ASR9K Model Driven Telemetry Whitepaper

Contents

Introduction

Audience

This white paper is intended to help customers obtain a quick understanding on Model Driven Telemetry (MDT) feature in general and how it has been implemented in Aggregation Services Router 9000 (ASR9K) including some design guidelines and configuration details. It also includes some deployment consideration, which will be helpful while deploying this feature using ASR9K. Overall, this white paper can be a quick reference guide for anyone working on this feature.

Although, Telemetry is introduced as a generic feature, focus is on ASR9K implementation; i.e., not all of the features supported by other cisco platforms are supported by ASR9K platform and some feature implementation may be specific to ASR9K.

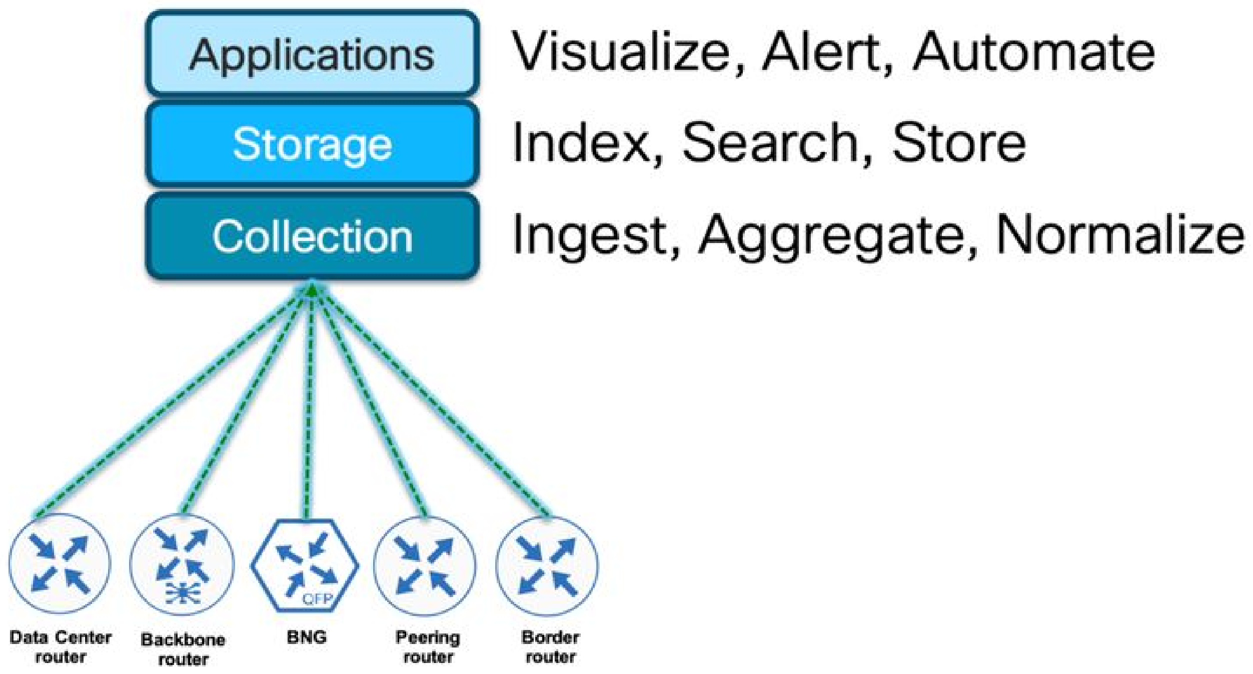
A Brief Introduction to Telemetry

To start with, in simple terms, Telemetry is the collection process of **useful operational data**. As per Wikipedia, Telemetry is an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to receiving equipment for monitoring. Telemetry word itself is derived from Greek roots: tele = remote, and metron = measure.

For the network management, Network operators have a long history of relying on Simple Network Management Protocol (SNMP). While SNMP is widely adopted for network monitoring, it was never used for configuration even though capability of configuration with snmp was always there. Operators have written automation scripts to handle day to day configuration tasks, but scripts are challenging for such tasks and difficult to manage.

Hence operators moved towards data model driven management. Network configuration is based on YANG data models pushed by protocols like netconf for example. Now just pushing the configuration doesn’t imply that configured service is running, there has to be a mechanism which can monitor services operational data at the same time as the configuration. This is where oper data models; which Telemetry uses to push information out of device; helps. Therefore, the configuration is YANG data model driven so must be the verification of service as well; as the case with Telemetry, in order to have the same object semantic. Hence the term is called **Model Driven Telemetry** or streaming Telemetry.

Model Driven Telemetry (MDT) was introduced in **cXR** (32 bit IOS XR) since release **6.1.1** and allows for collection and measurements of critical data in near real-time providing a quick answer to most of the modern network’s operational issues.



High-level Telemetry Architecture

MDT leverages structured data models supported by the networking device and provides critical data defined in those data models. **Telemetry** helps customers to **manage their multi-vendor network** using one common network management system, process and applications since the data collected from the network are standards based and are uniform across vendor implementation.

Rather than waiting for data retrieval (pull) from a centralized management station (typically SNMP NMS); with MDT, network devices pro-actively send out (**push**) performance data pertaining to the network vital functions, such as packet forwarding information, error statistics, system status, CPU and memory resources, etc.

Why Telemetry

Collecting data for analytical and troubleshooting purposes has always been an important aspect in monitoring the health of a network. There are several mechanisms available such as SNMP, CLI and Syslog to collect data from a network. While these methods served the network for a very long time but doesn't fit for modern network where demand for automation, services at scale are fundamental. Network health information, traffic statistics and critical infrastructure information are sent to a remote station in NMS, where they are used to enhance operational performance and to reduce troubleshooting time. A pull model like snmp where a client poll all the network nodes, is not efficient. Processing load on the network nodes increases when there are more no. of clients to poll. On the contrary, A push model have a capability to continuously stream data out of the network and notify the client. Telemetry enables the push model, which provides near-real-time access to monitoring data.

Streaming telemetry provides a mechanism to select data of interest from routers and to transmit it in a standard format to a remote management stations for monitoring. This mechanism enables fine tuning of the network based on real-time data, which is crucial for its seamless operation. The finer granularity and higher frequency of data available through telemetry enables better performance monitoring hence results in better troubleshooting.

It helps in a more service-efficient bandwidth utilization in the network, link utilization, risk assessment and scalability. With Streaming telemetry, more near real time data is available at the disposal of network operators which helps to improve decision-making.

The Need to move away from SNMP

SNMP has been around for three decades and the way it operates has not changed to match the monitoring needs of the modern networks. The real problem is the speed of execution of SNMP.

The three primary challenges posed by SNMP are actually part of its fundamental operational behavior and thus SNMP offers little/no room for improvement and Telemetry addresses all three inherently.

- **Speed of Execution and Need for Real Time Monitoring**

SNMP uses PULL Model - GetBulk / GetNext operations which work in a linear fashion by traversing the tables from one column to another till. In addition, multiple requests are required in case of large tables that cannot fit into one packet. This is the biggest bottleneck that causes SNMP to slow down and data being sent is often outdated by a certain time factor in minutes. This delay is simply not acceptable to modern network monitoring requirements.

MDT (Model Driven Telemetry) uses PUSH model and is inherently free from above listed limitation as it knows what data is to be sent to whom and at what interval. It only needs one lookup to gather data and uses pre-built Internal templates for ultra-fast speed of internal operations thus enabling delivery of much more data in considerably lesser time.

- **Additional Overheads and Lack of Optimization Options**

The data being pulled by SNMP is actually stored as internal data structures and needs to be converted internally by the node. This is additional work behind the scenes where network node maps internal data structures into SNMP format. There are internal optimizations performed, however they are still not enough.

On the other hand, Telemetry directly pulls out the internal data structures and performs minimal processing before it sends this data out, thus providing the most updated data with the least possible time and effort.

- **Linear Nature of Workload**

Every additional polling station will lead to additional workload on the node, even if we are polling the same exact data at same exact time. Parallel access of the same MIB from multiple polling stations can lead to slower response and higher CPU utilization. This is evident especially in the case of large tables, where multiple stations access different part of the same MIB table.

Telemetry on the other hand needs to pull data once and replicate the packets if same data is required by multiple destinations. Push model beats SNMP Pull for Speed & Scale.

With MDT, the approach to data collection radically changes and its fundamental principles are listed in the table below and compared with SNMP technology key-points.

Simple Network Management Protocol (SNMP)	**Model Driven Telemetry (MDT)**
Non Real-Time Information	Real-Time Information
Poorly scalable	Highly scalable
Pull-Model	Push-Model

Non Automated	Automation Ready/Data-Model Driven

Advantages of Streaming Telemetry

Streamed real-time telemetry data is useful in:

Capacity Planning/Traffic optimization: When bandwidth utilization and packet drops in a network are monitored frequently, it is easier to add or remove links, re-direct traffic, modify policing, and so on. With technologies like fast reroute, the network can switch to a new path and re-route faster than the SNMP poll interval mechanism. Streaming telemetry data helps in providing quick response time for faster traffic.

Better Visibility: Helps to quickly detect and avert failure situations that result after a problematic condition in the network.

Model Driven Telemetry Technical Specifications

The following section deals with technical functions and main components of IOS XR Model Driven Telemetry aka MDT.

Telemetry Functions

Telemetry framework is organized into three separate and interlinked functional blocks.

The first block is about **data representation**, which is how the information referring analysis or measurements is organized on board.

Second block is about **encoding**. Every sample interval, Telemetry translates the above measurement data into a format that can be serialized across the wire. Of course, the controller on the other end must be able to decode the data in order to have an identical copy of the original data sent by the device.

The last block is about **transport**. This is the protocol stack that is used to transfer data between devices.

The following table summarizes the main structure for Model Driven Telemetry building blocks:

Function	**Components**
Data Representation	YANG Data Models
Encoding	GPB / GPB Self-Describing
Transport	TCP/gRPC

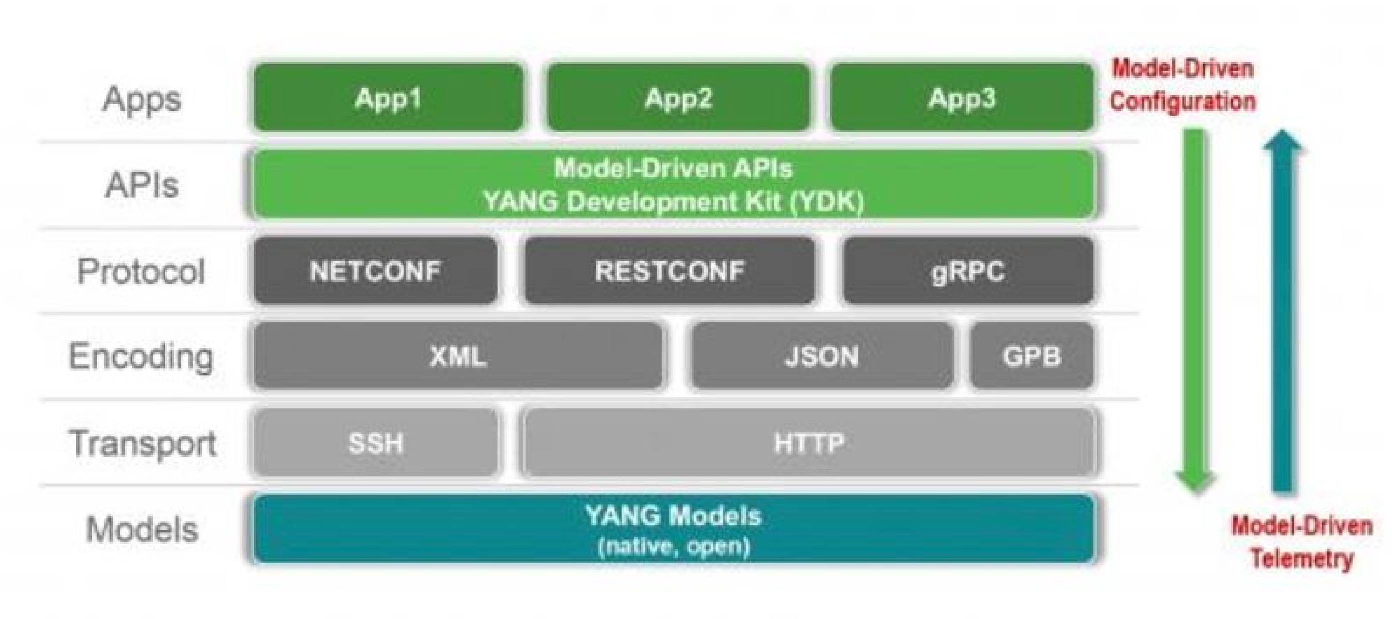
Table 3 Telemetry Building Blocks

Telemetry Components

Before understanding how Telemetry and underlying configuration pieces works, it’s important to understand the different components of Telemetry in order to evaluate an optimal setup. Telemetry relies on the IOS XR programmability stack where a new infrastructure framework provides the essential capabilities for network automation.

YANG recently became a standard for data modeling, and this is used by Cisco programmability stack to form structured dataset that can be encoded and carried as quick as possible over the network. YANG's flexibility gives the big advantage to be used also as a configuration tool for automation processes. These data models are coupled with specific encoding formats and transport protocols to make MDT a complete solution for Network Analytics.

For Model Driven Telemetry setup, the YANG data model become crucial component in order to enable the necessary data streaming for collection and analytics.



IOS XR Programmability Stack

YANG

Yang is defined as "data modeling language used to model configuration data, state data and notifications for network management protocols." Because of its decoupled nature from a typical programming language architecture, YANG can be implemented to interact with a big variety of tools.

YANG modeling data structure is built around the concept of modules & submodules which defines a hierarchy of data in a tree like fashion, that can be used for several operations including configuration actions and notification handling.

There are multiple sources of YANG Models available for use out of which below three are considered primary :

- Native Models / Cisco Specific
- OpenConfig
- IETF

Cisco-specific models: These are also called **native model** and are published by various device vendors, including Cisco. e.g. Cisco-IOS-XR-ptp-oper.yang

OpenConfig models: OpenConfig is an informal working group of network operators. OpenConfig defines common YANG models that all vendors should support to configure mission critical features. e.g. openconfig-interfaces.yang

IETF models: IETF also defines few common YANG Modules that describe basic config for interfaces, QOS, and define other common datatypes (like Ipv4, IPv6,etc.). e.g. ietf-syslog-types.yang

Cisco does support available Openconfg models. Vendors are converging to standardized way of modeling data to support a multi-vendor environment.

There are **three types** of Yang models:

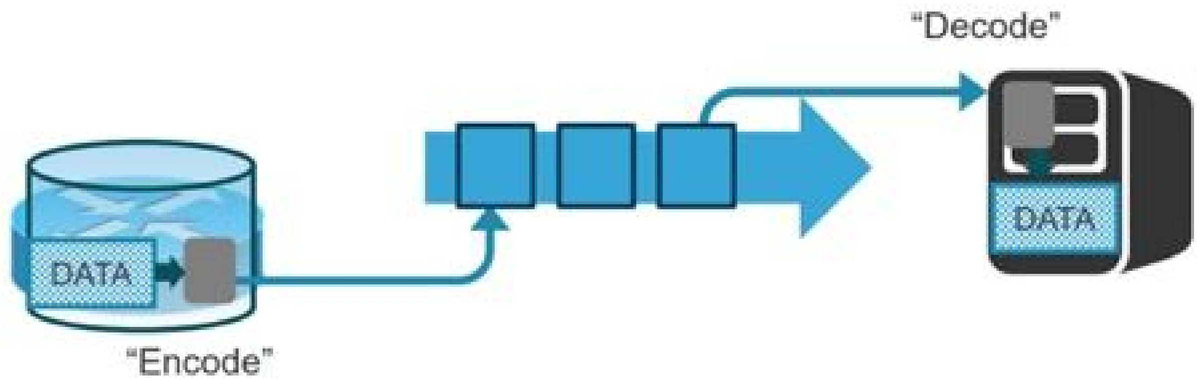
1. Operational
2. Configuration
3. Action

Telemetry only cares about **Operational Yang models** which can be identified as ***-oper-*.yang.**

YANG is defined into RFC 7950: https://tools.ietf.org/html/rfc7950.

Encoding

Encoding (or "serialisation") translates data (objects, state) into a format that can be transmitted across the network. When the receiver decodes ("de-serialises") the data, it has an semantically identical copy of the original data.

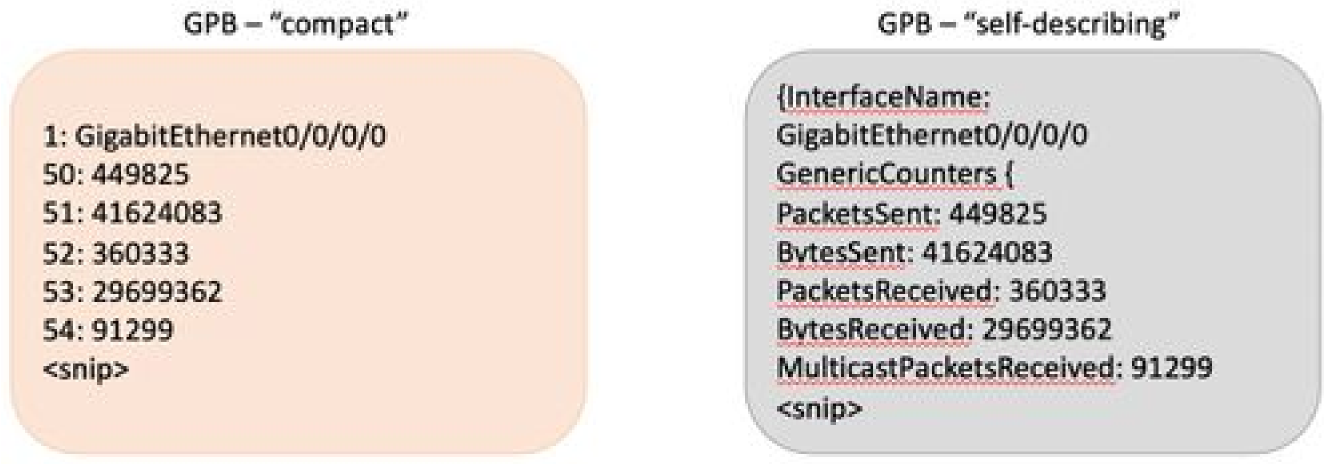


During the early development stages of telemetry, XML was initially considered as a first choice encoding format because of its tag based structure. The problem however with XML was its non-compact encoding structure. GPB (Google Protocol Buffers) was finally adopted by Cisco because it improves efficiency and speed in encoding operations.

There are two flavours of GPB as encoding options for Telemetry streaming:

1. Compact GPB
2. Self-describing GPB

The main difference between the two GPB telemetry formats is how they represent and encode the keys within a telemetry stream of data.


GPB – "compact"

```
1: GigabitEthernet0/0/0/0
50: 449825
51: 41624083
52: 360333
53: 29699362
54: 91299
<snip>
```

GPB – "self-describing"

```
{InterfaceName:
GigabitEthernet0/0/0/0
GenericCounters {
PacketsSent: 449825
BytesSent: 41624083
PacketsReceived: 360333
BytesReceived: 29699362
MulticastPacketsReceived: 91299
<snip>
```

JSON is another human friendly encoding schema available which is very easy to understand and almost any application will be able to decode.

From the deployment perspective there are few pros & cons of a encoding schema. Comparison about various encoding schema is given in the section Telemetry Design Guidelines.

Transport

Telemetry offers three possible choices for transport protocols:

- TCP
- gRPC
- UDP

Telemetry defines also two different initiation modes in order to start a session between the node and the collector:

- Dial-out
- Dial-in

The difference between the two modes consists only in how the transport session is established.

During dial-out sessions, the device initiates the connection by sending a syn packet towards the pre- configured server port. After the connection is established, data streams are immediately pushed away from the device.

For dial-in sessions, the router listens passively to a tcp port waiting for a server connection.

However, once the session is established, the router is not polled by the server itself because the device is still responsible for data pushing operations. In MDT, indeed the concept of data polling doesn't even exist.

TCP is, by default, the predefined method of transport for Telemetry because it is reliable and very easy to configure as an option.

gRPC is an modern open source framework designed to run in any environment. It is built on top of HTTP/2 and provides an enhanced and rich set of features.

Cadence-based Telemetry vs Event-based telemetry

The data from the subscribed data set is streamed out to the destination either at a configured periodic interval or only when an event occurs. This behavior is based on whether MDT is configured for cadence-based telemetry or event-based telemetry.

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

Its recommended to use Event Driven Telemetry for change-related events.

Telemetry Design Guidelines

As explained, there are many components in the telemetry stack, here are couple of guidelines to consider while implementing telemetry on XR devices.

How to Select Encoding Schema

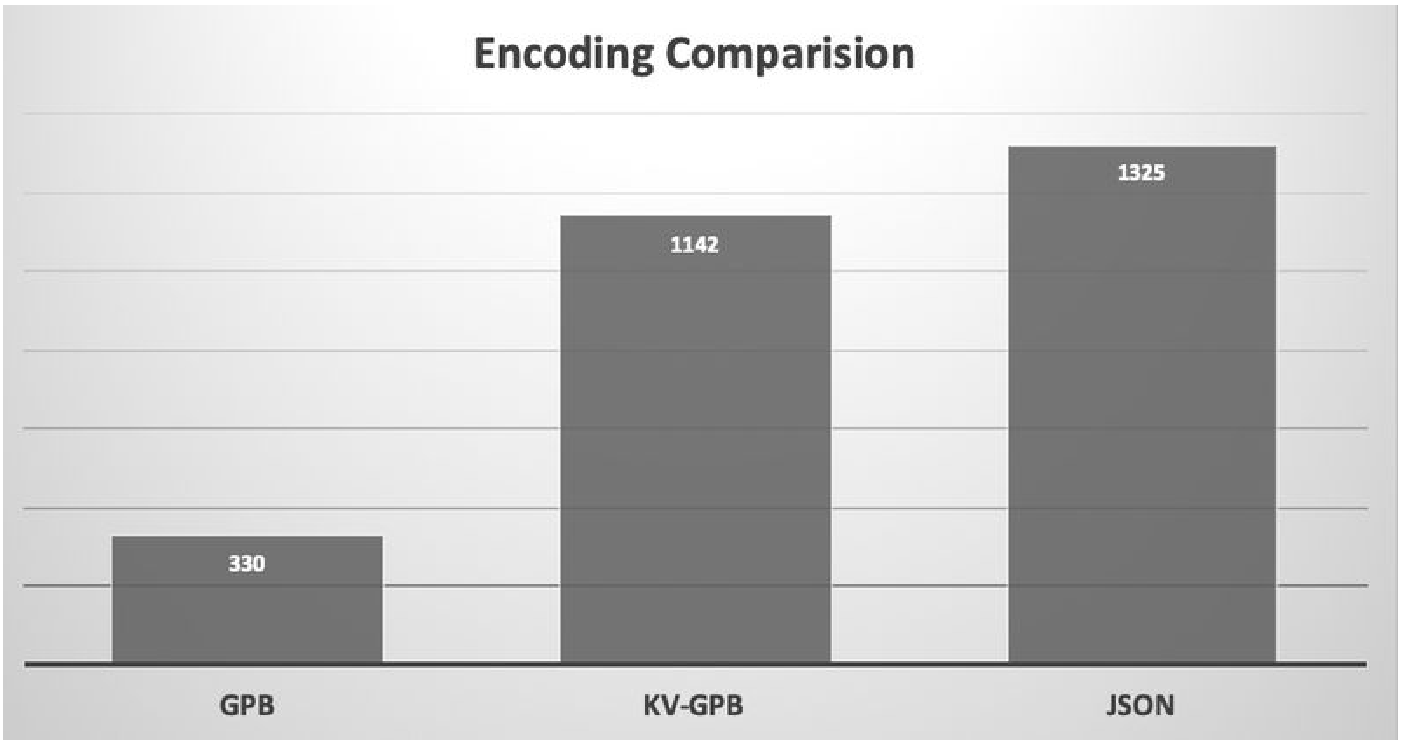
As stated, encoding or serialization translates data (objects, state) into a format that can be transmitted across the network. When the receiver decodes or de-serializes the data, it has a semantically identical copy of the original data.

Various encoding options vary in wire efficiency and ease of use.

Encoding	**Brief Description**	**Efficiency on Wire**	**Other Consideratior**
GPB (Compact)	Everything Binary (Except Values that are strings) 2X Faster, Operationally more complex (but not relative to SNMP)	High	Proto file per model
GPB - KV (Key-Value Pair)	String Keys and Binary Values (except values that are strings) 3X Larger, Native models: still need heuristics for key names	Medium to Low	Single .proto file for decocding
JSON	Everything Strings : Keys and Values	Low	Friendly. Human reac Application friendly ar easy to parse

GPB-KV gives a good & balance mid point for encoding schema.

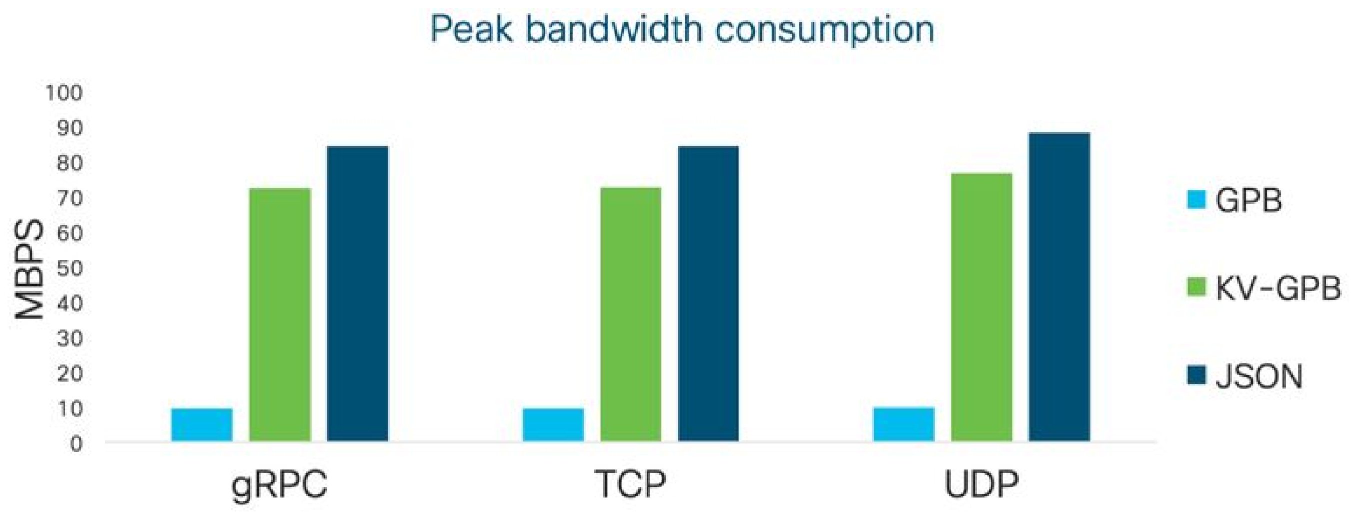
With respect to message length for chosen encoding schema, below is the comparison on the wire.



Encoding Comparison - Message Length in bytes

Transport Network Design Consideration

Different encoding option pose different bandwidth requirement. While considering Telemetry, network operator needs to take care of the sufficient bandwidth provisioning according to the chosen encoding schema. Just to have a fair idea, below bandwidth consumption per encoding schema comparison.



Network Bandwidth Comparison

Cisco recommend to use KV-GPB. It acts as a good mid-point between efficiency and convenience.

Evaluating Telemetry Configuration Options

While configuring Model-Driven Telemetry, Operator should have an understanding of all the different components that are involved in Telemetry. Based on the options that are available for transport, encoding and the direction of streaming as detailed above, one can further pick the combination that better suits an environment.

The four key components are

1. Transport
2. Encoding
3. Session Direction
4. YANG Data Models

Transport: As stated, node can deliver telemetry data either across using TCP, UDP or gRPC over HTTP/2.

While TCP is the preferred choice for simplicity, gRPC offers optional TLS capability that may be a considered as an additional benefit from security standpoint.

Encoding: Router can deliver telemetry data in two different flavors of Google Protocol Buffers: Compact and Self-Describing GPB.

Compact GPB is the most efficient encoding but requires a unique .proto for each YANG model that is streamed. Self-describing GPB is less efficient but it uses a single .proto file to decode all YANG models because the keys are passed as strings in the .proto.

Session Direction: There are two options for session initiating in telemetry deployment. The router can “dial-out” to the collector or the collector can “dial-in” to the router.

YANG is the industry accepted standard for data modeling and Cisco programmability stack also uses it to form structured dataset that can be encoded and carried as quick as possible over the network.

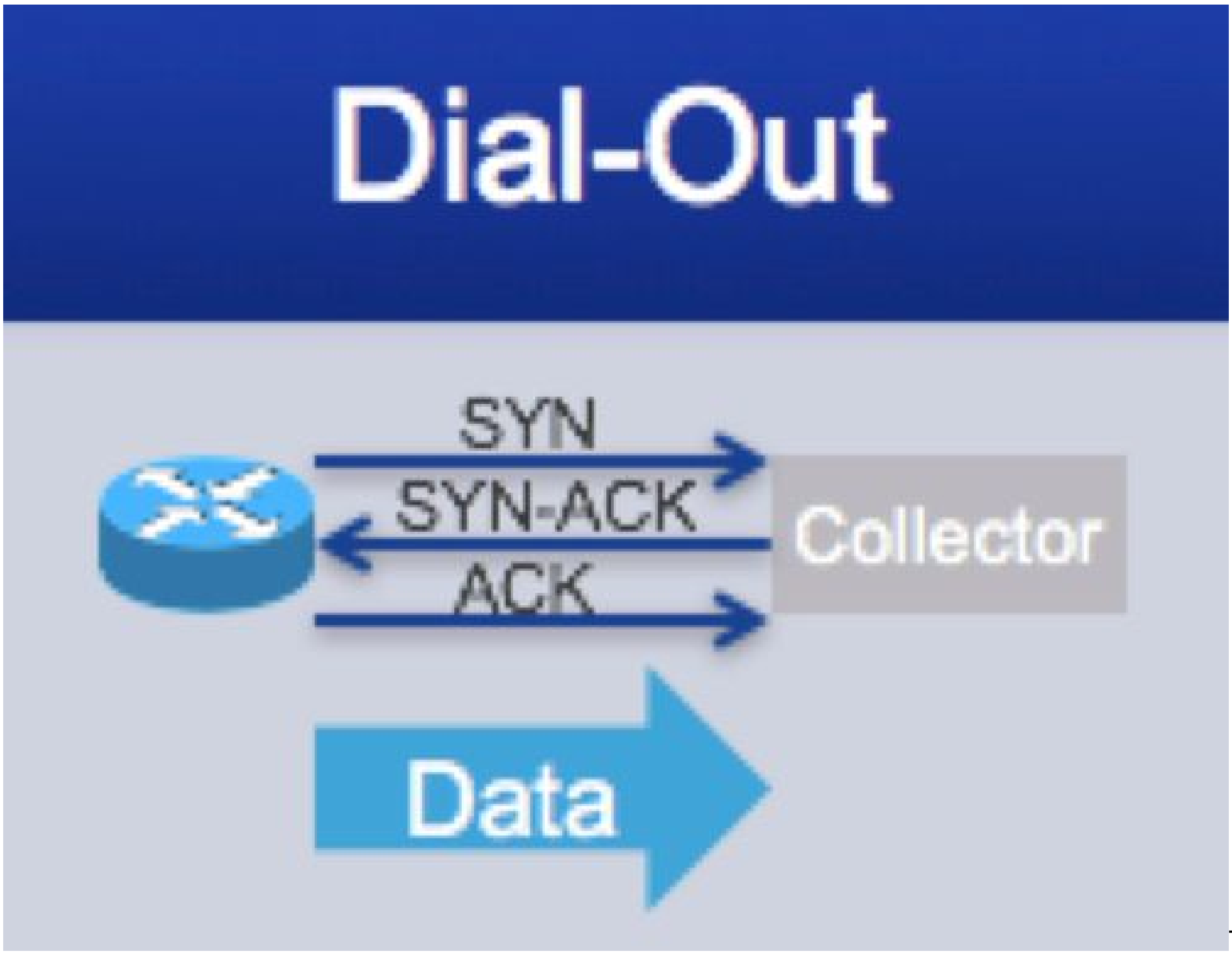
These data models coupled with specific encoding formats and transport protocols as discussed above make Model Driven Telemetry (MDT) a complete solution for Analytics.

Telemetry Configuration Examples

IOS-XR

Dial Out Configuration Break-Up

In Dial-Out mode, the router is responsible to initiate a TCP session to the collector and sends data which is specified by the sensor-group in the subscription.



Telemetry Dial-OutFrom configuration standpoint, Telemetry Configuration is a three-step process. Firstly, we identify the information that we want to stream and capture it under Sensor Group configuration. Secondly, we identify the destination to which the information has to be streamed and capture it under Destination Group configuration. Thirdly, we use the information identified in previous two steps to configure the actual subscription.

1. Define Sensor Groups
2. Define Destination Groups
3. Define Subscription

Define Sensor Groups

The Sensor group configuration identifies the information that is to be streamed. Following configuration template provides the configuration required to configure sensor groups.

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group <Sensor-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path <Sensor-Path>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

Following example shows actual example from the router CLI where a real example of Sensor

Group configuration:

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

We can have multiple Sensor Paths as a part of same SensorGroup definition:

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path sensor-path Cisco-IOS-XR-infra-
statsd-oper:infra-statistics/interfaces/interface/data-rate
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

Define Destination Groups

The Destination group configuration identifies the destination to which the information is to be streamed.

It has three key parameters

1. Session Direction
2. Encoding to be used
3. Transport Protocol to be used

Below is an example:

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)# destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#  address family ipv4 10.1.1.1 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#   encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#   protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

Define the Subscription

The Subscription binds the Sensor-Group and Destination-Group information together as the final piece of the configuration. The sample interval is defined as part of the Subscription.

Below is an example:

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

Complete Configuration Example

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 10.1.1.2 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

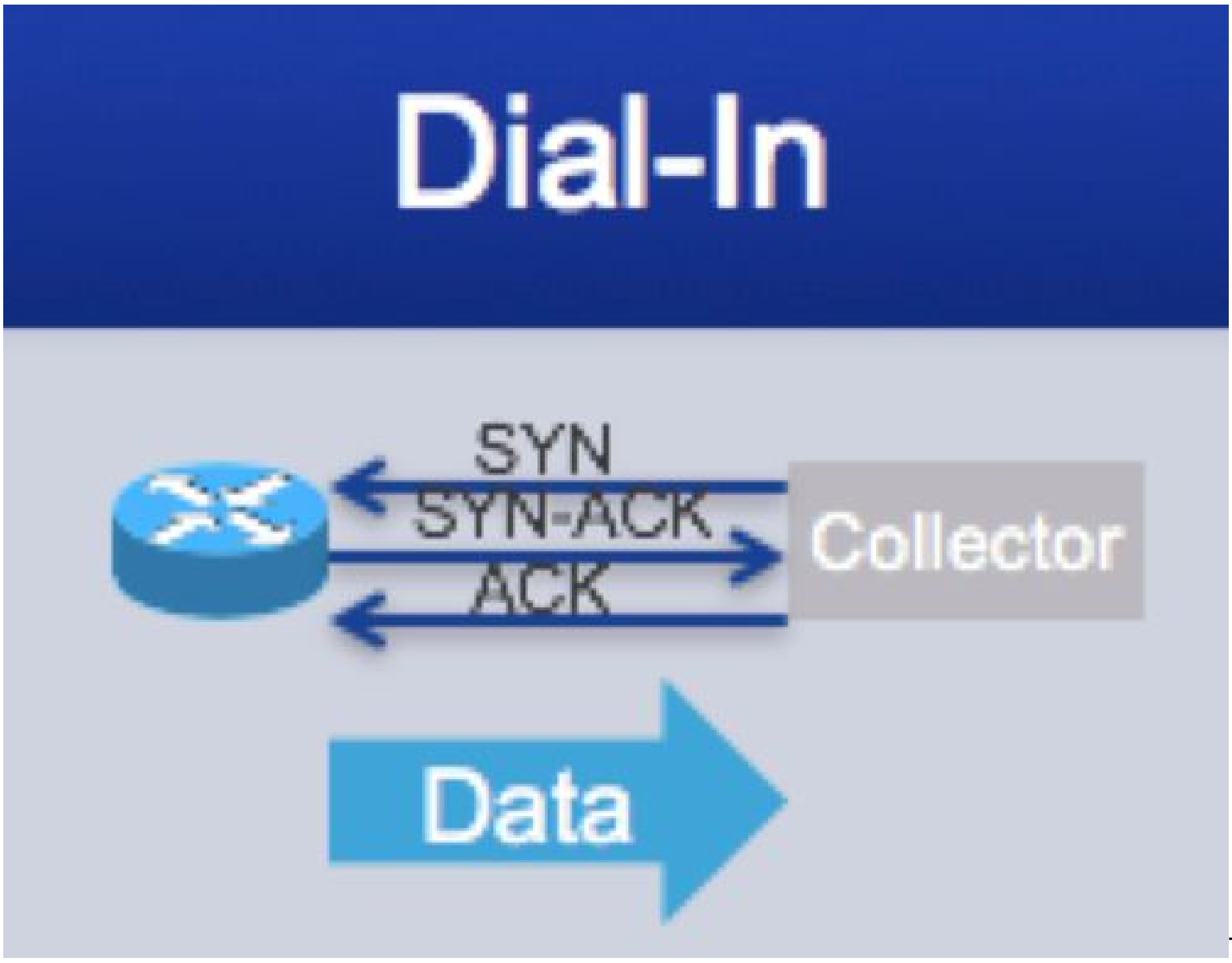
Advantages for Dial-Out

- Broader flexibility for transport options.
- No need to open ports for inbound management traffic.
- Anycast & Load-balancing.

Dial In Configuration Break-Up

In Dial-In mode, an MDT collector / receiver / orchestrator dials in to the router and subscribes dynamically to one or more sensor paths or subscriptions. The router acts as the server and the client as the receiver.

Only a single session is formed and the router streams telemetry data through the same session. This dynamic subscription terminates when the receiver cancels the subscription or when the session terminates.



Telemetry Dial-In

Since the collector “dials-in” to the router, there’s no need to specify each MDT destination in the configuration. Just enable the gRPC service on the router, connect your client, and dynamically enable the telemetry subscription you want.

From configuration standpoint, Telemetry Configuration is a three-step process similar to the one as described above. Firstly, we need to enable gRPC. Secondly, we identify the destination to which the information has to be streamed and capture it under Sesnor Group configuration. Thirdly, we use the information identified in previous two steps to configure the actual subscription.

1. Enable gRPC
2. Define Sensor Groups
3. Define Subscription

Click to Expand

Dial-In mode is only supported with gRPC

Dial-In mode is only supported with gRPC

Enable gRPC

Firstly, we need to enable gRPC server on the router to accept incoming connections from the collector.

Click to Expand

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

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

```
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)#commit
RP/0/RP0/CPU0:XR(config-grpc)#end
RP/0/RP0/CPU0:XR#
```

Define Sensor Groups

The Sensor group configuration identifies the information that is to be streamed. Following configuration template provides the configuration required to configure sensor groups.

Following example shows actual example from the router CLI where a real example of Sensor Group configuration

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path openconfig-
interfaces:interfaces/interface
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# commit
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# end
RP/0/RP0/CPU0:XR#
```

Define the Subscription

The Subscription binds the Sensor-Group and gRPC together as the final piece of the configuration. The sample interval is defined as part of the Subscription.

Following configuration template provides the configuration required to configure the subscription.

The following example shows actual example from the router CLI where we are creating a Subscription and binding the Sensor-Group and Destination-Group together and also defining the sample-rate.

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

Complete Configuration Template and Example

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 30000
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

Advantages for Dial-In

- A single channel for Configuration and Streaming
- Listening port on the router/device
- Transient Connection
- Currently only gRPC/gNMI available

Event Driven Telemetry

In Event driven Telemetry, the data from the subscribed data set is streamed out only when an event occurs.

Event Driven Telemetry Configuration

The configuration for event-based telemetry is similar to cadence-based telemetry with the only difference in configuration of Event Driven Telemetry is that of the sample interval. Configuring the sample interval value to zero sets the subscription for event-based telemetry.

Complete Configuration Template and Example for DIAL-OUT

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group <Sensor-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path <Sensor-Path>
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group <Destination-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 <Destination-IP>  port
<Destination-Port>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding <Encoding-Type>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol <Transport-Protocol>
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription <Subscription-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id <Sensor-Group-Name> sample-interval
<0>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id <Destination-Group-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

Following example shows actual example from the router CLI.

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

```
RP/0/RP0/CPU0:XR#conf
RP/0/RP0/CPU0:XR(config)#
RP/0/RP0/CPU0:XR(config)#telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#sensor-group SensorGroup101
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)# sensor-path Cisco-IOS-XR-infra-statsd-
oper:infra-statistics/interfaces/interface/latest/generic-counters
RP/0/RP0/CPU0:XR(config-model-driven-snsr-grp)#destination-group DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-dest)#address family ipv4 10.1.1.2 port 5432
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#encoding self-describing-gpb
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#protocol tcp
RP/0/RP0/CPU0:XR(config-model-driven-dest-addr)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 0
RP/0/RP0/CPU0:XR(config-model-driven-subs)#destination-id DestGroup101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

Complete Configuration Template and Example for DIAL-IN

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port <port-number>
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription <Subscription-Name>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id <Sensor-Group-Name> sample-interval
<0>
RP/0/RP0/CPU0:XR(config-model-driven-subs)#commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)#end
RP/0/RP0/CPU0:XR#
```

Following example shows actual example from the router CLI.

```
RP/0/RP0/CPU0:XR#
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)# grpc
RP/0/RP0/CPU0:XR(config-grpc)#port 57890
RP/0/RP0/CPU0:XR(config-grpc)telemetry model-driven
RP/0/RP0/CPU0:XR(config-model-driven)#subscription Subscription101
RP/0/RP0/CPU0:XR(config-model-driven-subs)#sensor-group-id SensorGroup101 sample-interval 0
RP/0/RP0/CPU0:XR(config-model-driven-subs)# commit
RP/0/RP0/CPU0:XR(config-model-driven-subs)# end
RP/0/RP0/CPU0:XR#
```

Validating Telemetry with SHOW Commands

From router standpoint, we can verify the parameters configured for each Sensor Group, Destination Group and Subscription

```
// ALL CONFIGURED SUBSCRIPTIONS

RP/0/RP0/CPU0:XR#show telemetry model-driven subscription

Subscription:  Subscription101          State: ACTIVE
-------------
```

```
  Sensor groups:
  Id                 Interval(ms)        State
  SensorGroup101     30000               **Resolved**

  Destination Groups:
  Id                 Encoding            Transport   State   Port    IP
  DestGroup101       self-describing-gpb tcp         Active  5432    172.16.128.3
```

// DETAILS ON A PARTICULAR SUBSCRIPTION

RP/0/RP0/CPU0:XR#show telemetry model-driven subscription Subscription101

```
Subscription:  Subscription101
-------------
  State:       ACTIVE
  Sensor groups:
  Id: SensorGroup101
    Sample Interval:      30000 ms
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:    Resolved

  Destination Groups:
  Group Id: DestGroup101
    Destination IP:       172.16.128.3
    Destination Port:     5432
    Encoding:             self-describing-gpb
    Transport:            tcp
    State:                Active
    Total bytes sent:     4893
    Total packets sent:   1
    Last Sent time:       2019-11-01 10:04:11.2378949664 +0000

    Collection Groups:
  ------------------
    Id: 1
    Sample Interval:      30000 ms
    Encoding:             self-describing-gpb
    Num of collection:    5
    Collection time:      Min:     6 ms Max:    29 ms
    Total time:           Min:     6 ms Avg:    12 ms Max:    29 ms
    Total Deferred:       0
    Total Send Errors:    0
    Total Send Drops:     0
    Total Other Errors:   0
    Last Collection Start:2019-11-01 10:06:11.2499000664 +0000
    Last Collection End:  2019-11-01 10:06:11.2499006664 +0000
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters

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

// ALL CONFIGURED DESTINATIONS

RP/0/RP0/CPU0:XR#show telemetry model-driven destination

```
  Group Id         IP               Port    Encoding            Transport   State
  -----------------------------------------------------------------------------
  DestGroup101     172.16.128.3     5432    self-describing-gpb tcp         Active
RP/0/RP0/CPU0:XR#
```

// PARTICULAR DESTINATION

```
RP/0/RP0/CPU0:XR#show telemetry model-driven destination DestGroup101
  Destination Group:  DestGroup101
  -----------------
    Destination IP:       172.16.128.3
    Destination Port:     5432
    State:                Active
    Encoding:             self-describing-gpb
    Transport:            tcp
    Total bytes sent:     83181
    Total packets sent:   17
    Last Sent time:       2019-11-01 10:12:11.2859133664 +0000

    Collection Groups:
    ------------------
      Id: 1
      Sample Interval:      30000 ms
    Encoding:             self-describing-gpb
      Num of collection:    17
      Collection time:      Min:     5 ms Max:    29 ms
      Total time:           Min:     6 ms Max:    29 ms Avg:    10 ms
      Total Deferred:       0
      Total Send Errors:    0
      Total Send Drops:     0
      Total Other Errors:   0
      Last Collection Start:2019-11-01 10:12:11.2859128664 +0000
      Last Collection End:  2019-11-01 10:12:11.2859134664 +0000
      Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters

RP/0/RP0/CPU0:XR#

// ALL CONFIGURED SENSOR GROUPS

RP/0/RP0/CPU0:XR#show telemetry model-driven sensor-group
  Sensor Group Id:SensorGroup101
    Sensor Path:        Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:  Resolved

// PARTICULAR SENSOR GROUPS

RP/0/RP0/CPU0:XR#show telemetry model-driven sensor-group SensorGroup101
  Sensor Group Id:SensorGroup101
    Sensor Path:        Cisco-IOS-XR-infra-statsd-oper:infra-
statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:  Resolved

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

Telemetry Collection stack

Beside the router configuration, A Telemetry based solution required several components like collector, database and Monitoring/Analytics software. These components can either be configured separately or can be part of a single comprehensive product.

- A decoder should be installed to take inbound packets and pass them on for further storage.
- A Time series database (aka TSDB) is required in order to store streamed information.
- A graphical tool is also needed to visualize data taken from the internal database.

It’s beyond the scope to describe the collection stack in detail. Cisco Crossworks Health Insights allows zero-touch telemetry where devices are automatically provisioned with telemetry configuration and tables/schema are created in a Time Series Database (TSDB). It streamlines the operational and network management overhead of collecting and cleansing data, thereby allowing operators to focus on their business goals. The utilisation of a common collector to collect network device data over SNMP, CLI, and model-driven telemetry allows to avoid data duplication and also reduces load on devices and the network.

Deployment Considerations for Telemetry in a Network

Following are various consideration to be made while analyzing deployment of Telemetry in a Network.

Scaling

Telemetry can stream considerable amount of data and careful consideration of the scalability aspects is recommended.

Stream only the required data

Each Yang model will have multiple leaf nodes. It’s advised to be specific about the information that is required and the information that is not required. It is recommended to explore the Yang Models and identify the data path required by the telemetry use-cases.

Consider the Amount of Streaming Data

The total amount of telemetry data being streamed will need consideration about following points:

1. Bandwidth allocation within the network
2. QoS
3. Performance of additional application/software like Collector softwareTime Series databaseAnalytics / Visualization software
4. Encoding efficiency (discussed in Section "Telemetry Design Guidelines") directly impacts the amount of data being streamed. Compact GPB is recommended wherever feasible.
5. Collection interval directly impact multiple aspects including but not limited to following. Bandwidth utilization in the networkStorage requirement on the DatabasePerformance of the device streaming the data

It is recommended to evaluate the frequency of the collection based on the application requirements.

Overall, it is recommended to consider filtering unwanted data at source or destination as considered feasible. We do have the option to filter undesired data. Filtering can be performed on two levels: -

1. At the Source – The devices streaming the data.
2. At the Destination – The Collector gathering and Normalizing data. (Filtering at collector is beyond the scope of this document)

Following example show data being filtered only for Hundered Gig interfaces within the sensor paths by applying wildcards.

```
sensor-path Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[interface-
name='HundredGigE*']/latest/generic-counters
```

References

https://blogs.cisco.com/sp/the-limits-of-snmp

https://blogs.cisco.com/sp/why-you-should-care-about-model-driven-telemetry

https://www.cisco.com/c/en/us/td/docs/iosxr/asr9000/telemetry/b-telemetry-cg-asr9000-61x.html