Model Driven Telemetry APIs Overview

Patrice Nivaggioli
Cisco
June 2021

API and Remote Calls

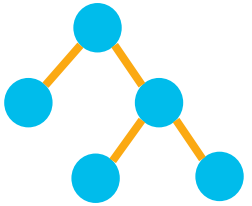
- An API is simply a specification of remote calls exposed to the API consumers
- Remote Calls
  - Define the data to be transmitted
  - Determine how the data is serialized over the wire
  - Choose a transport protocol

1. Data Models

Data Models: define the data to be transmitted

YANG Example

- Data modeling language
- Describes data hierarchy
  - config and operational data as a nodes tree structure
- Specifies nodes constraints, data types, etc.



YANG models: Interfaces modeling examples

IETF

```
module ietf-interfaces {
  revision 2018-02-20 {
    "RFC 8343: ...";
  }
  container interfaces {
    list interface {
      leaf name {...}
      leaf type {...}
      leaf enabled {...}
      leaf admin-status {...}
      leaf oper-status {...}
  container statistics {
      leaf in-octets {...}
      leaf out-octets {...}
  ...
```

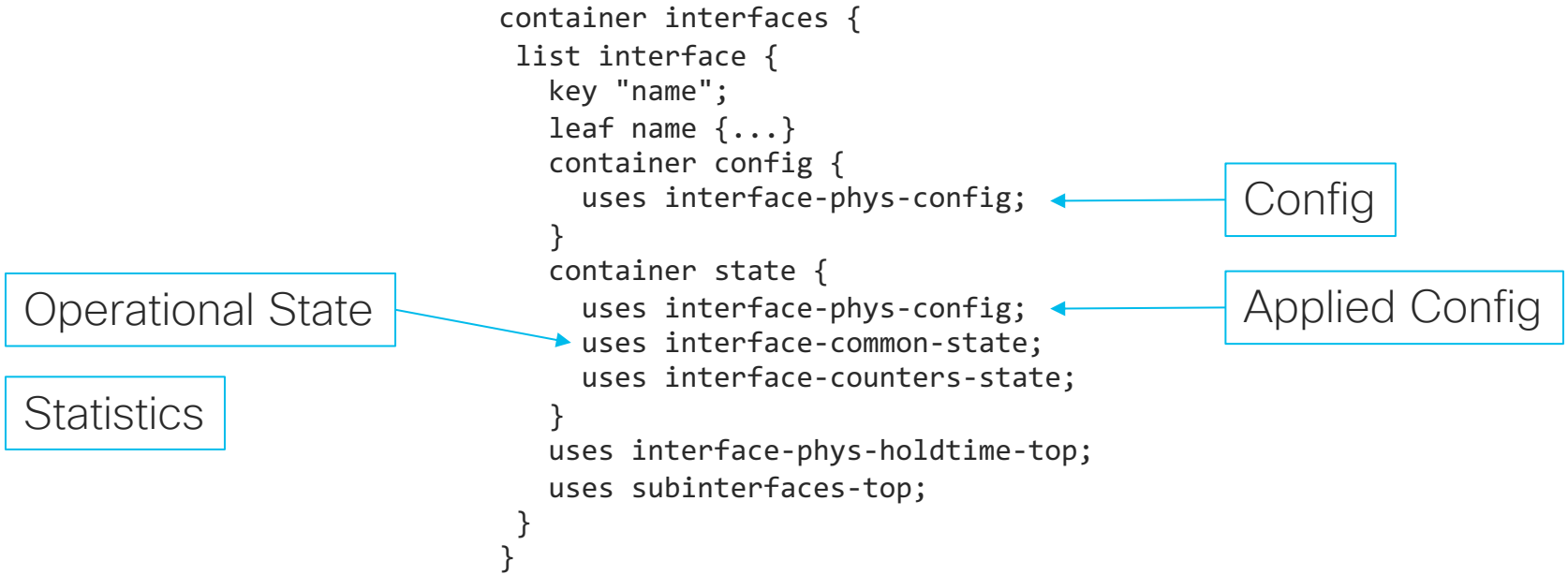
OpenConfig

```
module openconfig-interfaces {
  revision "2018-04-24" {
    reference "2.3.1";
  }
  container interfaces {
    list interface {
      leaf name {...}
      container config {
        leaf type {...}
        leaf enabled {...}
      container state {
        leaf admin-status {...}
        leaf oper-status {...}
      container counters {
        leaf in-octets {...}
        leaf out-octets {...}
  container subinterfaces {
     list subinterface {
        leaf index {...}
        ...
```

Cisco Native

```
module Cisco-IOS-XR-ifmgr-cfg {
  revision 2017-09-07 {
    ...
  }
  container interface-configurations {
    list interface-configuration {
      leaf interface-name {...}
      leaf active {...}
      leaf shutdown { type empty; }
      ...
module Cisco-IOS-XR-infra-statsd-oper {
  container infra-statistics {
    container interfaces {
      list interface {
        container latest {
          container generic-counters {
            leaf bytes-received {...}
            leaf bytes-sent {...}
...
```

OpenConfig Interfaces



OpenConfig

- Vendor neutral, driven by network operators
- Combines config and operational data (intended vs derived state)
  - Config
  - Statistics (e.g., counters)
  - Operational State (e.g., BGP session status)
  - Applied config (...is part of the state)
- Model consistency and semantic versioning

2. Encoding

Encoding and Data Format

- Determine how the data is serialized over the wire

JSON

```
{
 "person": {
   "name": "John Doe",
   "email": "jdoe@example.com"
 }
}
```

- Human readable/editable
- Can be parsed without knowing the schema in advance

XML

```
<person>
 <name>John Doe</name>
 <email>jdoe@example.com</email>
</person>
```

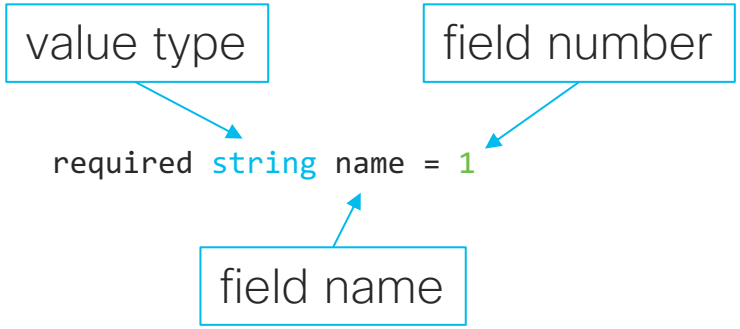
PROTOBUF

```
1 {
  1: "John Doe"
  2: "jdoe@example.com"
}
```

- Very dense (small output)
- Very fast processing
- Not human readable
- Need message definition to be meaningful

Protocol Buffers Message definition: .proto file

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }
  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }
  repeated PhoneNumber phone = 4
}
```



- Protocol Buffers requires a decoder ring
- The protocol definition file (.proto) defines the messages

Protocol Buffers Message Telemetry definition

Common to GPB and KV

```
syntax = "proto3";
option go_package = "telemetry_bis";

message Telemetry {
  oneof node_id {
      string node_id_str = 1;
  }
  oneof subscription {
      string subscription_id_str = 3;
  }
  string encoding_path = 6;
  uint64 collection_id = 8;
  uint64 collection_start_time = 9;
  uint64 msg_timestamp = 10;
  repeated TelemetryField data_gpbkv = 11;
  TelemetryGPBTable data_gpb = 12;
  uint64 collection_end_time = 13;
}
```

GPB-KV
Self-describing

GPB
Compact

GPB KV vs Compact

```
/* KV GPB specific payload definition */

message TelemetryField {
  uint64 timestamp = 1;
  string name = 2;
  oneof value_by_type {
    bytes bytes_value = 4;
    string string_value = 5;
    bool bool_value = 6;
    uint32 uint32_value = 7;
    uint64 uint64_value = 8;
    sint32 sint32_value = 9;
    sint64 sint64_value = 10;
    double double_value = 11;
    float float_value = 12;
  }
  repeated TelemetryField fields = 15;
}
```

String keys

```
/*(Compact)GPB specific payload definition */

message TelemetryGPBTable {
  repeated TelemetryRowGPB row = 1;
}

message TelemetryRowGPB {
  uint64 timestamp = 1;
  bytes keys = 10;
  bytes content = 11;
}
```

Binary keys

GPB KV example

```
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" }
  }
  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 Compact example

```
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\003\000\320\003\000\330\003\t\340\003\00
      0\350\003\000\360\003\377\001"
  }
```

```
  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\003\000\320\003\300\204=\330\003\000\34
      0\003\000\350\003\000\360\003\377\001"
  }
...
collection_end_time: 1485794640480
```

3. Transport

Transport

NETCONF	RESTCONF	gRPC
• SSH	• HTTP	• HTTP/2

• RPC
  • <GET-CONFIG>
  • <EDIT-CONFIG>
  • <COMMIT>
  • <LOCK>
  • ...
	• METHODS
	  • GET
	  • POST
	  • DELETE
	  • PUT
	  • ...
		• RPC
		  • Unary
		  • Server Streaming
		  • Client Streaming
		  • Bidirectional Streaming

HTTP/2 and gRPC

HTTP/2

- Binary, easier framing
- Header compression
- Request and response multiplexing over a single TCP connection
- Bidirectional streams

gRPC

- Strongly typed service and message definition
- Takes care of all the underlying plumbing
- Runs over HTTP/2
- Cloud Native Computing Foundation Project

gRPC Service Interface Definitions

gNMI

```
service gNMI {
  rpc Capabilities(CapabilityRequest)
returns (CapabilityResponse);
  rpc Get(GetRequest)
returns (GetResponse);
  rpc Set(SetRequest)
returns (SetResponse);
  rpc Subscribe(stream SubscribeRequest)
returns (stream SubscribeResponse);
}
```

gNOI

```
service System {
  rpc Ping(PingRequest)
returns (stream PingResponse) {}
  rpc Traceroute(TracerouteRequest)
returns (stream TracerouteResponse) {}
  rpc Time(TimeRequest)
returns (TimeResponse) {}
  rpc SetPackage(stream SetPackageRequest)
returns (SetPackageResponse) {}
…
}
```

gRIBI

```
service gRIBI {
  rpc Modify(stream ModifyRequest)
returns (stream ModifyResponse);
  rpc Get(GetRequest)
returns (stream GetResponse);\
}
```

- OpenConfig Service Interfaces
  - gNMI: gRPC Network Management Interface
  - gNOI: gRPC Network Operations Interface
  - gRIBI: gRPC Routing Information Base Interface
- Ref. https://github.com/openconfig