Model-Driven APIs for Network Programmability

Santiago Álvarez
March 2016
# Model-Driven Programmability Stack

<table>
<thead>
<tr>
<th>Apps</th>
<th>App1</th>
<th>App2</th>
<th>App3</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIs</td>
<td>Model-Driven APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YANG Development Kit (YDK)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>NETCONF</td>
<td>RESTCONF</td>
<td>gRPC</td>
</tr>
<tr>
<td>Encoding</td>
<td>XML</td>
<td>JSON</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>SSH</td>
<td>HTTP</td>
<td></td>
</tr>
<tr>
<td>Models</td>
<td>Data Models (native, open)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Benefits of Model-Driven Programmability

- Model based, structured, computer friendly
- Multiple model types (native, OpenConfig, IETF, etc.)
- Models decoupled from transport, protocol end encoding
- Choice of transport, protocol and encoding
- Model-driven APIs for abstraction and simplification
- Wide standard support while leveraging open source
Model-Driven APIs

- Simplify app development
- Abstract transport and encoding
- API generated from YANG model
- One-to-one correspondence between model and class hierarchy
- Multi-language (Python, C++, Ruby, Go, etc.)
Generation of Model-Driven APIs Using YANG Development Kit (YDK)

YDK

Data Models (YANG) → YDK-gen → API Generator

YDK-Py

Python Docs
C++ Docs
Ruby Docs
go Docs
C# Docs

 Docs

Docs
YDK-Py API Structure

- **Models**
  - (BGP, IS-IS, etc)

- **Services**
  - (CRUD, etc)

- **Providers**
  - (NETCONF, etc)
Client-Side Validation

- YDK service will automatically perform local (client-side) validation
- Type check (enum, string, etc.)
- Value check (range, pattern, etc.)
- Semantic check (key uniqueness/presence, mandatory leaves, etc.)
# Cisco YDK-Py OC-BGP “Hello world”
from ydk.services import CRUDService
from ydk.providers import NetconfServiceProvider
from ydk.models.bgp import bgp

if __name__ == "__main__":
    provider = NetconfServiceProvider(address="10.0.0.1",
                                      port=830,
                                      username="admin",
                                      password="admin",
                                      protocol="ssh")

    crud = CRUDService()  # create CRUD service
    bgp = bgp.Bgp()  # create oc-bgp object
    bgp.global_.config.as_ = 65000  # set local AS number
    crud.create(provider, bgp)  # create on NETCONF device

    provider.close()
    exit()

# End of script

module: openconfig-bgp
    +--rw bgp!
        +--rw global
            |    +--rw config
            |        +--rw as
            |        +--rw router-id?
            |    +--ro state
            |        +--ro as
            |        +--ro router-id?
            |        +--ro total-paths?
            |        +--ro total-prefixes?
            |...

A YDK-Py “Hello World” Using OC-BGP
Roadmap

- Model bundles
- YANG deviation support
- YANG RPC support
- NETCONF service
- Additional language support in (YDK-gen)
Resources

- YDK Python API (YDK-Py)
  https://github.com/CiscoDevNet/ydk-py
- YDK-Py additional sample apps
  https://github.com/CiscoDevNet/ydk-py-samples
- YDK Generator (YDK-gen)
  https://github.com/CiscoDevNet/ydk-gen
- Related blog posts
  http://blogs.cisco.com/author/santiagoalvarez
Demo
OC-BGP IPv4/IPv6 Unicast (Python)
TOMORROW starts here.