bytes from the memory pool that are currently unused on the physical entity. This object is a 64-bit version of cempMemPoolFree.	Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/detail/free-physical-memory

CISCO-ENTITY-FRU-CONTROL-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to the field replaceable units on the monitored system.

OID Name	**OID Number**	**OID Description**	**XPATH**
cefcFRUPowerOperStatus	1.3.6.1.4.1.9.9.117.1.1.2.1.2	Operational FRU power state.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/power-operational-state
cefcFRUPowerAdminStatus	1.3.6.1.4.1.9.9.117.1.1.2.1.1	Administratively desired FRU power state.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/power-administrative-st
cefcModuleStatusLastChangeTime	1.3.6.1.4.1.9.9.117.1.2.1.1.4	The value of sysUpTime at the time the cefcModuleOperStatus is changed.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/last-operational-state-ch
cefcModuleUpTime	1.3.6.1.4.1.9.9.117.1.2.1.1.8	This object provides the up time for the module since it was last re-initialized. This object is not persistent; if a module reset, restart, power off, the up time starts from zero.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/card-up-time

cefcModuleResetReason	1.3.6.1.4.1.9.9.117.1.2.1.1.3	This object identifies the reason for the last reset performed on the module.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/card-reset-reason
cefcModuleOperStatus	1.3.6.1.4.1.9.9.117.1.2.1.1.2	This object shows the module's operational state.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/card-operational-state
cefcModuleAdminStatus	1.3.6.1.4.1.9.9.117.1.2.1.1.1	This object provides administrative control of the module.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity info/card-administrative-stat

CISCO-ENTITY-SENSOR-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to sensor entities on the node.

OID Name	**OID Number**	**OID Description**	**XPATH**
entSensorValue	1.3.6.1.4.1.9.9.91.1.1.1.1.4	This variable reports the most recent measurement seen by the sensor. To correctly display or interpret this variable's value, you must also know entSensorType, entSensorScale, and entSensorPrecision. However, you can compare entSensorValue with the threshold values given in entSensorThresholdTable without any semantic knowledge.	Cisco-IOS-XR-invmgr-oper:inventory/entities/ent sensor-info/value
entSensorThresholdEvaluation	1.3.6.1.4.1.9.9.91.1.2.1.1.5	This variable indicates the result of the most recent evaluation of the threshold. If the threshold condition is true, entSensorThresholdEvaluation is true(1). If the threshold condition is false, entSensorThresholdEvaluation is false(2). Thresholds are evaluated at the rate indicated by entSensorValueUpdateRate.	Cisco-IOS-XR-invmgr-oper:inventory/entities/ent

CISCO-FLASH-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to flash storage on the system.

OID Name	**OID Number**	**OID Description**	**XPATH**
ciscoFlashPartitionName	1.3.6.1.4.1.9.9.10.1.1.4.1.1.10	Flash partition name used to refer to a partition by the system. This can be any alpha-numeric character string of the form AAAAAAAAnn, where A represents an optional alpha character and n a numeric character. Any numeric characters must always form the trailing part of the string. The system will strip off the alpha characters and use the numeric portion to map to a partition index. Flash operations get directed to a device partition based on this name. The system has a concept of a default partition. This would be the first partition in the device. The system directs an operation to the default partition whenever a partition name is not specified. The partition name is therefore mandatory except when the operation is being done on the default partition, or the device has just one partition (is not partitioned).	Cisco-IOS shellutil-filesystem oper:file-system/no system/typ
ciscoFlashPartitionSizeExtended	1.3.6.1.4.1.9.9.10.1.1.4.1.1.13	Flash partition size. It should be an integral multiple of ciscoFlashDeviceMinPartitionSize. If there is a single partition, this size will be equal to ciscoFlashDeviceSize. This object is a 64-bit version of ciscoFlashPartitionSize	Cisco-IOS shellutil-filesystem oper:file-system/no system/siz
ciscoFlashPartitionFreeSpaceExtended	1.3.6.1.4.1.9.9.10.1.1.4.1.1.14	Free space within a Flash partition. Note that the actual size of a file in Flash includes a small overhead that represents the file system's file header. Certain file systems may also have a partition or device header overhead to be considered	Cisco-IOS shellutil-filesystem oper:file-system/no system/fre

		when computing the free space. Free space will be computed as total partition size less size of all existing files (valid/invalid/deleted files and including file header of each file), less size of any partition header, less size of header of next file to be copied in. In short, this object will give the size of the largest file that can be copied in. The management entity will not be expected to know or use any overheads such as file and partition header lengths, since such overheads may vary from file system to file system. Deleted files in Flash do not free up space. A partition may have to be erased in order to reclaim the space occupied by files. This object is a 64-bit version of ciscoFlashPartitionFreeSpace	

CISCO-PROCESS-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related CPU Usage and resource allocation for processes.

OID Name	**OID Number**	**OID Description**	**XPATH**
cpmCPUTotal1minRev	1.3.6.1.4.1.9.9.109.1.1.1.1.7	The overall CPU busy percentage in the last 1 minute period. This object deprecates the object cpmCPUTotal1min and increases the value range to (0..100).	Cisco-IOS-XR-wdsysm fd-oper:system-monitoring/cpu-utilization/total-cpu-on minute
cpmCPUTotal5minRev	1.3.6.1.4.1.9.9.109.1.1.1.1.8	The overall CPU busy percentage in the last 5 minute period. This object deprecates the object cpmCPUTotal5min and increases the value range to (0..100).	Cisco-IOS-XR-wdsysm fd-oper:system-monitoring/cpu-utilization/total-cpu-fiv minute
cpmCPUTotal15minRev	1.3.6.1.4.1.9.9.109.1.1.1.1.31	The overall CPU busy percentage in the last 15 minute period. This object	Cisco-IOS-XR-wdsysm fd-oper:system-monitoring/cpu-

		deprecates the object cpmCPUTotal15min and increases the value range to (0..100).	utilization/total-cpu-fift minute
cpmProcessName	1.3.6.1.4.1.9.9.109.1.2.1.1.2	The name associated with this process. If the name is longer than 32 characters, it will be truncated to the first 31 characters, and a `*' will be appended as the last character to imply this is a truncated process name.	Cisco-IOS-XR-wdsysm fd-oper:system-monitoring/cpu-utilization/process-cpu/process-name
cpmProcessTextSegmentSize	1.3.6.1.4.1.9.9.109.1.2.3.1.15	This indicates the text memory of a process and all its shared objects.	Cisco-IOS-XR-procme oper:processes-memory/nodes/node/pr ids/process-id/text-seg-
cpmProcessDynamicMemorySize	1.3.6.1.4.1.9.9.109.1.2.3.1.18	This indicates the amount of dynamic memory being used by the process.	Cisco-IOS-XR-procme oper:processes-memory/nodes/node/pr ids/process-id/dyn-limi
cpmProcessDataSegmentSize	1.3.6.1.4.1.9.9.109.1.2.3.1.16	This indicates the data segment of a process and all its shared objects.	Cisco-IOS-XR-procme oper:processes-memory/nodes/node/pr ids/process-id/data-seg-
cpmProcExtMemAllocatedRev	1.3.6.1.4.1.9.9.109.1.2.3.1.1	The sum of all the dynamically allocated memory that this process has received from the system. This includes memory that may have been returned. The sum of freed memory is provided by cpmProcExtMemFreedRev. This object deprecates cpmProcExtMemAllocated.	Cisco-IOS-XR-procme oper:processes-memory/nodes/node/pr ids/process-id
cpmProcExtMemFreedRev	1.3.6.1.4.1.9.9.109.1.2.3.1.2	The sum of all memory that this process has returned to the system. This object deprecates cpmProcExtMemFreed.	Cisco-IOS-XR-procme oper:processes-memory/nodes/node/pr ids/process-id

ENTITY-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related physical entities on the system.

OID Name	**OID Number**	**OID Description**	**XPATH**
entPhysicalName	1.3.6.1.2.1.47.1.1.1.1.7	The textual name of the physical entity. The value of this object should be the name of the component as assigned by the local device and should be suitable for use in commands entered at the device's `console'. This might be a text name, such as `console' or a simple component number (e.g., port or module number), such as `1', depending on the physical component naming syntax of the device. If there is no local name, or this object is otherwise not applicable, then this object contains a zero-length string. Note that the value of entPhysicalName for two physical entities will be the same in the event that the console interface does not distinguish between them, e.g., slot-1 and the card in slot-1.	Cisco-IOS-XR-snmp-entitymib-oper:entity-physical-index
entLogicalDescr	1.3.6.1.2.1.47.1.2.1.1.2	A textual description of the logical entity. This object should contain a string which identifies the manufacturer's name for the logical entity, and should be set to a distinct value for each version of the logical entity.	Cisco-IOS-XR-snmp-agent-oper:snmp/information/system-name/
entPhysicalDescr	1.3.6.1.2.1.47.1.1.1.1.2	A textual description of	Cisco-IOS-XR-snmp-agent-

		physical entity. This object should contain a string which identifies the manufacturer's name for the physical entity, and should be set to a distinct value for each version or model of the physical entity.	oper:snmp/Cisco-IOS-XR-snmp-entity oper:entity-mib/entity-physical-indexes
entPhysicalContainedIn	1.3.6.1.2.1.47.1.1.1.1.4	The value of entPhysicalIndex for the physical entity which 'contains' this physical entity. A value of zero indicates this physical entity is not contained in any other physical entity. Note that the set of 'containment' relationships define a strict hierarchy; that is, recursion is not allowed. In the event a physical entity is contained by more than one physical entity (e.g., double-wide modules), this object should identify the containing entity with the lowest value of entPhysicalIndex.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/unique-id
entPhysicalClass	1.3.6.1.2.1.47.1.1.1.1.5	An indication of the general hardware type of the physical entity. An agent should set this object to the standard enumeration value which most accurately indicates the general class of the physical entity, or the primary class if there is more than one. If no appropriate standard registration identifier exists for this physical entity, then the value 'other(1)' is returned. If the value is unknown by this agent, then the value 'unknown(2)' is returned.	Cisco-IOS-XR-invmgr-oper:inventory/entities

entPhysicalHardwareRev	1.3.6.1.2.1.47.1.1.1.1.8	The vendor-specific hardware revision string for the physical entity. The preferred value is the hardware revision identifier actually printed on the component itself (if present). Note that if revision information is stored internally in a non-printable (e.g., binary) format, then the agent must convert such information to a printable format, in an implementation-specific manner. If no specific hardware revision string is associated with the physical component, or this information is unknown to the agent, then this object will contain a zero-length string.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/hardware-revision
entPhysicalFirmwareRev	1.3.6.1.2.1.47.1.1.1.1.9	The vendor-specific firmware revision string for the physical entity. Note that if revision information is stored internally in a non-printable (e.g., binary) format, then the agent must convert such information to a printable format, in an implementation-specific manner. If no specific firmware programs are associated with the physical component, or this information is unknown to the agent, then this object will contain a zero-length string.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/firmware-revision
entPhysicalSoftwareRev	1.3.6.1.2.1.47.1.1.1.1.10	The vendor-specific software revision string for the physical entity. Note that if revision	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/software-revision

		information is stored internally in a non-printable (e.g., binary) format, then the agent must convert such information to a printable format, in an implementation-specific manner. If no specific software programs are associated with the physical component, or this information is unknown to the agent, then this object will contain a zero-length string.	
entPhysicalSerialNum	1.3.6.1.2.1.47.1.1.1.1.11	The vendor-specific serial number string for the physical entity. The preferred value is the serial number string actually printed on the component itself (if present). On the first instantiation of an physical entity, the value of entPhysicalSerialNum associated with that entity is set to the correct vendor-assigned serial number, if this information is available to the agent. If a serial number is unknown or non-existent, the entPhysicalSerialNum will be set to a zero-length string instead. Note that implementations which can correctly identify the serial numbers of all installed physical entities do not need to provide write access to the entPhysicalSerialNum object. Agents which cannot provide non-volatile storage for the entPhysicalSerialNum	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/serial-number

		strings are not required to implement write access for this object. Not every physical component will have a serial number, or even need one. Physical entities for which the associated value of the entPhysicalIsFRU object is equal to 'false(2)' (e.g., the repeater ports within a repeater module), do not need their own unique serial number. An agent does not have to provide write access for such entities, and may return a zero-length string. If write access is implemented for an instance of entPhysicalSerialNum, and a value is written into the instance, the agent must retain the supplied value in the entPhysicalSerialNum instance associated with the same physical entity for as long as that entity remains instantiated. This includes instantiations across all re-initializations/reboots of the network management system, including those which result in a change of the physical entity's entPhysicalIndex value.	
entPhysicalMfgName	1.3.6.1.2.1.47.1.1.1.1.12	The name of the manufacturer of this physical component. The preferred value is the manufacturer name string actually printed on the component itself (if present). Note that comparisons between instances of the entPhysicalModelName, entPhysicalFirmwareRev,	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/manufacturer-name

		entPhysicalSoftwareRev, and the entPhysicalSerialNum objects, are only meaningful amongst entPhysicalEntries with the same value of entPhysicalMfgName. If the manufacturer name string associated with the physical component is unknown to the agent, then this object will contain a zero-length string.	
entPhysicalModelName	1.3.6.1.2.1.47.1.1.1.1.13	The vendor-specific model name identifier string associated with this physical component. The preferred value is the customer-visible part number, which may be printed on the component itself. If the model name string associated with the physical component is unknown to the agent, then this object will contain a zero-length string.	Cisco-IOS-XR-invmgr-oper:inventory/entities/entity/attributes basic-bag/model-name

IF-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to interface characteristics and counters.

OID Name	**OID Number**	**OID Description**	**XPATH**
ifMtu	1.3.6.1.2.1.2.2.1.4	The size of the largest packet which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/mtu

ifPhysAddress	1.3.6.1.2.1.2.2.1.6	The interface's address at its protocol sub-layer. For example, for an 802.x interface, this object normally contains a MAC address. The interface's media-specific MIB must define the bit and byte ordering and the format of the value of this object. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-type-information/bundle-information/member/mac-address
ifType	1.3.6.1.2.1.2.2.1.3	The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority (IANA), through updating the syntax of the IANAifType textual convention.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-type
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/packets-sent
ifHCOutUcastPkts	1.3.6.1.2.1.31.1.1.1.11	The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. This object is a 64-bit version of ifOutUcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/packets-sent

		other times as indicated by the value of ifCounterDiscontinuityTime.	
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/packets-received
ifHCInUcastPkts	1.3.6.1.2.1.31.1.1.1.7	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. This object is a 64-bit version of ifInUcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/packets-received
ifOutErrors	1.3.6.1.2.1.2.2.1.20	For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/output-errors

ifOutDiscards	1.3.6.1.2.1.2.2.1.19	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/output-drops
ifOutMulticastPkts	1.3.6.1.2.1.31.1.1.1.4	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/multicast-packets-sent
ifHCOutMulticastPkts	1.3.6.1.2.1.31.1.1.1.12	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifOutMulticastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/multicast-packets-sent

		ifCounterDiscontinuityTime.	
ifInMulticastPkts	1.3.6.1.2.1.31.1.1.1.2	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. For a MAC layer protocol, this includes both Group and Functional addresses. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/multicast-packets-received
ifHCInMulticastPkts	1.3.6.1.2.1.31.1.1.1.8	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifInMulticastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/multicast-packets-received
ifInErrors	1.3.6.1.2.1.2.2.1.14	For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character-oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol. Discontinuities in the value of this counter can occur at re-initialization of the	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/input-errors

		management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	
ifInDiscards	1.3.6.1.2.1.2.2.1.13	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/input-drops
ifOutOctets	1.3.6.1.2.1.2.2.1.16	The total number of octets transmitted out of the interface, including framing characters. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/bytes-sent
ifHCOutOctets	1.3.6.1.2.1.31.1.1.1.10	The total number of octets transmitted out of the interface, including framing characters. This object is a 64-bit version of ifOutOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/bytes-sent
ifInOctets	1.3.6.1.2.1.2.2.1.10	The total number of octets received on the interface, including framing characters. Discontinuities in the value of	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-

		this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	statistics/full-interface-stats/bytes-received
ifHCInOctets	1.3.6.1.2.1.31.1.1.1.6	The total number of octets received on the interface, including framing characters. This object is a 64-bit version of ifInOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/bytes-received
ifOutBroadcastPkts	1.3.6.1.2.1.31.1.1.1.5	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/broadcast-packets-sent
ifHCOutBroadcastPkts	1.3.6.1.2.1.31.1.1.1.13	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. This object is a 64-bit version of ifOutBroadcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/broadcast-packets-sent

ifInBroadcastPkts	1.3.6.1.2.1.31.1.1.1.3	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/broadcast-packets-received
ifHCInBroadcastPkts	1.3.6.1.2.1.31.1.1.1.9	The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. This object is a 64-bit version of ifInBroadcastPkts. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/interface-statistics/full-interface-stats/broadcast-packets-received
ifIndex	1.3.6.1.2.1.2.2.1.1	A unique value, greater than zero, for each interface. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/if-index
ifDescr	1.3.6.1.2.1.2.2.1.2	A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the interface hardware/software.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface/description
ifSpeed	1.3.6.1.2.1.2.2.1.5	An estimate of the interface's current bandwidth in bits per	Cisco-IOS-XR-pfi-im-cmd-

		second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. If the bandwidth of the interface is greater than the maximum value reportable by this object then this object should report its maximum value (4,294,967,295) and ifHighSpeed must be used to report the interace's speed. For a sub-layer which has no concept of bandwidth, this object should be zero.	oper:interfaces/interface-xr/interface/bandwidth
ifOperStatus	1.3.6.1.2.1.2.2.1.8	The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. If ifAdminStatus is down(2) then ifOperStatus should be down(2). If ifAdminStatus is changed to up(1) then ifOperStatus should change to up(1) if the interface is ready to transmit and receive network traffic; it should change to dormant(5) if the interface is waiting for external actions (such as a serial line waiting for an incoming connection); it should remain in the down(2) state if and only if there is a fault that prevents it from going to the up(1) state; it should remain in the notPresent(6) state if the interface has missing (typically, hardware) components.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-non-dynamics/interface-non-dynamic/oper-state
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. When a managed system initializes, all interfaces start with ifAdminStatus in the down(2)	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-non-dynamics/interface-non-dynamic/admin-state

		state. As a result of either explicit management action or per configuration information retained by the managed system, ifAdminStatus is then changed to either the up(1) or testing(3) states (or remains in the down(2) state).	
ifName	1.3.6.1.2.1.31.1.1.1.1	The textual name of the interface. The value of this object should be the name of the interface as assigned by the local device and should be suitable for use in commands entered at the device's `console'. This might be a text name, such as `le0' or a simple port number, such as `1', depending on the interface naming syntax of the device. If several entries in the ifTable together represent a single interface as named by the device, then each will have the same value of ifName. Note that for an agent which responds to SNMP queries concerning an interface on some other (proxied) device, then the value of ifName for such an interface is the proxied device's local name for it. If there is no local name, or this object is otherwise not applicable, then this object contains a zero- length string.	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-briefs/interface-brief/interface-name
ifHighSpeed	1.3.6.1.2.1.31.1.1.1.15	An estimate of the interface's current bandwidth in units of 1,000,000 bits per second. If this object reports a value of `n' then the speed of the interface is somewhere in the range of `n-500,000' to `n+499,999'. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal	Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-briefs/interface-brief/bandwidth64-bit

		bandwidth. For a sub-layer which has no concept of bandwidth, this object should be zero.	

IP-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to Internet Protocol (IP) statistics and operational values.

OID Name	OID Number	OID Description	XPATH
icmpInDestUnreachs	1.3.6.1.2.1.5.3	The number of ICMP Destination Unreachable messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInParmProbs	1.3.6.1.2.1.5.5	The number of ICMP Parameter Problem messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInSrcQuenchs	1.3.6.1.2.1.5.6	The number of ICMP Source Quench messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInEchos	1.3.6.1.2.1.5.8	The number of ICMP Echo (request) messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInEchoReps	1.3.6.1.2.1.5.9	The number of ICMP Echo Reply messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInTimestamps	1.3.6.1.2.1.5.10	The number of	Cisco-IOS-XR-ipv4-io-oper:ipv4-

		ICMP Timestamp (request) messages received.	network/nodes/node/statistics/traffic/icmp-stats
icmpInAddrMasks	1.3.6.1.2.1.5.12	The number of ICMP Address Mask Request messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpInAddrMaskReps	1.3.6.1.2.1.5.13	The number of ICMP Address Mask Reply messages received.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutMsgs	1.3.6.1.2.1.5.14	The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutDestUnreachs	1.3.6.1.2.1.5.16	The number of ICMP Destination Unreachable messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutTimeExcds	1.3.6.1.2.1.5.17	The number of ICMP Time Exceeded messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutParmProbs	1.3.6.1.2.1.5.18	The number of ICMP Parameter Problem messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats

icmpOutSrcQuenchs	1.3.6.1.2.1.5.19	The number of ICMP Source Quench messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutRedirects	1.3.6.1.2.1.5.20	The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutEchos	1.3.6.1.2.1.5.21	The number of ICMP Echo (request) messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutEchoReps	1.3.6.1.2.1.5.22	The number of ICMP Echo Reply messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutTimestamps	1.3.6.1.2.1.5.23	The number of ICMP Timestamp (request) messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutAddrMasks	1.3.6.1.2.1.5.25	The number of ICMP Address Mask Request messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
icmpOutAddrMaskReps	1.3.6.1.2.1.5.26	The number of ICMP Address Mask Reply messages sent.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/icmp-stats
ipAdEntIfIndex	1.3.6.1.2.1.4.20.1.2	The index value which uniquely identifies the interface to which this	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/

		entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of RFC 1573's ifIndex.	
ipAdEntAddr	1.3.6.1.2.1.4.20.1.1	The IP address to which this entry's addressing information pertains.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/interfaces/interface/vrfs/vrf/detail/primary-address
ipAdEntNetMask	1.3.6.1.2.1.4.20.1.3	The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/interfaces/interface/vrfs/vrf/detail/prefix-length
ipAdEntBcastAddr	1.3.6.1.2.1.4.20.1.4	The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/interfaces/interface/vrfs/vrf/detail/direct-broadcast

		ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.	
ipNetToMediaPhysAddress	1.3.6.1.2.1.4.22.1.2	The media-dependent `physical' address.	Cisco-IOS-XR-ipv4-arp-oper:arp/nodes/node/entries/entry/hardware-address

IPMIB-COMMMON

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to IP statistics.

OID Name	**OID Number**	**OID Description**	**XPATH**
ipIfStatsHCOutTransmits	1.3.6.1.2.1.4.31.3.1.31	The total number of IP datagrams that this entity supplied to the lower layers for transmission. This object counts the same datagrams as ipIfStatsOutTransmits but allows for larger values. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ipIfStatsDiscontinuityTime.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/ip stats/packets-forwarded
ipIfStatsInReceives	1.3.6.1.2.1.4.31.3.1.3	The total number of input IP datagrams received, including those received in error. Discontinuities in the value of this counter can occur at re-initialization of	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/ip stats/input-packets

		the management system, and at other times as indicated by the value of ipIfStatsDiscontinuityTime.	
ipIfStatsHCInReceives	1.3.6.1.2.1.4.31.3.1.4	The total number of input IP datagrams received, including those received in error. This object counts the same datagrams as ipIfStatsInReceives but allows for larger values. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ipIfStatsDiscontinuityTime.	Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/statistics/traffic/ip stats/input-packets

LLDP-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to Link Layer Discovery Protocol (LLDP) operational data on the monitored node.

OID Name	**OID Number**	**OID Description**	**XPATH**
lldpLocPortId	1.0.8802.1.1.2.1.3.7.1.3	The string value used to identify the port component associated with a given port in the local system.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/port-id-detail
lldpLocPortIdSubtype	1.0.8802.1.1.2.1.3.7.1.2	The type of port identifier encoding used in the associated 'lldpLocPortId' object.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/mib/port-id-sub-type
lldpLocChassisIdSubtype	1.0.8802.1.1.2.1.3.1	The type of encoding used to identify the chassis	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/mib/chassis-id-sub-type

		associated with the local system.	
lldpLocSysName	1.0.8802.1.1.2.1.3.3	The string value used to identify the system name of the local system. If the local agent supports IETF RFC 3418, lldpLocSysName object should have the same value of sysName object.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/detail/system-name
lldpRemSysName	1.0.8802.1.1.2.1.4.1.1.9	The string value used to identify the system name of the remote system.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/detail/system-name
lldpRemChassisId	1.0.8802.1.1.2.1.4.1.1.5	The string value used to identify the chassis component associated with the remote system.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor/chassis-id
lldpRemChassisIdSubtype	1.0.8802.1.1.2.1.4.1.1.4	The type of encoding used to identify the chassis associated with the remote system.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor
lldpRemPortIdSubtype	1.0.8802.1.1.2.1.4.1.1.6	The type of port identifier encoding used in the associated 'lldpRemPortId' object.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device neighbor
lldpRemPortId	1.0.8802.1.1.2.1.4.1.1.7	The string value used to identify	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/devices/device

		the port component associated with the remote system.	neighbor
lldpLocChassisId	1.0.8802.1.1.2.1.3.2	The string value used to identify the chassis component associated with the local system.	Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/details/detail/ll neighbor/chassis-id

MPLS-TE-STD-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to Multiprotocol Label Switching (MPLS) Traffic Engineering operational values on the managed device.

OID Name	**OID Number**	**OID Description**	**XPATH**
mplsTunnelName	1.3.6.1.2.1.10.166.3.2.2.1.5	The canonical name assigned to the tunnel. This name can be used to refer to the tunnel on the LSR's console port. If mplsTunnelIsIf is set to true then the ifName of the interface corresponding to this tunnel should have a value equal to mplsTunnelName. Also see the description of ifName in RFC 2863.	Cisco-IOS-XR-mpls-te-oper:mpl te/p2p-p2mp-tunnel/tunnel-heads head/tunnel-name
mplsTunnelDescr	1.3.6.1.2.1.10.166.3.2.2.1.6	A textual string containing information about the tunnel. If there is no description this object contains a zero length string. This object is may not be signaled by MPLS signaling protocols, consequentally the value of this object at transit and egress LSRs MAY be automatically generated or absent.	openconfig-network-instance:net instances/network-instance/mpls/lsps/constrained-path/tunnels/tunnel/state/descript

mplsTunnelPerfHCPackets	1.3.6.1.2.1.10.166.3.2.9.1.2	High capacity counter for number of packets forwarded by the tunnel.	openconfig-network-instance:net instances/network-instance/mpls/lsps/constrained-path/tunnels/tunnel/state/counters
mplsTunnelPerfHCBytes	1.3.6.1.2.1.10.166.3.2.9.1.5	High capacity counter for number of bytes forwarded by the tunnel.	openconfig-network-instance:net instances/network-instance/mpls/lsps/constrained-path/tunnels/tunnel/state/counters
mplsTunnelHopIpAddr	1.3.6.1.2.1.10.166.3.2.4.1.5	The Tunnel Hop Address for this tunnel hop. The type of this address is determined by the value of the corresponding mplsTunnelHopAddrType. The value of this object cannot be changed if the value of the corresponding mplsTunnelHopRowStatus object is 'active'.	Cisco-IOS-XR-mpls-te-oper:mpl te/p2p-p2mp-tunnel/tunnel-heads head/destination/destination-addr

RFC2465-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to IPv6 global values.

OID Name	OID Number	OID Description	XPATH
ipv6AddrPfxLength	1.3.6.1.2.1.55.1.8.1.2	The length of the prefix (in bits) associated with the IPv6 address of this entry.	Cisco-IOS-XR-ipv6-ma-oper:ipv6-network/nodes/node/interface-data/vrfs/vrf/briefs/brief/address/prefix-length
ipv6AddrAnycastFlag	1.3.6.1.2.1.55.1.8.1.4	This object has the value 'true(1)', if this address is an anycast address and the value 'false(2)' otherwise.	Cisco-IOS-XR-ipv6-ma-oper:ipv6-network/nodes/node/interface-data/vrfs/vrf/briefs/brief/address/is-anycast

SNMP-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to the SNMP agent itself if available.

OID Name	OID Number	OID Description	XPATH

sysUpTime	1.3.6.1.2.1.1.3	String representing the system Uptime	Cisco-IOS-XR-snmp-agent-oper:snmp/information/system-up-time/
sysObjectID	.1.3.6.1.2.1.1.2.0	String representing the system OID	Cisco-IOS-XR-snmp-agent-oper:snmp/information/system-oid/
sysDescr	1.3.6.1.2.1.1.1	String representing the system description	Cisco-IOS-XR-snmp-agent-oper:snmp/information/system-descr

TCP-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to TCP specific counters.

OID Name	**OID Number**	**OID Description**	**XPATH**
tcpInErrs	1.3.6.1.2.1.6.14	The total number of segments received in error (e.g., bad TCP checksums).	Cisco-IOS-XR-ip-tcp-oper:tcp/nodes/node/statistics/ipv4-traffic/tcp-checksum-error-packets
tcpInSegs	1.3.6.1.2.1.6.10	The total number of segments received, including those received in error. This count includes segments received on currently established connections.	Cisco-IOS-XR-ip-tcp-oper:tcp/nodes/node/statistics/ipv4-traffic/tcp-input-packets
tcpOutSegs	1.3.6.1.2.1.6.11	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.	Cisco-IOS-XR-ip-tcp-oper:tcp/nodes/node/statistics/ipv4-traffic/tcp-output-packets

UDP-MIB

The next table represents the OID name and number and the correspondent XPATH to be setup on model-driven telemetry sensor-groups related to UDP specific counters.

OID Name	**OID Number**	**OID Description**	**XPATH**
udpOutDatagrams	1.3.6.1.2.1.7.4	The total number of UDP	Cisco-IOS-XR-ip-udp-

		datagrams sent from this entity.	oper:/udp/nodes/node/statistics/ipv4-traffic/udp-output-packets Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv6-traffic/udp-output-packets
udpNoPorts	1.3.6.1.2.1.7.2	The total number of received UDP datagrams for which there was no application at the destination port.	Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv4-traffic/udp-no-ports-packets Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv6-traffic/udp-no-ports-packets
udpInErrors	1.3.6.1.2.1.7.3	The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.	Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv4-traffic/udp-checksum-error-packets Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv6-traffic/udp-checksum-error-packets
udpInDatagrams	1.3.6.1.2.1.7.1	The total number of UDP datagrams delivered to UDP users.	Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv4-traffic/udp-input-packets Cisco-IOS-XR-ip-udp-oper:/udp/nodes/node/statistics/ipv6-traffic/udp-input-packets

SNMP Traps migration

SNMP traps are messages triggered by dynamic events on the managed device. Therefore, these messages behave analogous to the concept of EDT that we covered before.

Configuration side, MDT allows the same structure for EDT, which depends on the implementation on the telemetry collector in terms of Dial-In or Dial-Out choice or capabilities.

Security considerations

SNMPv2 uses only community as an authentication/authorization mechanism. However, SNMPv3 as we covered before on SNMP section, could use credentials for authentication and AES encryption model for protecting the information.

In Telemetry approach, IOS XR allows for the usage of gRPC/TLS techniques based on certificates to perform authentication. These certificates could be used with a central point of trust (a CA server for example). After the process of building a trust relationship, all telemetry messages are sent inside a gRPC session which is encrypted with TLS accomplishing the same benefits of SNMPv3.