Stratum’s Phal Attribute DB

”What is it Good for?”

Craig Stevens
Dell EMC

Why are you here?

- You’re a developer that’s been tasked with porting Stratum to your hardware platform
- Your interested in how Stratum interfaces into the underlying platform hardware in a flexible and abstract way
- You’ve heard what a great group of people the Stratum team are
- You stumbled into the wrong room and you’re about to leave…..

Stratum Architecture

External gRPC Services

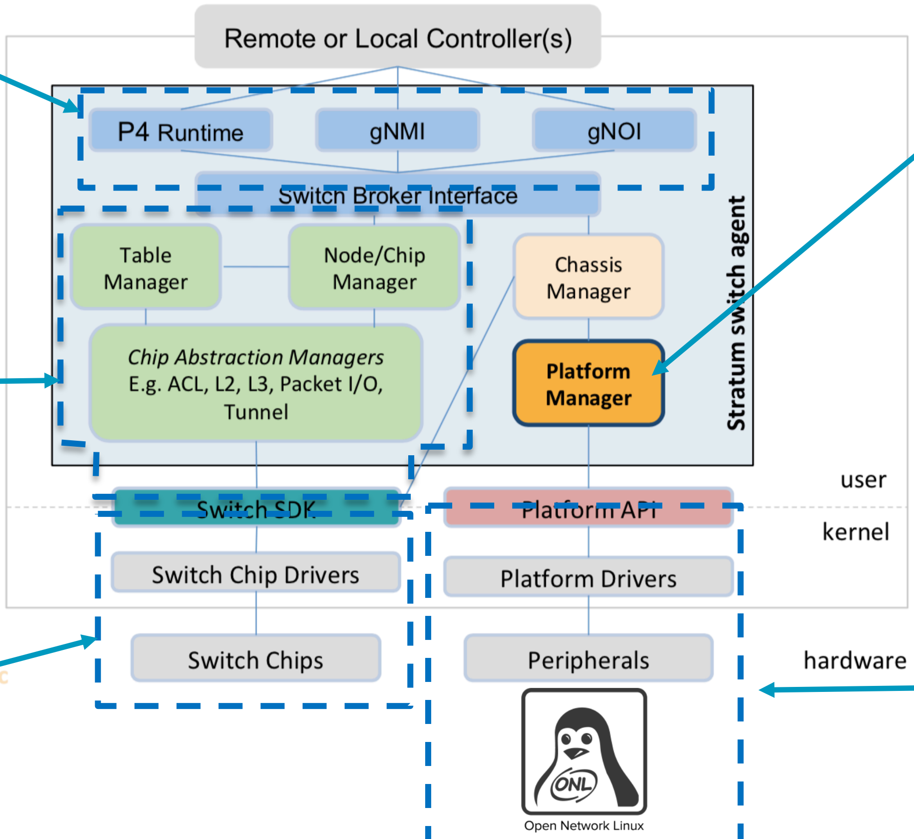
- Exposes the P4 Runtime, gNMI and gNOI services
- Calls down into the Switch Interface

Switch Chip Managers

- Programs and manages the switch chips
- PI and FPM based implementations (includes chip specific SDK Wrappers)

Switch Chip Drivers

- Vendor specific hardware and drivers (i.e. Tofino & Tomahawk)



Platform Manager (PHal)

- Abstract platform hardware management
- Provides caching layer for platform API calls
- Get, Set and Subscribe (streaming)

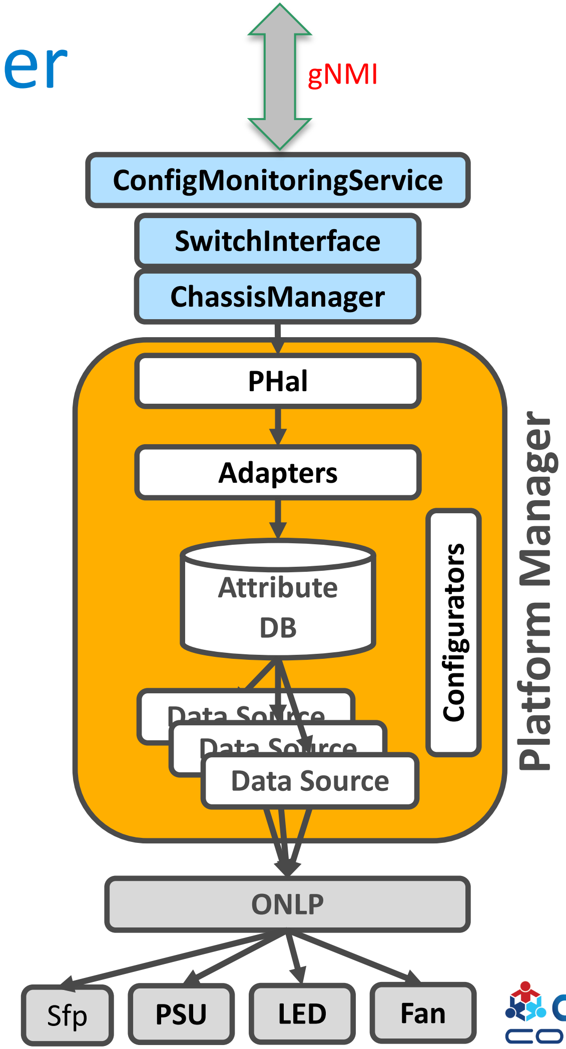
Platform OS

- Currently Stratum using ONL
- Exposes APIs for platform hardware (i.e. Sfp, Fan, PSU, LED, Thermals)

Open Pluggable Architecture

Platform Manager

- Provides an abstract way of managing the platform hardware (i.e. Sfps, PSUs, Fans, LEDs & Thermal sensors)
- PHal (Platform Hardware Abstraction Layer)
  - PHal class provides the high level interface for the platform
  - Manages the platform events
- Adapters
  - Translates Client (i.e. Phal) requests from attribute database protobuf
  - Calls into the Phal Attribute Database for access to attributes
- Attribute Database
  - Provides abstracted access to platform attributes
  - Caches the attributes to allow scaling of gNMI requests (i.e. Get, Set & Subscribe requests)
- Configurators
  - Reads the phal db config file and wires in the datasource attributes into the attribute database
  - Dynamic configurators will do this on-demand (i.e. when an Sfp is inserted or removed).



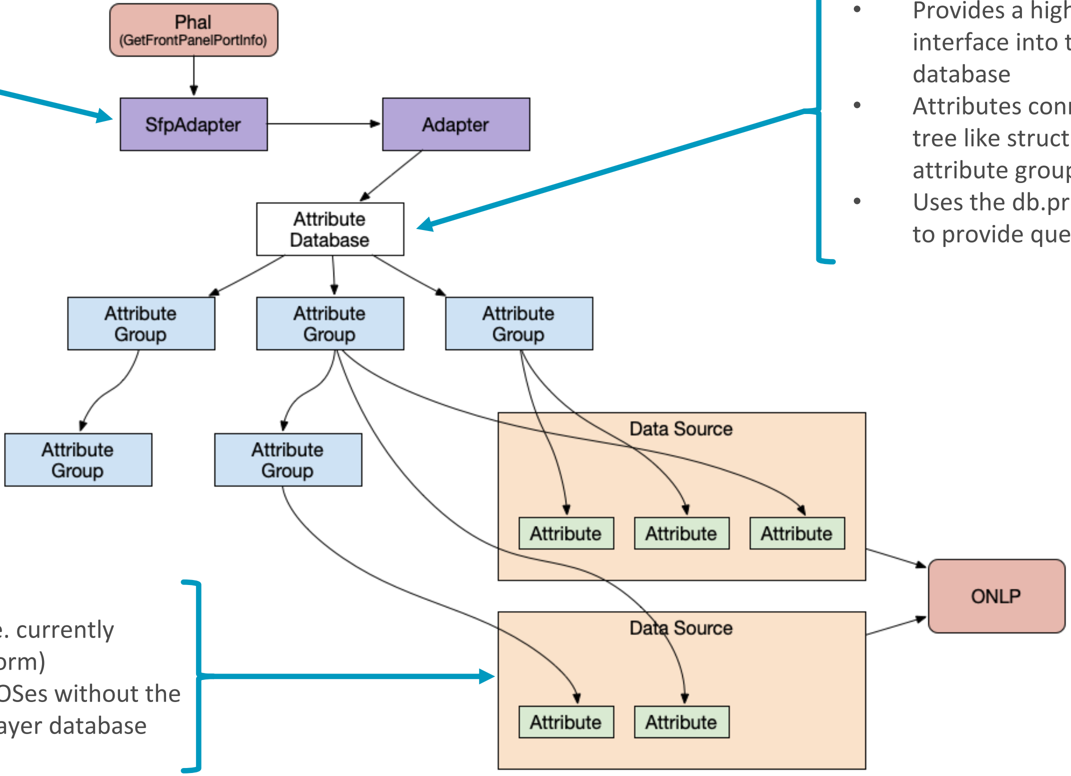
The PHal Attribute DB

Adapters

- make Get, Set or Subscribe calls into the DB
- translate PhalDB message to the callers format (i.e. Sfp to hal.proto)

Attribute Database

- Provides a high level query interface into the attribute database
- Attributes connected into a tree like structure of attribute groups
- Uses the db.proto protobuf to provide query responses

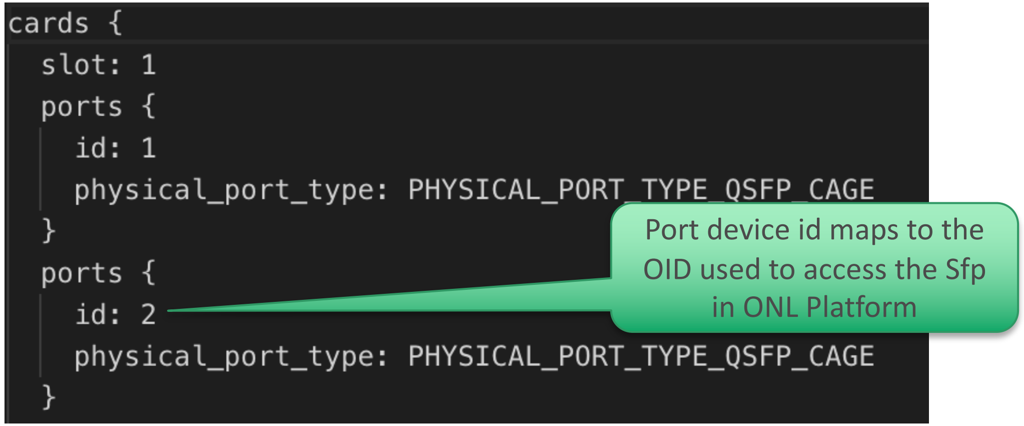


Datasources

- Are platform OS specific (i.e. currently implemented for ONL Platform)
- Can be rewritten for other OSes without the need to change the upper layer database queries or code.

Reference: https://github.com/opennetworkinglab/stratum/blob/master/stratum/docs/phal.md

PHal Configuration File


Port device id maps to the OID used to access the Sfp in ONL Platform

```
cards {
  slot: 1
  ports {
    id: 1
    physical_port_type: PHYSICAL_PORT_TYPE_QSFP_CAGE
  }
  ports {
    id: 2
    physical_port_type: PHYSICAL_PORT_TYPE_QSFP_CAGE
  }
```

- Provides configuration of platform hardware and how it gets wired into the attribute database
- Slot and device ids are optional (if not specified then a 1-base index is used based on the position in the config file)
- Default cache policy is no-cache (i.e. ONL Platform API called on every Get/Poll)

Device ID:
0x00000002

+

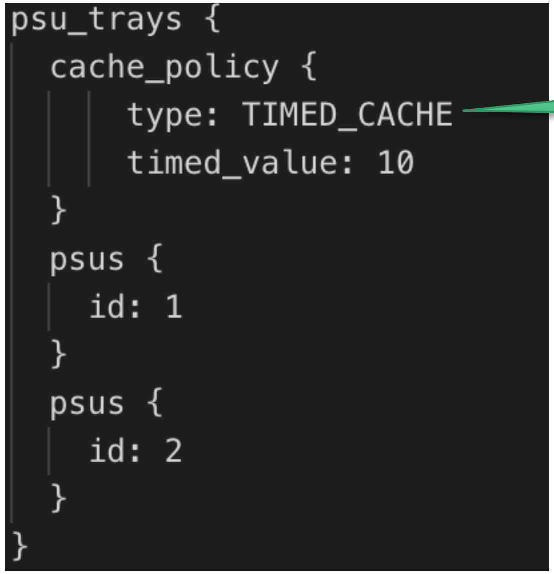
ONLP Device Types:
0x03000000: Thermal
0x04000000: Fan
0x05000000: PSU
0x06000000: LED
0x07000000: Sfp

→

ONLP OID
0x07000002 = 117440514

Reference: https://github.com/opennetworkinglab/stratum/blob/master/stratum/hal/lib/phal/phal.proto

PHal Configuration File (Cache Policy)



```
psu_trays {
  cache_policy {
      type: TIMED_CACHE
      timed_value: 10
  }
  psus {
    id: 1
  }
  psus {
    id: 2
  }
}
```

A cache policy can be specified at the chassis, group or device level.

Cache Policy Types:

- **NO_CACHE (default):** ONLP API called for every Get/Poll of an attribute
- **NEVER_UPDATE:** ONLP API never called
- **FETCH_ONCE:** ONLP API called once for an attribute and then cached value used
- **TIMED_CACHE:** ONLP API only called based on the cache time value for a device

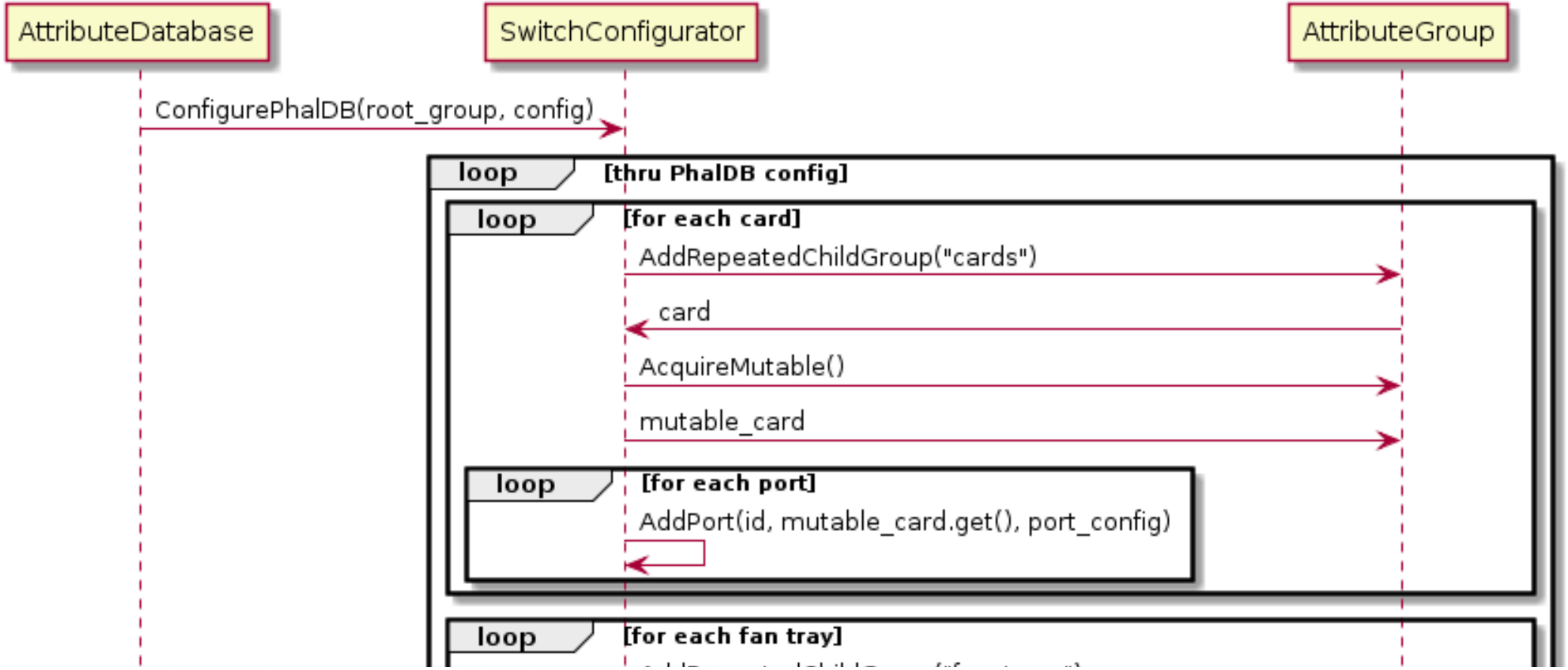
Reference: https://github.com/opennetworkinglab/stratum/blob/master/stratum/hal/lib/phal/phal.proto

The Switch Configurator

- Switch configurator uses the phal.proto protobuf to configure the attribute database
- Stratum uses two modes of generating the PHal DB configuration
    1. Reads in a given PHal DB configuration file specified on stratum startup with the "–phal_config_path" flag
    2. Stratum will automatically generate a default phal configuration based on the OIDs retrieved from the ONLP API (Note: default Cache Policy on "No_Cache" used for all attributes, not recommended for production).
- The switch configurator will then use the PHal configuration to wire the datasource attributes into the attribute database (see workflow on following page)

Reference: https://github.com/opennetworkinglab/stratum/blob/master/stratum/docs/configurators.md

Switch Configurator Workflow

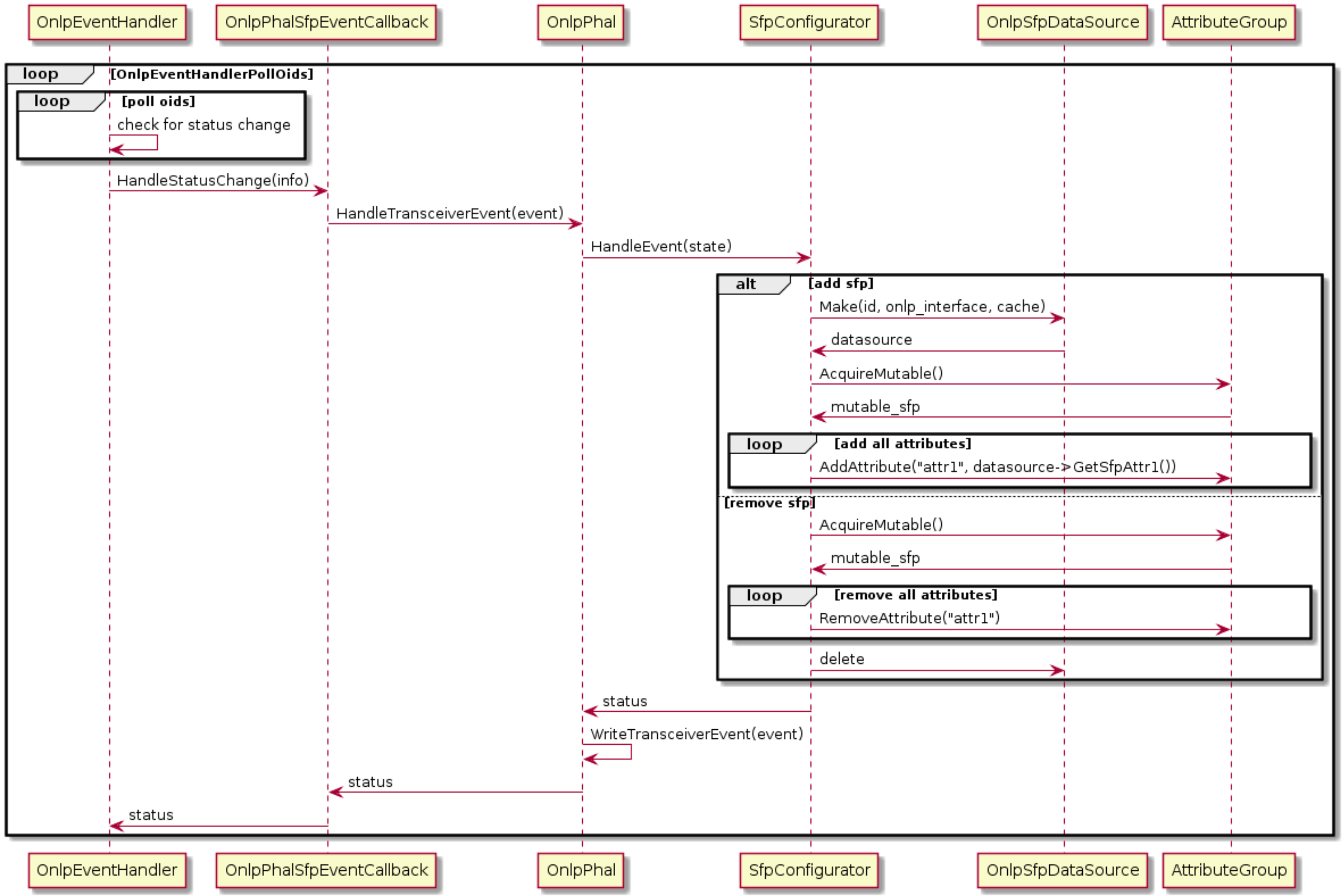


Sfp Configurator

- SfpConfigurator is a “dynamic configurator” called when the Sfp is either inserted or removed.
- Handles the rewiring of the attribute database (i.e. adds or removes the appropriate attributes when the sfp is inserted or removed)
- Is called from the ONLP event handler when Sfp state changes are noticed.
- PSU and Fan configurators soon to be dynamic
    - (have been written and awaiting PR approval before being merged into master)

Reference: https://github.com/opennetworkinglab/stratum/blob/master/stratum/docs/configurators.md

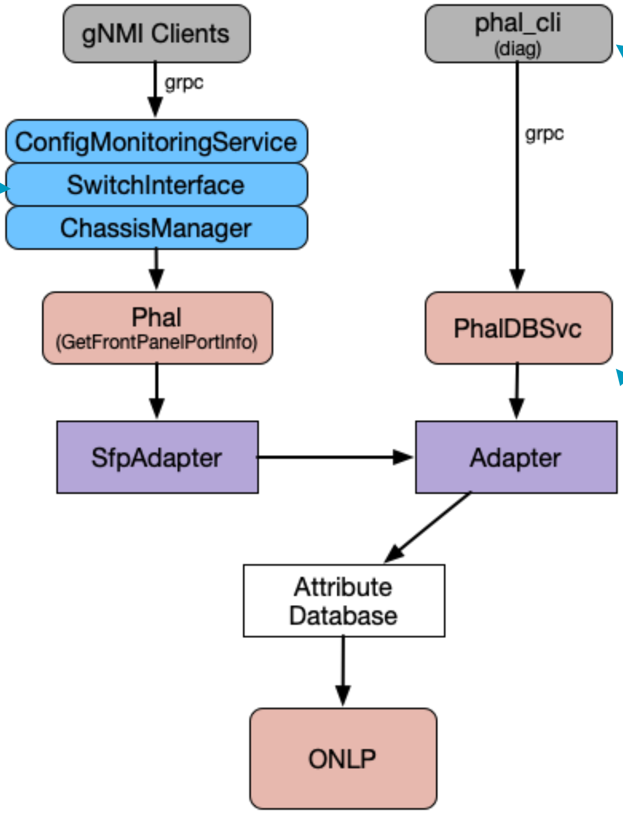
Sfp Configurator Workflow



The PHal DB Cli Tool

gNMI

- Is the main configuration and management interface for Stratum
- Uses the OpenConfig models
- Is wired into the Phal Attribute DB using Adapters
- Currently only the Sfp attributes are wired in



phal_cli tool

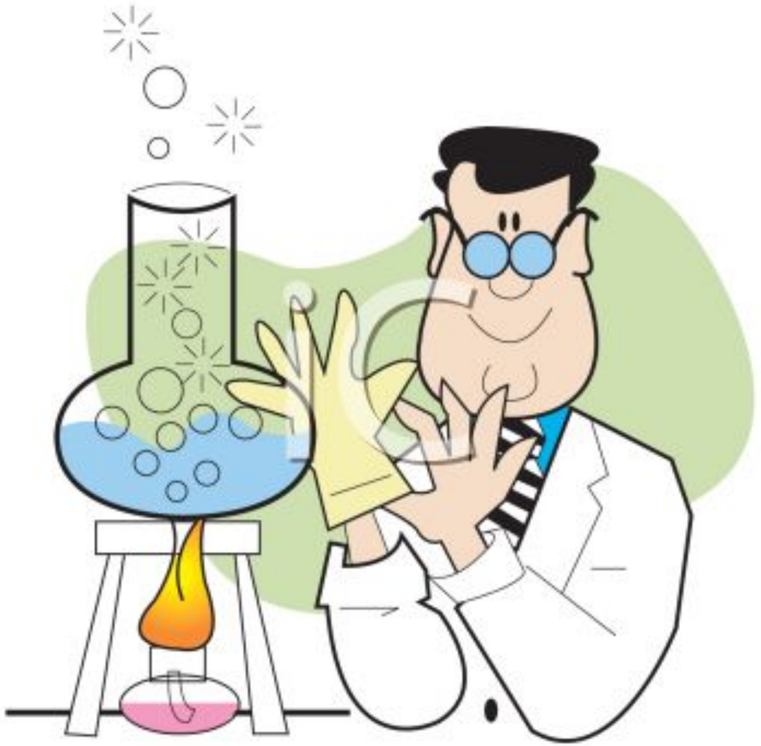
- Provides a high level query cli into the attribute database
- Can be run against a running stratum instance for development and diagnostics
- Uses the PhalSBSvc gRPC service so can be run on or off box

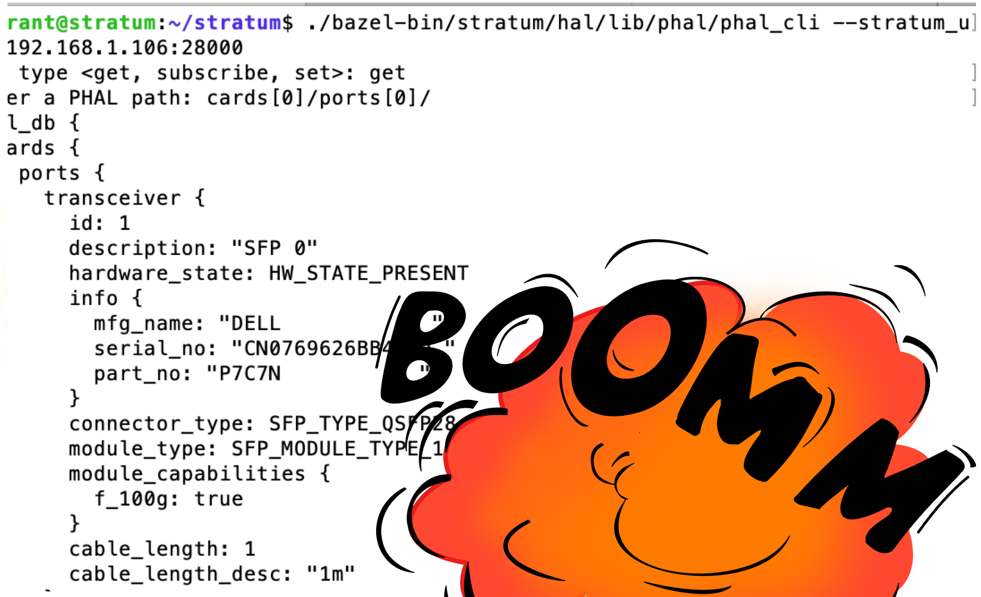
PhalDBSvc

- Exposes a Get, Set & Susbcribe service to a running Stratum Phal Attribute DB
- Can be called on or off box for development and diagnostics purposes

Reference: https://github.com/opennetworkinglab/stratum/blob/phaldb_adapter/stratum/docs/phal_development_tutorial.md

Demo





```
rant@stratum:~/stratum$ ./bazel-bin/stratum/hal/lib/phal/phal_cli --stratum_u
192.168.1.106:28000
 type <get, subscribe, set>: get
er a PHAL path: cards[0]/ports[0]/
l_db {
ards {
 ports {
   transceiver {
     id: 1
     description: "SFP 0"
     hardware_state: HW_STATE_PRESENT
     info {
       mfg_name: "DELL
       serial_no: "CN0769626BB4
       part_no: "P7C7N
     }
     connector_type: SFP_TYPE_QSFP28
     module_type: SFP_MODULE_TYPE_1
     module_capabilities {
       f_100g: true
     }
     cable_length: 1
     cable_length_desc: "1m"
```

How to Engage with Community

- Get started with the basic tutorial
  - https://github.com/stratum/tutorial
- Stratum uses github for source management, issue tracking and pull requests
  - File bugs, request features and submit patches on github
- Join the Stratum announcements mailing list
  - https://lists.stratumproject.org/listinfo/stratum-announce
  - (We will provide more details on joining developer lists and slack soon)
- Attend Stratum Technical Steering Team call
  - (Currently alternative between Wed 4:30pm & Thu 10am Pacific)

Thank You

Follow Up Links:

https://www.opennetworking.org/stratum/