OpenConfig + gNMI: A quick primer

- gNMI == Google Network Management Interface
  - Open source protocol
  - Stream state and configuration data to/from targets
    (i.e. network devices or gNMI-enabled services)
  - Defined as a specification and protobuf model
  - Provided RPCs are **Subscribe**, **Get**, **Set**, & **Capabilities**
OpenConfig + gNMI: A quick primer

- OpenConfig == working group that defines vendor-agnostic data models
  - Develops various standards and protocols for managing networks
  - Models are defined in YANG (RFC 6020)
  - Tree structure of “leaves”

*Select portion of the OpenConfig tree for example purposes*
OpenConfig + gNMI: “Streaming Telemetry”

- Polled telemetry (pull)
  - SNMP
  - Screen scraping
  - Proprietary APIs
- Streaming telemetry (push)
  - gNMI
  - NETCONF
  - sFlow
- Models
  - OpenConfig
  - IETF Models
  - Proprietary
OpenConfig + gNMI: Existing gNMI Systems

- Surveyed existing systems that support gNMI and OpenConfig\(^1\)
- We found they were limited in their capabilities:
  - Only supported one or few devices
  - No high availability
  - Focused on Subscribe with no plans to add other RPCs
  - Lacking support for dynamic target and client connections
- Reference work at github.com/openconfig/gnmi
  - Functional code with no robust system

\(^1\) See Appendix A
gnmi-gateway: Goals

- Provide highly available gNMI streams to clients (clustering)
- Allow for many dynamic clients and targets (target loaders)
- Distribute gNMI streams to non-gNMI systems (exporters)
- Utilize existing code from github.com/openconfig/gnmi
- Plan support for all gNMI RPCs (Subscribe, Get, Set, & Capabilities)
- Plan for extensibility
- No coding needed to start using gNMI data

- We built a service in Golang to accomplish our goals: gnmi-gateway
gnmi-gateway: Deployment Scenarios

![Diagram showing deployment scenarios]

- **Single instance (no clustering)**
  - Target List (e.g. NMS)
  - gNMI Targets
  - gNMI Clients
  - Non-gNMI (Exporter) Services

Colin McIntosh
colin@netflix.com
gnmi-gateway: Deployment Scenarios
gnmi-gateway: Deployment Scenarios

![Diagram of gnmi-gateway deployment scenarios]

- Target List (e.g. NMS)
- Detached Exporter (no clustering)
- gNMI Targets
- gNMI Clients
- Non-gNMI (Exporter) Services

Colin McIntosh
colin@netflix.com
gnmi-gateway: Deployment Scenarios

Detached Exporters (with clustering)

Zookeeper

gNMI Targets

Replication
Message flow
Coordination

Exporter
Exporter
Exporter

Non-gNMI Services

gNMI Clients

gnmi-gateway
gnmi-gateway
gnmi-gateway
gnmi-gateway
**gnmi-gateway: Clustering**

- Multiple gnmi-gateway instances can be clustered to provide quick failover for gNMI streams
- Currently using Zookeeper, but can be extended to other services
gnmi-gateway: Clustering
Target Loaders allow for dynamic loading/unloading of gNMI targets

Examples
- Watch an NMS for new devices with specific tags
- Form connections to static targets like an SNMP poller that has a gNMI Subscribe interface

Extendable with a Golang interface

Included Target Loaders:
- Watched File
- Netbox

```go
type TargetLoader interface {
    // Get the Configuration once.
    GetConfiguration() (*targetpb.Configuration, error)
    // Start the loader, if necessary.
    Start() error
    // Start will be called once by the gateway after StartGateway is called.
    // Start watching the configuration for changes and send the entire
    // configuration to the supplied channel when a change is detected.
    WatchConfiguration(chan<- *connections.TargetConnectionControl) error
}
```
**gnmi-gateway: Target Loaders**

Target Configuration:
- **Target Name**: Device Hostname / UUID
- **Request Target**: * or Target Name
- **Addresses**: List of reachable addresses
- **Creds**: auth username & password
- **TLSConfig**: mutual TLS certs
- **SubscribeRequest**: list of subscriptions (paths) to send
- **Meta**: Arbitrary target metadata

**Target Loader Thread**
- (one per target loader)
  - **Get necessary data for Target Loader**
  - **Generate new target configurations**
  - **Send changed target configurations to gnmi/client**
  - **Wait for source data to change**
gnmi-gateway: Exporters

- Exporters forwarded data to gNMI + OpenConfig unaware systems
- For example, sending metrics to a time-series database
- Extendable with a Golang interface
- Included Exporters:
  - Prometheus
  - Atlas
  - Kafka

```
// Exporter is an interface to send data to other systems and protocols.

type Exporter interface {
  Start(*cache.Cache) error
  Export(*ctree.Leaf) (*ctree.Leaf)
}
```
gnmi-gateway: Exporters

rate(interfaces_interface_state_counters_out_octets(interfaces_interface_name="Port-Channel16")[5m]) * 8

Graph: Prometheus

Colin McIntosh
colin@netflix.com
gnmi-gateway: Exporters

Cluster Member List

gNMI Server Thread (one per instance)

gnmi/subscribe

Exporter Goroutine (zero or many per instance)

Exporter...

Note: notifications from cluster members are blocked from being sent to other cluster members and exporters to prevent a loop and duplicate exports.

gNMI Clients

Other Cluster Members

Non-gNMI Services
gnmi-gateway: Other Good Stuff

- Included Dockerfile for easy testing
- Examples for gnmi-gateway configurations
- Design docs and diagrams are available in GitHub
gnmi-gateway: Next Steps

- Add Get, Set, & Capabilities RPCs
- Develop new exporters
  - InfluxDB
  - Hive
- Include additional configuration examples
- Include a more extensive deployment guide in the docs
“Live” Demo

Colin McIntosh
colin@netflix.com
Thank You.

github.com/openconfig/gnmi-gateway

Colin McIntosh
colin@netflix.com
Appendix A - Existing gNMI services

During our research we searched for services that provided gNMI or OpenConfig capabilities and along with the ability to export data to different protocols and systems. These are the existing services we reviewed:

- [gnmi_collector](#)
- [gNMI Plugin for Telegraf](#)
- [Panoptes](#)
- [Cisco Big Muddy](#)
Appendix B - Internals

Diagram Link
Appendix C - OpenConfig & gNMI Links

- gNMI GitHub Repo
- gNMI Specification
- OpenConfig FAQ