gNMI Overview

NETCONF WG, IETF 101
robjs@google.com, on behalf of the gNMI team
Motivation

➔ Inform IETF of open source development and implementations. Particularly as an alternative to NETCONF/RESTCONF

➔ Feed development and deployment experience back to IETF
Lessons learnt from production deployments

➔ Invite interested parties to contribute to development
Protocol specification and reference implementation are open source

➔ Not asking for adoption within NETCONF WG/IETF
What is gNMI?

➔ Protocol for configuration manipulation and state retrieval.
   ◆ Data handled by gNMI must be able to be described using a path consisting of element names and map<string,string> attributes.
   ◆ No requirement for this to be YANG-modelled.

➔ Built on top of gRPC - an open source framework developed by Google and managed by CNCF.
   ◆ RPC framework built on top of HTTP/2
   ◆ Unary, server streaming, client streaming and bi-directional streaming RPCs
   ◆ Multiplexing of RPCs over a single channel provided by library

➔ Protobuf service definition, and encoding for payload.
gNMI RPCs (I)

➔ **Set**
- Manipulate the (writeable) state of a target.
- Simplified transaction model - each unary Set RPC is a transaction.
- No requirements for long-lived candidates - push staging of modification to client.

➔ **Subscribe**
- Streaming RPC for target to send state to client.
- Immutable subscriptions with an overall mode:
  - STREAM - “streaming telemetry” - long-lived push from device.
  - POLL - client-requested streaming.
  - ONCE - target advertises entire dataset and closes RPC.
- STREAM data can be SAMPLE, ON_CHANGE or a mix (target defined) - cadence-based sampling, and event-driven updates.
- Critically for data fidelity, state is **always time stamped at target**
gNMI RPCs (II)

➔ **Get**
- Snapshot of path state at a particular time.
- Typical use case is for configuration state retrieval.
- Scaling implications of serialising large object for target.

➔ **Capabilities**
- Used to understand encodings and models that are supported by a target.
Extending gNMI

➔ Collaborative approach for extensions - GitHub issue discussion.
  ◆ Aiming to keep core specification confined to the common set of cases

➔ Extensions can be carried per message.
  ◆ Can be used to extend protocol - e.g., proxying, master arbitration for writers.
  ◆ Well known extensions where address multiple use cases.
  ◆ Registered extensions (assigned ID, and opaque contents) for arbitrary extension.

➔ Intended only where expanding on existing RPC function.
  ◆ New RPCs can be defined in an extension service - multiple services can run per device.
Lessons learnt through gNMI development.

➔ **Timestamping is critical.**
- Improves fidelity of telemetry - especially useful where devices implement caching.
- gNMI’s use of <path, value> in telemetry ensures this is simple to include.

➔ **Encoding of values is best done using native types.**
- Support JSON-encoding, but using 7951 encoding means that telemetry variables that are 64-bit integers become strings - not ideal in the collector.
- Adopted native protobuf encodings, with a mapping from the schema types if required.

➔ **Overall on-the-wire efficiency must be considered:**
- Significant volume of data on scaled systems (QoS, Interfaces) or large data sets (BGP RIB, device RIB)
- Prefixing approach allows significant data reduction.
- Use of protobuf structure for aggregated datasets allows for binary encoding
Development Approach for gNMI

➔ **Specification**
- Essentially companion document for the protobuf service definition.

➔ **Reference tool implementations:**
- `gnmi_cli` - tool for interacting with gNMI implementations.
- Fake target for use in testing.
- Telemetry collector implementation *mostly* open source.
- Reference server implementation being published.

➔ **In the future - compliance test suite.**
- Requires some knowledge of the underlying data tree supported, so will be use-case specific.
Resources

➔ github.com/openconfig/gnmi - reference collector code, and protos.

➔ github.com/openconfig/reference - protocol specification.

➔ github.com/google/gnxi - reference implementation for target, and additional tooling.