gRPC Network Management Interface

`draft-openconfig-rtgwg-gnmi-spec-00`

Rob Shakir, Anees Shaikh, Paul Borman, Marcus Hines,
Carl Lebsack, Chris Morrow (Google)

IETF 98
RTGWG

What is gNMI ?

specification of RPCs and behaviors for managing state on a network device

supports state retrieval (via streaming telemetry or snapshots) and state modification (configuration)

built on the open source gRPC framework (gRPC ⊂ gNMI)

- gNMI defines a gRPC service using protobuf IDL

designed to carry any tree-structured data (not limited to YANG-modeled data)

- addressable via paths
- has well-defined serialization

Why gNMI ?

provides a single service for state management (streaming telemetry and configuration)

built on a modern standard, secure transport and open RPC framework with many language bindings

supports very efficient serialization and data access

- 3x-10x smaller than XML

offers an implemented alternative to NETCONF, RESTCONF, …

- early-release implementations on multiple router and transport platforms
- reference tools published by OpenConfig

Disclaimers

`draft-openconfig-rtgwg-gnmi-spec` is an *informational* draft

- normative reference is published in github
- share operational requirements and design considerations with community
- provide awareness of related work outside IETF

is gNMI now the ‘OpenConfig standard’ ?

- no
- OpenConfig operators use, or plan to use, various RPC frameworks including gNMI/gRPC, NETCONF, RESTCONF, ...

The gNMI service

```
option (gnmi_service) = "0.2.0";
service gNMI {
 // Retrieve the set of capabilities supported by the target.
 rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);
 // Retrieve a snapshot of data from the target.
 rpc Get(GetRequest) returns (GetResponse);
 // Modify the state of data on the target.
 rpc Set(SetRequest) returns (SetResponse);
 // Subscribe to a stream of values of particular paths within the data
tree.
 rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

Some basic message types

message **Update**

```
message Path {
  // An element of the path.
  repeated string element = 1;
  // Label to disambiguate the path.
  string origin = 2;
}
```

paths encoded as an array of path components

gNMI paths use a simplified variant of XPATH syntax

```
message Value {
  bytes value = 1;
  Encoding type = 2;
}
```

multiple supported encodings, incl. JSON, JSON_IETF, PROTO, ASCII, BYTES

```
message Error {
  // Canonical gRPC error code.
  uint32 code = 1;
  // Human readable error.
  string message = 2;
  // Optional additional information.
  google.protobuf.Any data = 3;
}
```

reuse gRPC canonical errors -- spec maps behaviors onto these error codes

Capabilities RPC

```
message CapabilityResponse {
  repeated ModelData supported_models = 1;
  repeated Encoding supported_encodings = 2;
  string gNMI_version = 3;
}
```

interrogate device to learn which models and data encodings are supported

```
message ModelData {
  string name = 1;
  string organization = 2;
  string version = 3;
}
```

model data intended to reference entries in a YANG catalog

e.g., `draft-openconfig-netmod-model-catalog`

Set RPC

```
message SetRequest {
  Path prefix = 1;
  repeated Path delete = 2;
  repeated Update replace = 3;
  repeated Update update = 4;
}
```

requests in a Set RPC are considered part of a single transaction

```
message SetResponse {
  Path prefix = 1;
  repeated UpdateResult response = 2;
  Error message = 3;
}
```

response includes results for each element of the request

top-level error message to indicate overall success / failure

Subscribe RPC (streaming)

```
message SubscribeRequest {
  oneof request {
    SubscriptionList subscribe = 1;
    ...
  }
}
message Subscription {
  Path path = 1;
  SubscriptionMode mode = 2;
  uint64 sample_interval = 3;
  bool suppress_redundant = 4;
  uint64 heartbeat_interval = 5;
}
message SubscribeResponse {
  oneof response {
    Notification update = 1;
    bool sync_response = 3;
    Error error = 4;
  }
}
```

subscriptions primarily consist of a path and a mode

- modes: SAMPLE, ON_CHANGE, TARGET_DEFINED

subscribe RPC supports streaming, polling, and get-once operation

targets send streaming notifications (update or delete values)

notification includes the path and a timestamp

Ongoing / upcoming work on gNMI

current gNMI definition supports only NMS-initiated connections to target devices

- extend to “dial-out” to support target-initiated connections

new services for operational commands

- e.g. ping, traceroute, reboot, clear BGP session, update firmware, …
- considering as a set of microservices , separate from main gNMI service

native Protobuf value encoding

- avoid type-casting to strings during encoding

Additional material

gRPC : an open, multi-platform RPC framework

gRPC is a open source version of Google’s microservice communication framework

gRPC leverages standard HTTP/2 as its transport layer

- binary framing, header compression
- bidirectional streams, server push support
- connection multiplexing across requests and streams

gRPC features

- load-balancing, app-level flow control, call-cancellation
- serialization with protobuf (efficient wire encoding)
- multi-platform, many supported languages
- open source, under active development

@grpcio

www.grpc.io

see `draft-kumar-rtgwg-grpc-protocol-00` for protocol details

Streaming telemetry and gRPC

Streaming telemetry benefits over SNMP

- devices stream data based on a specified frequency or upon state change
- data is sent as soon as it is available, reducing the need to buffer
- no single large request for all data (unlike SNMP polling)
- data sent incrementally, e.g., only for those data items that have changed
- ability to distribute the telemetry sources (e.g., directly to linecards)
- users issue subscription requests via RPC for data of interest
- data exported in a well-structured, common format, e.g., based on YANG models
- device and collector communicate over a secure, authenticated, reliable channel