Overview

❖ YANG components
  ➢ ONOS YANG tools
  ➢ YANG management system

❖ Application development using YANG
  ➢ Interface modeling in YANG
  ➢ NBI development based on YANG
  ➢ YANG based (SBI) driver / provider’s interaction with device
ONOS YANG tools

Application / device schema

ONOS YANG TOOLS

Generated Java Files for Interaction with YMS

maven / buck plugin

Metadata containing Application Schema.

Used by YMS to ENCODE and DECODE applications external interaction.
ONOS YANG Tools

Maven plugin

```xml
<dependency>
  <groupId>org.onosproject</groupId>
  <artifactId>onos-yang-maven-plugin</artifactId>
  <version>1.10</version>
</dependency>

<plugin>
  <groupId>org.onosproject</groupId>
  <artifactId>onos-yang-maven-plugin</artifactId>
  <version>1.10</version>
  <executions>
    <execution>
      <goals>
        <goal>yang2java</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Buck plugin

```python
yang_library(  
  deps = COMPILE,  
  name = 'xxx-yangmodel',  
  srcs = glob(['src/main/yang/*.yang']).
)
```

Output

```
\checkmark target
  \checkmark classes
  \checkmark generated-sources
    \checkmark org
    \checkmark yang
      \checkmark v1
    \checkmark urn
  \checkmark temp
  \checkmark yang
  \checkmark resources
```
YMS: It is the core of YANG in ONOS, it provides a framework for the drivers/applications to register their schema. It automates the CODEC functionality required by driver/application. It uses YANG utils generated java objects for communication with drivers/application.
RESTCONF CLIENT/ GUI / NETCONF etc

YANG Management System (NBI)

YANG based NBI in ONOS

Application Service Invocation

Pre-Requisites
1) Application model it's interface in YANG.
2) Application uses YANG Utils to auto-generate YANG modeled JAVA classes.

call flow
Step 1: Application register it's schema with YMS.
Step 2: Restconf client invokes application's Rest interface.
Step 3: Restconf server translates JSON to abstract data tree using YMS and send to YMS for execution.
Step 4: YMS identifies application schema corresponding to request and invokes application JAVA interface.

External Notification

Pre-Requisites
1) Application model it's notification in YANG.
2) Application uses YANG Utils to auto-generate YANG modeled JAVA classes.

call flow
Step 1: Application register it's schema with YMS APP.
Step 5: YMS registers as a listener.
Step 6: Application raises a notification.
Step 7: YMS sends the notification to Restconf server in abstract data tree.
Step 8: Restconf server notifies the event to Restconf client as a server sent event.
YANG based SBI in ONOS

Pre-Requisites
1) Driver model its Device Schema in YANG.
2) Driver uses YANG Utils to auto-generate YANG modeled JAVA classes.

Call Flow
Step 1: Driver uses YMS framework to register device schema.
Step 2: Driver creates YANG modeled JAVA class object and calls YMS to obtain corresponding XML.
Step 3: Driver uses Netconf to communicate with device.

Likewise reverse flow will use YMS to obtain YANG modeled JAVA class object from a given XML.
Interface Modeling in YANG

device service with port info

ONOS topology schema

link service
Application Service Implementation

**Pre-Requisites**
1. Application model it's interface in YANG.
2. Application uses YANG Utils to auto-generate YANG modeled JAVA classes.
3. Application register it's schema with YMS.

**call flow**
Step 1: Applications performs the business logic or retrieves the data.
Step 2: Construct the information YANGtools generated classes.
Step 3: Pass the data to YMS to return to external client.
SBI development based on YANG

Pre-Requisites
1) Driver model its Device Schema in YANG.
2) Driver uses YANG Utils to auto-generate YANG modeled JAVA classes.

Call Flow
Step 1: Driver uses YMS framework to register device schema.
Step 2: Driver creates YANG modeled JAVA class object and calls YMS to obtain corresponding XML.
Step 3: Driver uses Netconf to communicate with device.

Likewise reverse flow will use YMS to obtain YANG modeled JAVA class object from a given XML.
Roadmap

◆ ONOS YANG store
  ➢ Seamless usage for applications to support YANG based store

◆ Common Driver
  ➢ Abstracting driver / provider from protocol interaction
  ➢ Model device access as remote data store
  ➢ RPC support for device interaction
Thank You