OpenDaylight Netconf/Yang tutorial

ENOG13 - St.Petersburg

Evgeny Zobnitsev (FACTOR GROUP)

Agenda

- **Orchestration problem statement**
- Netconf overview
- Restconf overview
- Yang overview
- OpenDaylight overview
- DEMO

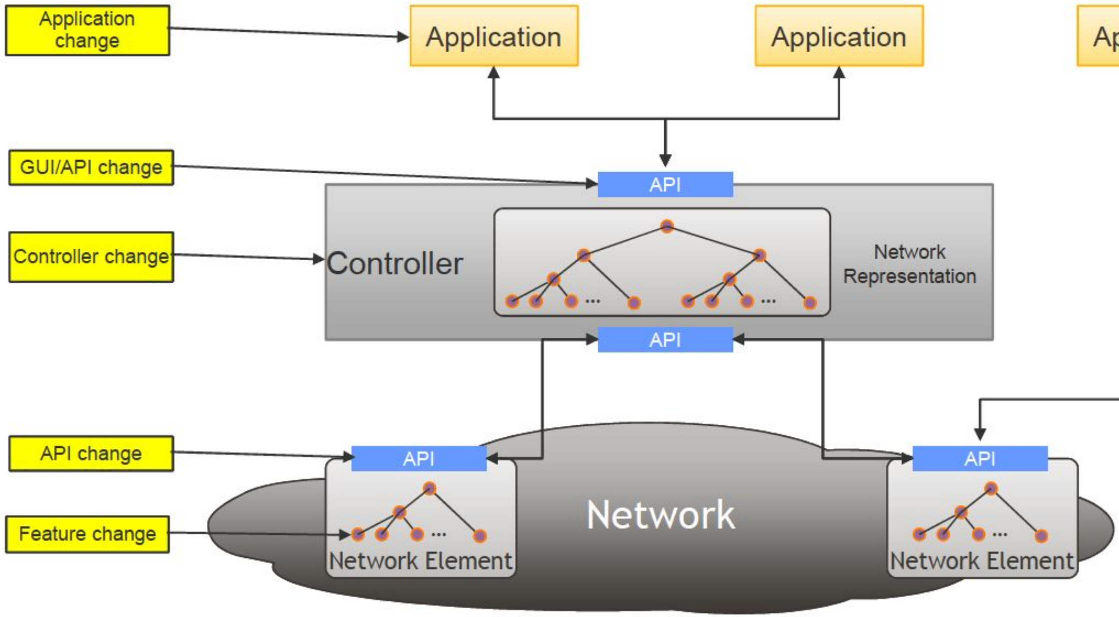
Network configuration management today

- Various proprietary CLIs of the network elements (NEs)
- Various proprietary APIs of the network elements and controller
- Manual configuration or expect scripts, as automation
- Imperative, incremental NEs configuration, lack of abstractions
- Configuration screen scraping from NEs, using parsing
  - The most bad thing, no programmatic knowledge when the CLI command finished.
  - CLI designed to work with humans, not machines.
- SNMP - has not solved this problems (see RFC3535)

Network configuration management today

- Service lifecycle management - workflow-based
  - Workflows describes to the provisioning system, in detail, which steps to take to reach a final state.
  - Each service livecycle action has to be explicitly defined: service create, service change, service edit, service delete.

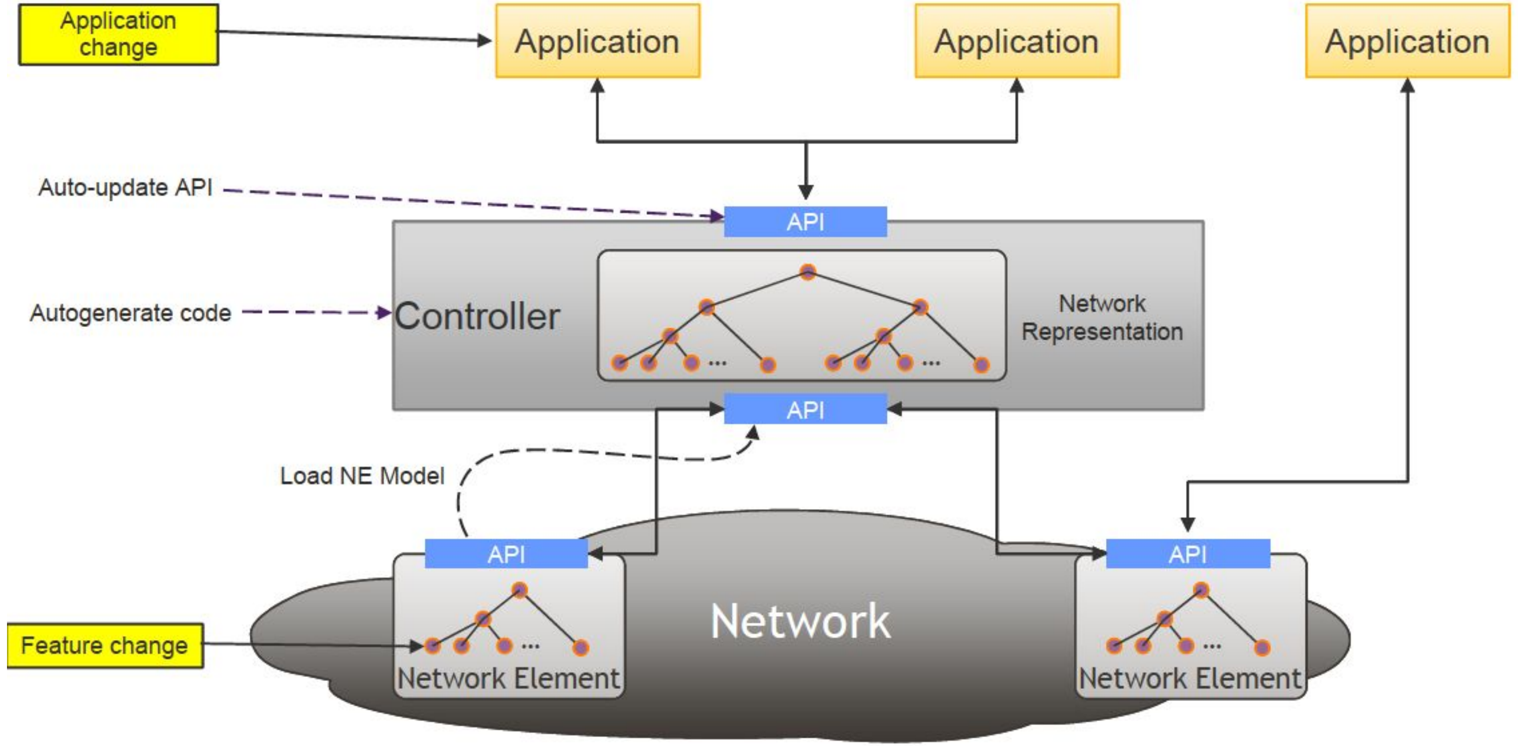
Network application life cycle today



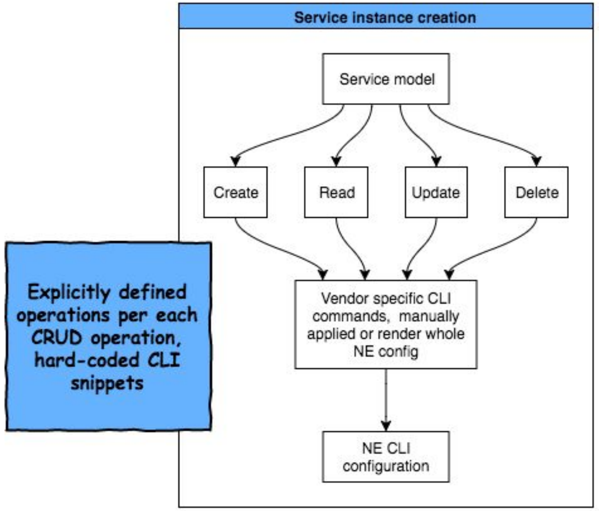
Network configuration management future

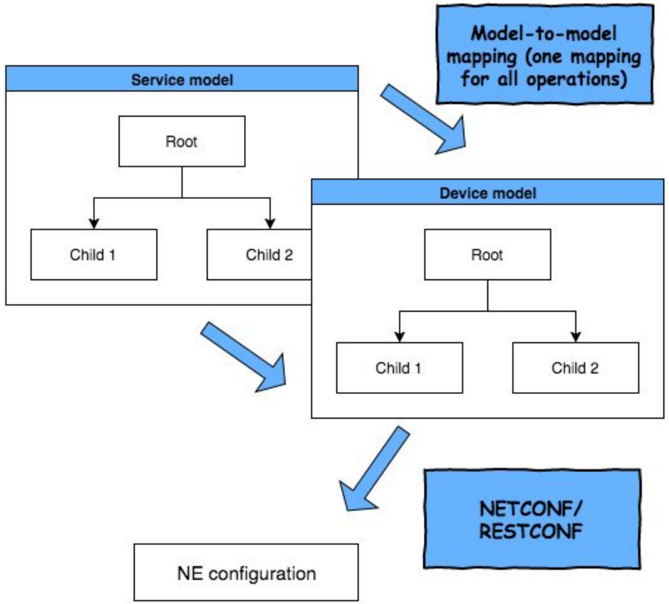
- **Model-driven topology of the network**
  - describes structure of the network.
- **Model-driven services description, auto-generating APIs**
  - describes structure of the network services, APIs.
- **Model-driven configuration of the network elements (NEs)**
  - describes structure of network element configuration, provided and supported by vendor.
- **Model-driven telemetry of the network elements (NEs)**
  - describes structure of network element supported telemetry, provided and supported by vendor.

Network Application Life Cycle (End-to-End Model-Driven Architecture)



Comparison of Model-to-model and Workflow-based service-to-network mapping


Service instance creation
Service model
Create
Read
Update
Delete
Vendor specific CLI commands, manually applied or render whole NE config
NE CLI configuration
Explicitly defined operations per each CRUD operation, hard-coded CLI snippets



OPENDAYLIGHT

Agenda

- Orchestration problem statement
- **Netconf overview**
- Restconf overview
- Yang overview
- OpenDaylight overview
- DEMO

Key Features of NETCONF (RFC6241)

- **Domain-Specific Knowledge**
  - NETCONF was specifically developed to support network configuration
- **Support for Transactionality**
  - NETCONF support Atomicity, Consistency, Isolation, Durability (ACID) transactionality
- **Vendor Device Independence**
  - NETCONF has abilities to get capabilities just from device during <hello> exchange
- **Support of run RPCs to run actions**
  - NETCONF has abilities to run the RPC that device support
- **Device versions built-in support**
  - NETCONF has abilities to run <get-schema> RPC, to download YANG model of device configuration just from device itself

Support for Transactionality

The four properties that define a transaction: **ACID**

- **Atomicity**
  - Transactions are indivisible, all-or-nothing
- **Consistency**
  - Transactions are all-at-once
  - There is no internal order inside a transaction, it is a set of changes, not a sequence
  - Implies that { create A, create B } and { create B, create A } are identical
  - Implies that a system behaving differently with respect to the sequence is not transactional
- **Independence**
  - Parallel transactions do not interfere with each other
  - Transactions appear to happen always-in-sequence
- **Durability**
  - Committed data always-sticks, i.e. remains in the system even in the case of a fail-over, power failure, restart, etc

NETCONF Transport

NETCONF messages are encoded in XML

- Each message is framed by:
  - NETCONF 1.0: a character sequence]]>]]>
  - NETCONF 1.1: a line with the number of characters to read in ASCII
- NETCONF messages are encrypted by SSH
- SSH provides authentication, integrity and confidentiality
- NETCONF is connected oriented using TCP
  - No need for manager to request resends

NETCONF extensibility

- When a NETCONF Manager connects to a NETCONF Server (NE), they send <hello> message
- The contents of the <hello> message declares which NETCONF Capabilities each party is capable of:
  - Some capabilities are defined by the base NETCONF specification
  - Each Yang Data model the device knows is also a capability
  - Other specifications (standards body or proprietary) also define capabilities
- By declaring support for a capability in <hello>, the manager will know which operations it can submit to the NE.
- Extensions go in separate XML namespaces, which makes it easier to build backwards and forwards compatible management applications.

NETCONF transactions

- NETCONF allows a Manager to send down a set of configuration changes, or an entirely new configuration, in a single <edit-config> transaction.
- When doing so, the Manager does not need to:
  - Figure out which order to send down the configuration changes in. All different sequences are treated equal.
  - Recover, if the transaction fails. If the transaction was unsuccessful because of:
    - inconsistency in the configuration
    - an out of memory condition
    - any other reason

    … none of the transaction content has been activated.
- The transaction did not roll back. It was simply never activated.

NETCONF base operations

- <get>
- <get-config>
- <edit-config>
  - test-option (:validate)
  - error-option
  - operation
- <copy-config>
- <commit> (:candidate, :confirmed)
- <discard-changes> (:candidate)
- <cancel-commit> (:candidate)
- <delete-config>
- <lock>
- <unlock>
- <close-session>
- <kill-session>

NETCONF additional operations by capabilities

- <commit>, <discard-changes> (:candidate)
- <validate> (:validate)
- Copy candidate to running
- Discard changes in candidate (copy running to candidate)
- <create-subscription> (:notification)
- <partial-lock>, <partial-unlock> (:partial-lock)
- <commit>, <cancel-commit> (:commit)
- <get-schema> (:ietf-netconf-monitoring)

Agenda

- Orchestration problem statement
- Netconf overview
- **Restconf overview**
- Yang overview
- OpenDaylight overview
- DEMO

Key Features of RESTCONF (RFC 8040)

- It provides a uniform, standardized way for Web applications to access the configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations, and event notifications within a network element.
- The RESTCONF protocol operates on the configuration datastores defined in NETCONF. It defines a set of Create, Read, Update, Delete (CRUD) operations that can be used to access these datastores.
- The YANG language defines the syntax and semantics of datastore content, operational data, protocol operations, and REST operations that are used to access the hierarchical data within a datastore.
- In NETCONF, YANG data nodes are identified with XPath expressions, starting from the document root to the target resource. RESTCONF uses URI-encoded path expressions to identify the YANG data nodes.

Key Features of RESTCONF

- The RESTCONF protocol operates on a hierarchy of resources, each of which can be thought of as a collection of data and a set of allowed methods operating on that data. Resources are accessed via a set of URIs using syntax specified in RFC 8040. The set of YANG modules supported by the server determine the RPC operations, top-level data nodes, and event notification messages supported by the server.

- The RESTCONF protocol does not include a data resource discovery mechanism. Instead,the definitions within the YANG modules advertised by the server are used to construct an RPC operation and data resource identifiers.

RESTCONF resources

- **Root Resource Discovery.** When first connected, clients retrieve the "/.well-known/host-meta" and use the link contained in the resource in subsequent RESTCONF requests.
- **RESTCONF Media Types.** The RESTCONF protocol defines two application-specific media types, yang-data+xml and yang-data+json, for encoding of the YANG data.
- **API Resource.** The RESTCONF API resource contains the root resource for the RESTCONF datastore and operation resources. It is the top-level resource located at "/restconf". The API resource has three child resource: data, operation, yang-library-version.
- **Datastore Resource.** The datastore resource represents the combined configuration and operational state data resources that can be accessed by a RESTCONF client. The datastore resource is handled by the RESTCONF server and cannot be created or deleted by clients.
- **Data Resource.** A data resource represents a YANG data node that is a descendant node of a datastore resource. Each YANG-defined data node can be uniquely targeted by the request-line of an HTTP method. Containers, leafs, leaf-list entries, and list entries are all data resources. List entries are identified by the name of the list followed by "=" and the value of the key(s).

RESTCONF resources

- **Operation Resource.** An operation resource represents an RPC operation defined with the YANG “rpc” statement or a data-model-specific action defined with a YANG 1.1 “action” statement. An operation is invoked using a POST method on the resource. All operation resources representing RPC operations supported by the server are found in the “/restconf/operations” subtree, while operation resources representing YANG actions are identified in the “/restconf/data” subtree matching their location in the YANG-model.

- **Schema Resource.** Clients can retrieve YANG modules from the server. In order to retrieve a YANG module, a client first retrieves the URL for the relevant schema which is stored in the “schema” leaf in the module entry in the yang-library.

- **Event stream (notification) resource.** An event stream resource represents a source for system-generated event notifications. Each stream is created and modified by the server only. A client can retrieve a stream resource or initiate a long-poll server sent event stream.

RESTCONF HTTP methods vs NETCONF

RESTCONF operation	Description	NETCONF operation
HEAD	Get without a body	<none>
OPTION	Discover which operations are sup-ported by a data resource	<none>
GET	Retrieve data and metadata	<get>, <get-config>
POST	Create a data resource	<edit-config> (nc:operation="create")
POST	Invoke an RPC operation	Call RPC directly

RESTCONF HTTP methods vs NETCONF

RESTCONF operation	Description	NETCONF operation
PUT	Create or replace a data resource	<edit-config> (nc:operation="create/ replace"), <copy-config> (PUT on datastore)
PATCH	Create or update but not delete a data resource	<edit-config> (nc:operation depends on patch content)
DELETE	Delete a data resource	<edit-config> (nc:operation="delete")

Agenda

- Orchestration problem statement
- Netconf overview
- Restconf overview
- **Yang overview**
- OpenDaylight overview
- DEMO

Key Features of Yang (RFC6020/7950)

- Service & NE Data Models vs. Information Models (UML)
  - Yang is the data modeling language
- Domain-Specific Language
  - Yang was specifically developed to support network configuration
- NE configuration modeling
  - Yang is rich enough to model NE configuration (often follow the CLI)
- Service configuration modeling
  - Yang is rich enough to model services in the same language as the NE
- Network topology modeling
  - Any new device can be supported, by publishing Yang
- Vendors must create and publish Yang models of their own devices via ietf-netconf-monitoring RFC feature

Yang module structure

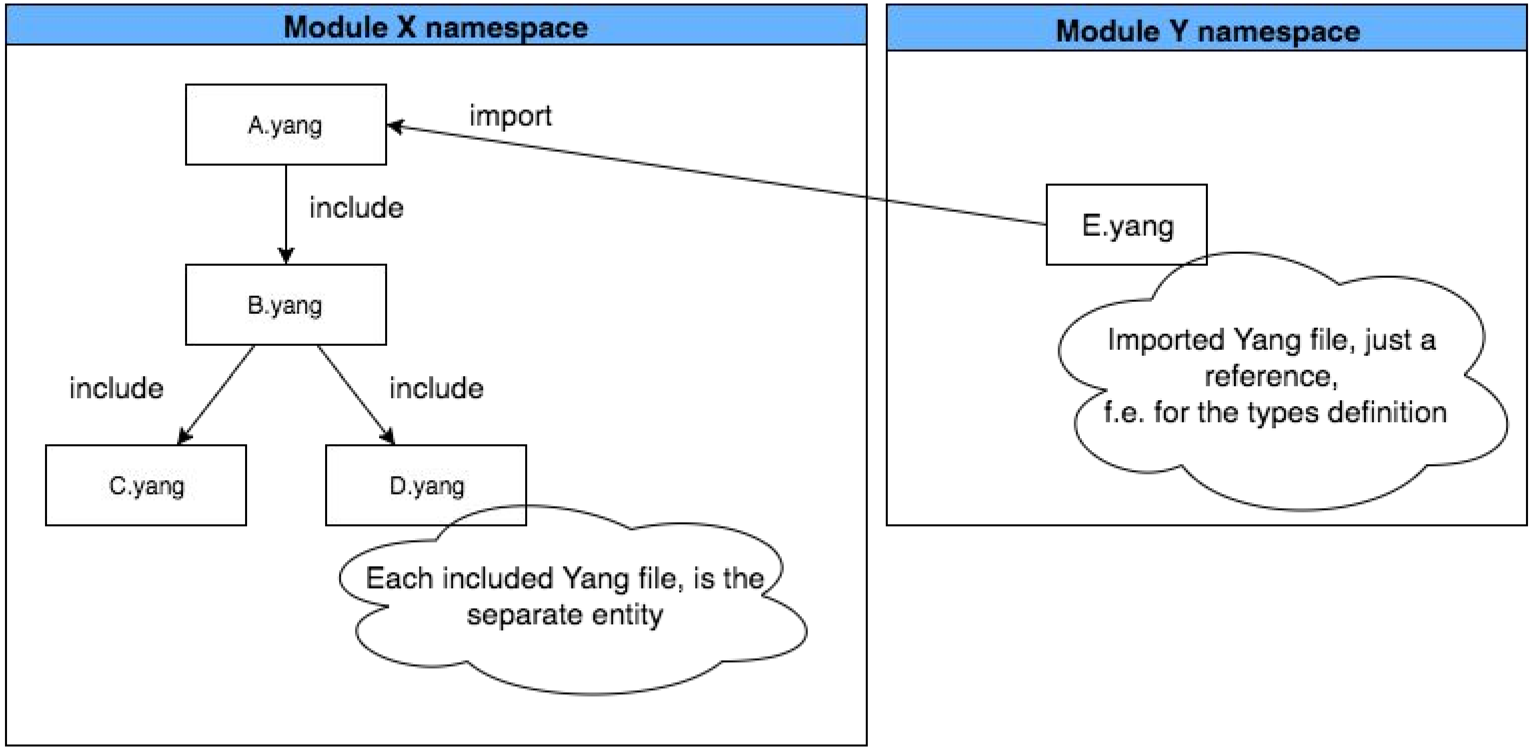
- Header
- Imports & Includes
- Type definitions
- Configuration & Operational data declarations
- Action (RPC) & Notification declarations

Yang header

```
module mplsl3vpn {
  namespace "http://ru/fgts/mplsl3vpn";

  prefix mplsl3vpn;
  import ietf-inet-types {
    prefix inet;
  }
    import junos {
    prefix junos;
  }
  include "ietf-bgp-l3vpn";
  organization "Factor Group";
  contact e@zobnitsev.ru;
  description "This is MPLS L3VPN service for FGTS Lab";
  revision 2014-02-20 {
    description "Initial revision.";
  }
```

Yang imports and includes


Module X namespace
Module Y namespace
A.yang
import
include
B.yang
include
include
C.yang
D.yang
E.yang
Each included Yang file, is the separate entity
Imported Yang file, just a reference, f.e. for the types definition

Yang base types

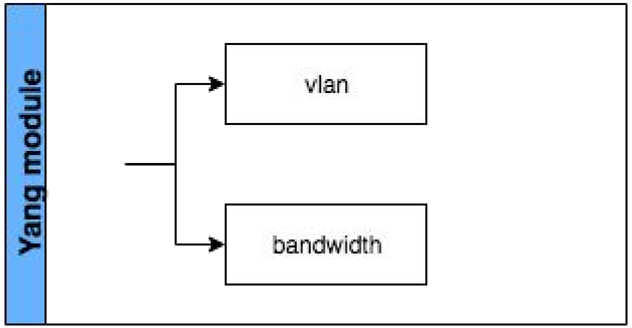
- Most Yang elements have a data type
- Type may be a base type or derived type
- Derived types may be simple typedefs or groupings (structures)
- There are a lot of ready to use IETF types, just import them

Type Name	Meaning
int8/16/32/64	Integer
uint8/16/32/64	Unsigned integer
decimal64	Non-integer
string	Unicode string
enumeration	Set of alternatives
boolean	True or false
bits	Boolean array
binary	Binary BLOB
leafref	Reference
identityref	Unique identity
empty	No value, void
	…and more

Yang data definitions – leaf statement

Holds a single value of a particular type

```
leaf vlan {
  mandatory true;
  type uint16 {
  range "1..4095";
  }
  description "PE-to-CE interface VLAN id";
 }
leaf bandwidth {
 description "Service bandwidth";
 type string {
   description "nn[k,m,g]::Service bandwidth";
   pattern "[0-9]+k|[0-9]+m|[0-9]+g";
 }
}
```



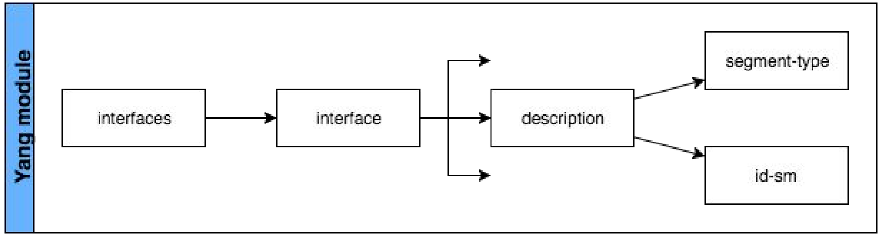
Leaf statement attributes

config	Whether this leaf is a configurable value ("true") or operational value ("false"). Inherited from parent container if not specified
default	Specifies default value for this leaf. Implies that leaf is optional
mandatory	Whether the leaf is mandatory ("true") or optional ("false")
must	XPath constraint that will be enforced for this leaf
type	The data type (and range etc) of this leaf
when	Conditional leaf, only present if XPath expression is true
description	Human readable definition and help text for this leaf
reference	Human readable reference to some other element or spec
units	Human readable unit specification (e.g. Hz, MB/s, C)
status	Whether this leaf is "current", "deprecated" or "obsolete"

Yang data definitions – container statement

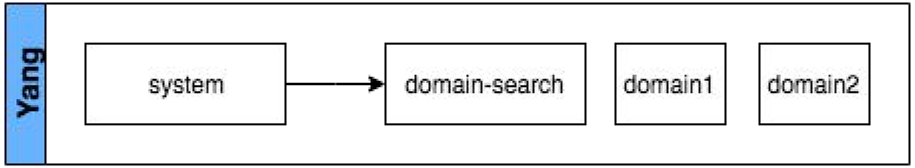
Groups related leafs and containers

```
container description {
  description "Service description";
  leaf segment-type {
    type enumeration {
      enum B2B;
      enum B2C;
      enum B2O;
    }
    description "Market segment type";
  }
  leaf id-sm {
    type string;
    description "Order number in Service Manager";
  }
}
```



Yang data definitions – leaf-list statement

Holds multiple values of a particular type

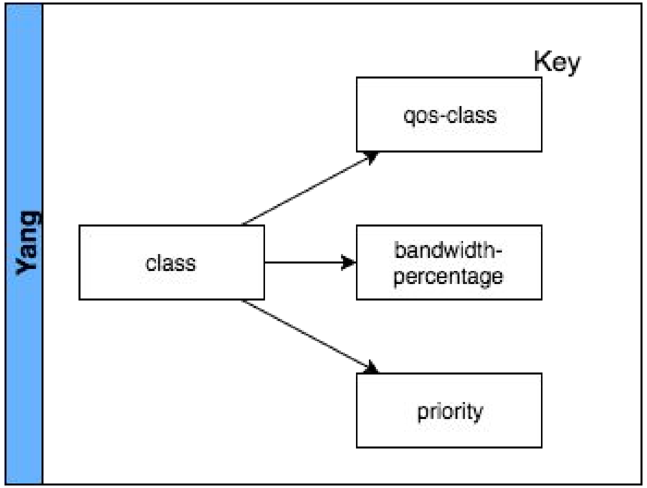


```
leaf-list domain-search {
  type string;
  ordered-by user;
  description "List of domain names to search";
}
```

Yang data definitions – list statement

Holds multiple values of any type

```
list class {
  key "qos-class";
  leaf qos-class {
    type leafref {
      path "/qos/qos-class/name";
    }
  }
  leaf bandwidth-percentage {
    type uint32;
  }
  leaf priority {
    type empty;
  }
}
```



Leaf-list and list statements attributes

max-elements	Max number of elements in list. If max-elements is not specified, there is no upper limit, i.e. "unbounded"
min-elements	Min number of elements in list. If min-elements is not specified, there is no lower limit, i.e. 0
ordered-by	List entries are sorted by "system" or "user". System means elements are sorted in a natural order (numerically, alphabetically, etc). User means the order the operator entered them in is preserved. "ordered-by user" is meaningful when the order among the elements have significance, e.g. DNS server search order or firewall rules.

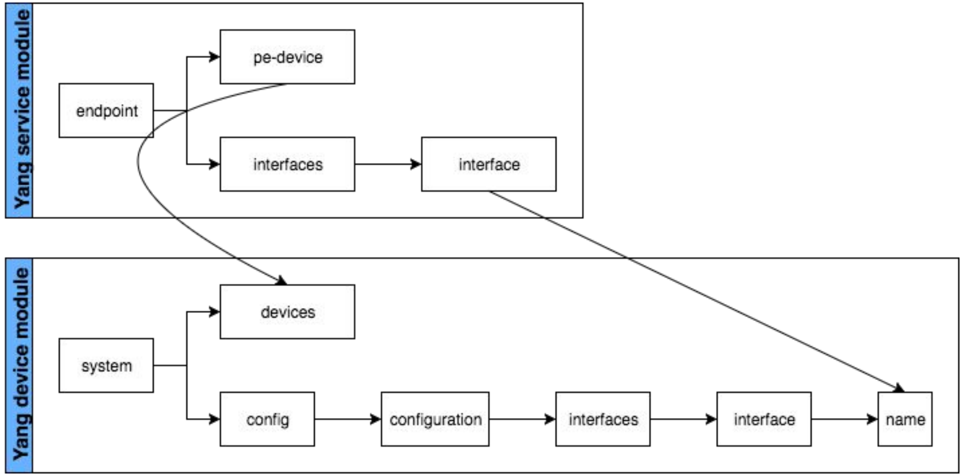
Yang data definitions – leafref statement

To make an element reference one of the rows in a list, set the element type to leafref

- A valid leafref can never be null/empty
- But the parent leaf can be optional
- A valid leafref can never point to a row that has been deleted or renamed
- System checks validity of leafrefs automatically

Yang data definitions – leafref statement

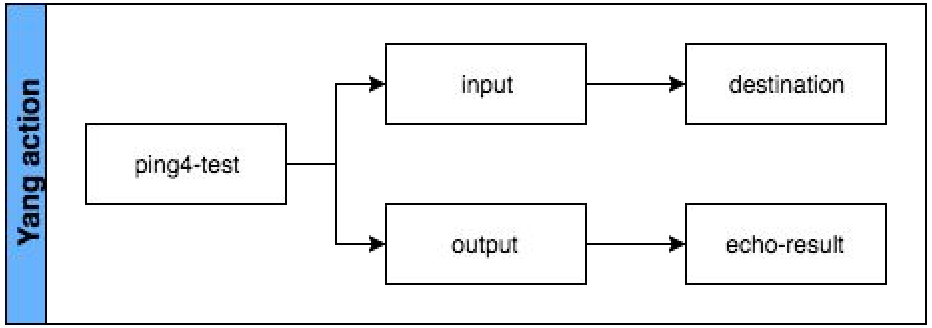
```
list endpoint {
  leaf pe-device {
    description "PE-device selection";
    mandatory true;
    type leafref {
      path "/system:devices/system:device\
          /system:name";
    }
  }
  container interfaces {
    description "Interface parameters on the PE-device";
    leaf interface {
      type leafref {
      path deref(../../pe-device)/../system:config/junos:configuration/junos\
          :interfaces/junos:interface/junos:name;
      }
    }
}
```



Yang actions (RPCs)

Administrative actions with input and output parameters

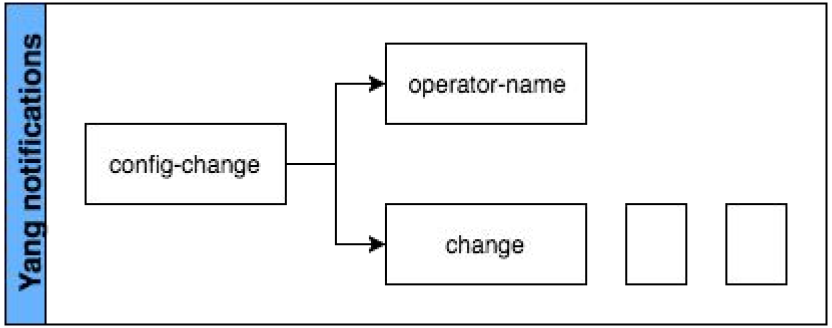
```
rpc ping4-test {
  input {
    leaf destination {
      type inet:ipv4-address;
    }
  }
  output {
    leaf-list echo-result {
      type enumeration {
        enum "reachable" {
          value 0;
          description "Received reply";
        }
        enum "unreachable" {
          value 1;
          description "No reply during timeout";
        }
      }
```



Yang notifications

Notification with output parameters

```
notification config-change {
  description
    ”The configuration changed";
  leaf operator-name {
    type string;
  }
  leaf-list change {
    type instance-identifier;
  }
}
```

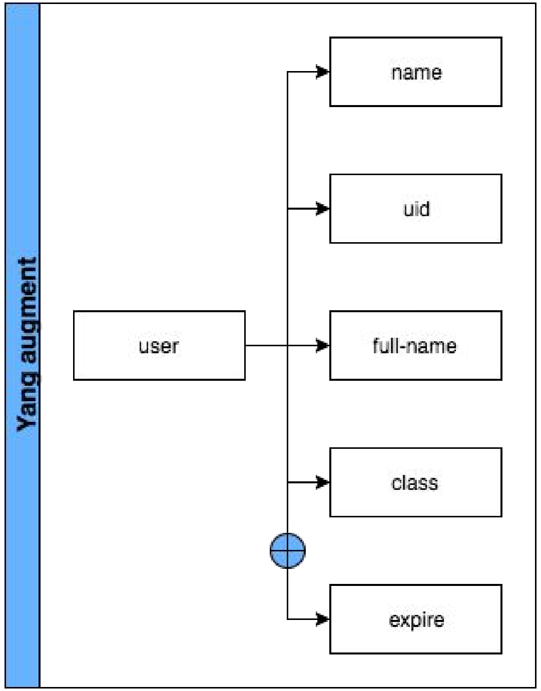


```
<change>/ex:system/ex:services/ex:ssh/ex:port</change>
<change>/ex:system/ex:user[ex:name='fred']/ex:type</change>
<change>/ex:system/ex:server[ex:ip='192.0.2.1'][ex:port='80’]</change>
```

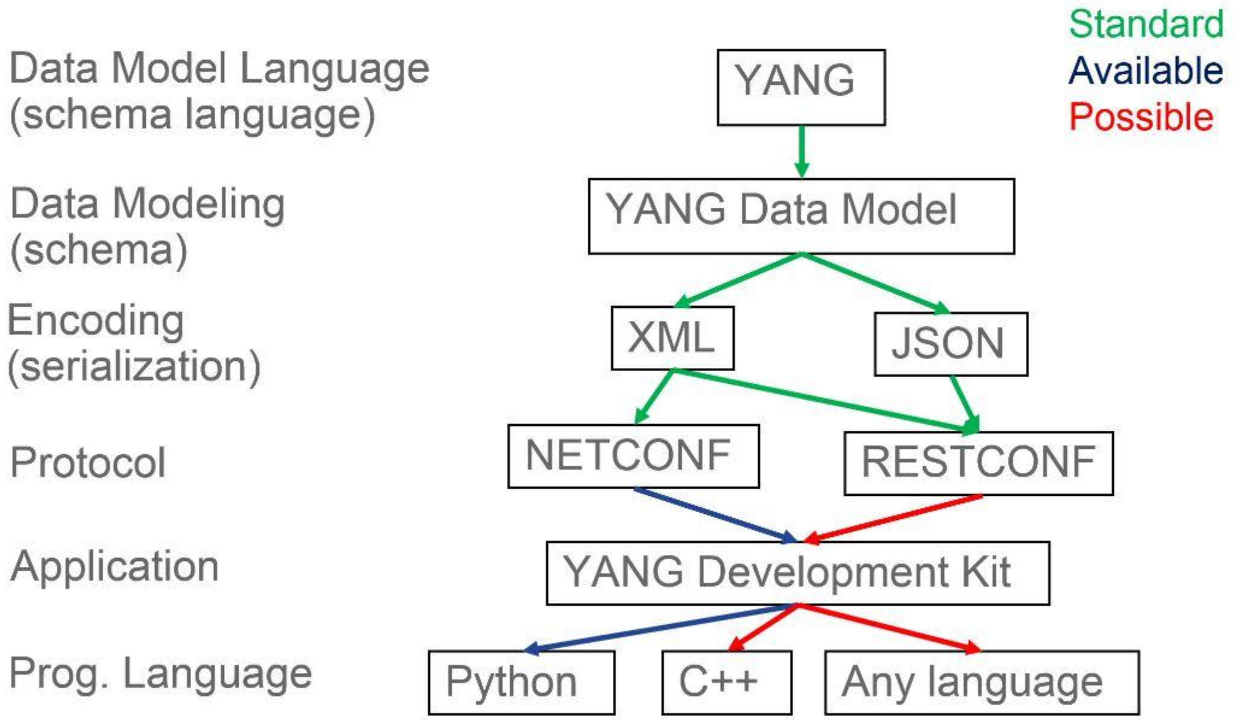
Yang augment statement

```
list user {
  key name;
  leaf name {
    type string;
  }
  leaf uid {
    type uint32;
  }
leaf full-name {
    type string;
  }
  leaf class {
    type string;
    default viewer;
  }
}
```

```
augment /sys:system/sys:user {
  leaf expire {
    type ietf-yang-types:date-and-time;
  }
}
```



All protocols interaction & standardization



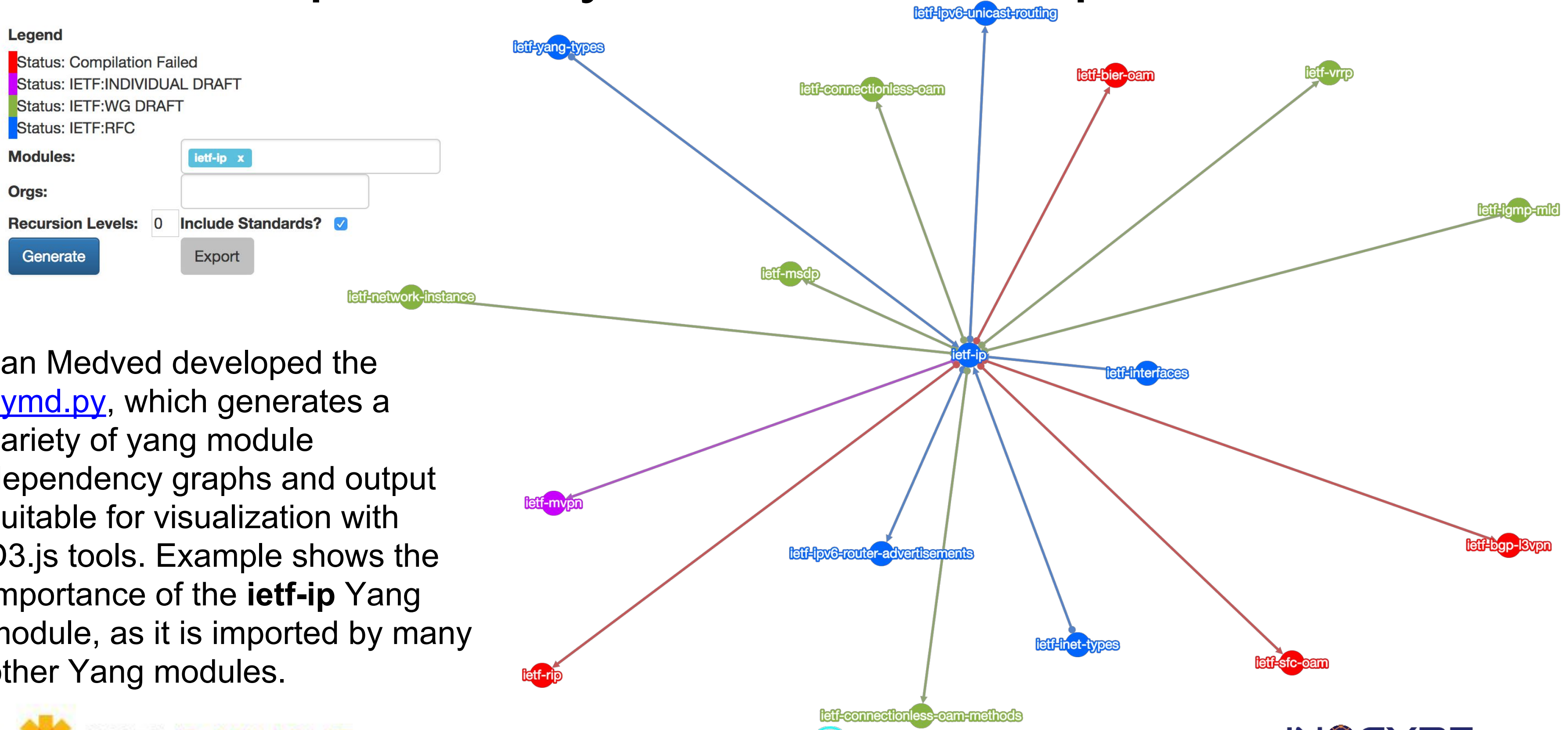
* Source - Benoît Claise blog:
http://www.claise.be/2016/12/yang-opensource-tools-for-data-modeling-driven-management/

Extracting Yang modules out from RFCs

- To extract the Yang module(s) out of IETF draft for the validation, Jan Medved created the xym.py, the eXtracting Yang Module Python script for IETF drafts and RFCs.

- pyang is Yang validator, transformator and code generator,written in python, lead by Martin Bjorklund, and constantly improved by the community , especially now that the Yang 1.1 specifications [RFC 7950] have published. At the IETF, pyang is now part of the submission tool (thanks to Qin Wu and Dapeng Liu): when posting an IETF draft containing a Yang module, the Yang module language is validated.

Visual dependency tool – f.e. ietf-ip


Legend

🟥 Status: Compilation Failed
🟪 Status: IETF:INDIVIDUAL DRAFT
🟩 Status: IETF:WG DRAFT
🟦 Status: IETF:RFC

Modules: ietf-ip x

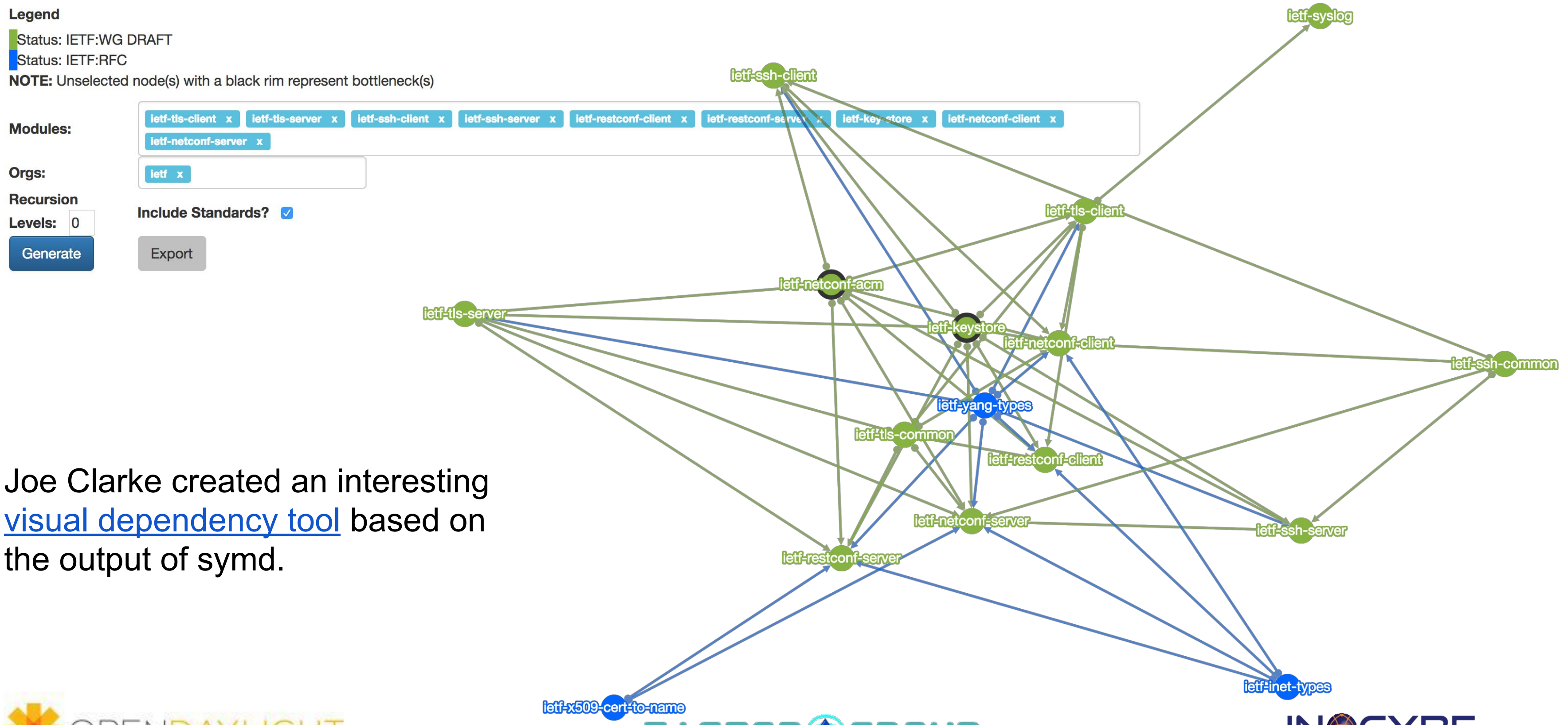
Orgs:

Recursion Levels: 0 **Include Standards?** ✓

Jan Medved developed the symd.py, which generates a variety of yang module dependency graphs and output suitable for visualization with D3.js tools. Example shows the importance of the **ietf-ip** Yang module, as it is imported by many other Yang modules.

OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

Yang - Impact analysis, bottlenecks.


Legend

🟩 Status: IETF:WG DRAFT
🟦 Status: IETF:RFC
NOTE: Unselected node(s) with a black rim represent bottleneck(s)

Modules: ietf-tls-client x, ietf-tls-server x, ietf-ssh-client x, ietf-ssh-server x, ietf-restconf-client x, ietf-restconf-server x, ietf-keystore x, ietf-netconf-client x, ietf-netconf-server x

Orgs: ietf x

Recursion Levels: 0

Include Standards? ✓

Joe Clarke created an interesting visual dependency tool based on the output of symd.

OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

Yang Catalog

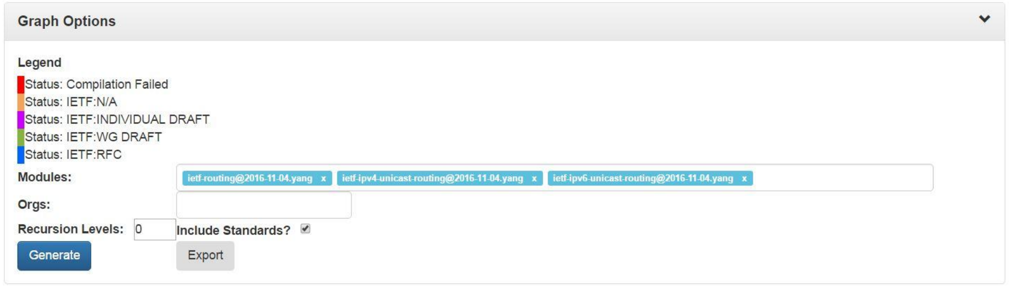
- A Yang model catalog and registry that allows users to find models relevant to their use cases from the large and growing number of Yang modules being published.
- This server is running:
  - A NETCONF and REST (not RESTCONF-compliant yet) server loaded with the Yang module from draft-openconfig-netmod-model-catalog. It currently only allows public read access to the content. Feel free to reach out through the github forum if you are interested in write access. The username is oper and the password is oper.
  - A Yang Validator, a web frontend that allows for validation of Yang modules and IETF drafts.
  - A Yang Search, a web frontend that allows for searches over the content of the module catalog.
  - A Yang impact analysis tool.
  - A Yang Explorer that includes a Yang browser and RPC-builder application to experiment with Yang modules

Yang Catalog - Yang DB search

The Yang DB search laid a framework for the multi-SDO impact analysis, including a color scheme for the standard maturity levels.

Currently it includes:

- IETF
- BBF
- OpenConfig



Graph Options

Legend

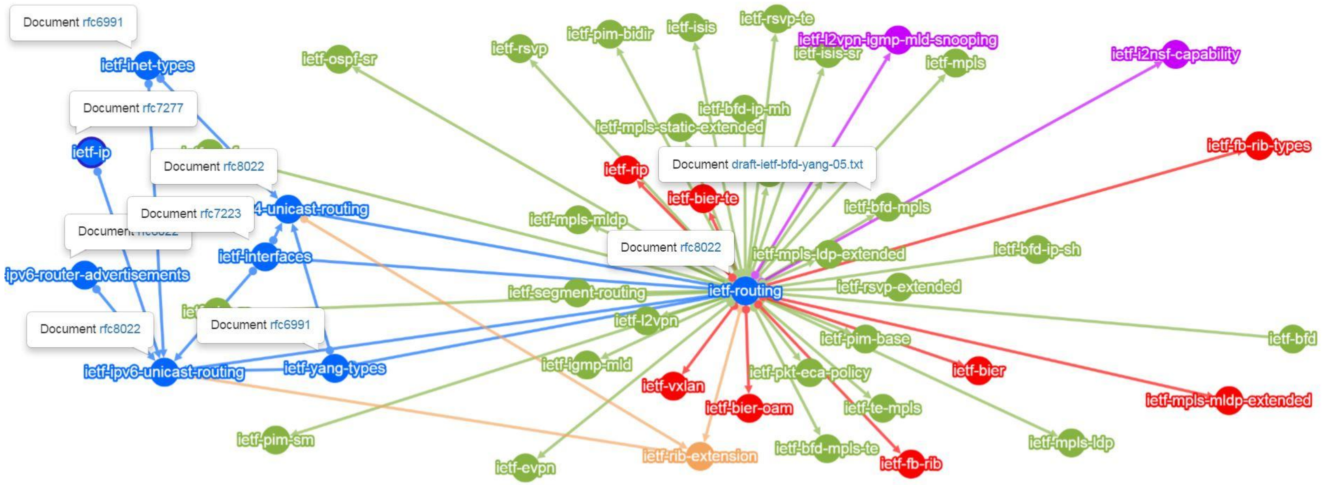
🟥 Status: Compilation Failed
🟧 Status: IETF:N/A
🟪 Status: IETF:INDIVIDUAL DRAFT
🟩 Status: IETF:WG DRAFT
🟦 Status: IETF:RFC

Modules: ietf-routing@2016-11-04.yang x ietf-ipv4-unicast-routing@2016-11-04.yang x ietf-ipv6-unicast-routing@2016-11-04.yang x

Orgs:

Recursion Levels: 0

Include Standards? ✓

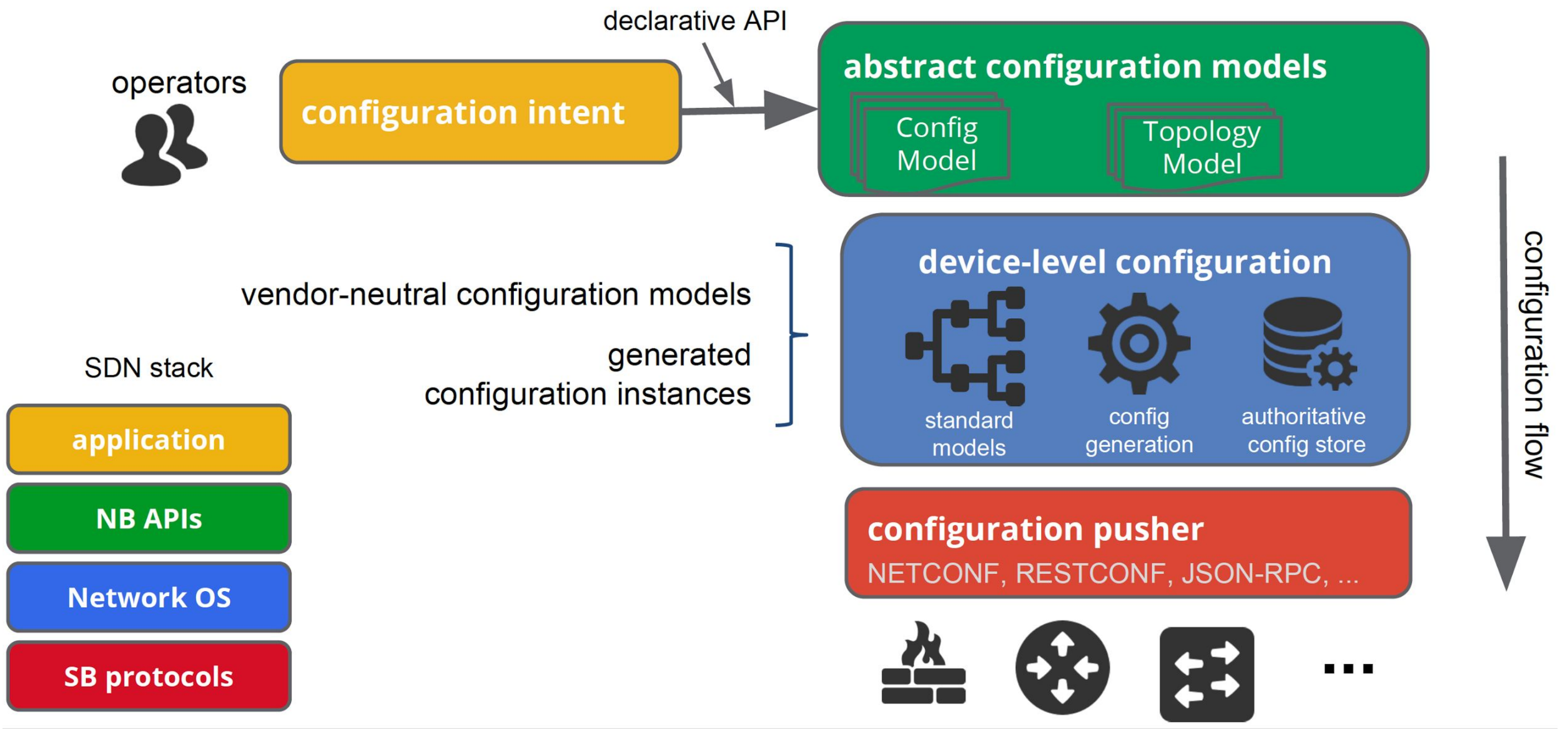


OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

OpenConfig

- Informal industry collaboration of network operators
- Focus: define vendor-neutral configuration and operational state models based on real operations
- Adopted Yang data modeling language
- Participants: Apple, AT&T, BT, Comcast, Cox, Facebook, Google, Level3, Microsoft, Verizon, Yahoo!
- Primary output is model code, published as open source via public github repo
- Ongoing interactions with standards and open source communities (e.g., IETF, ONF, ODL, ONOS)

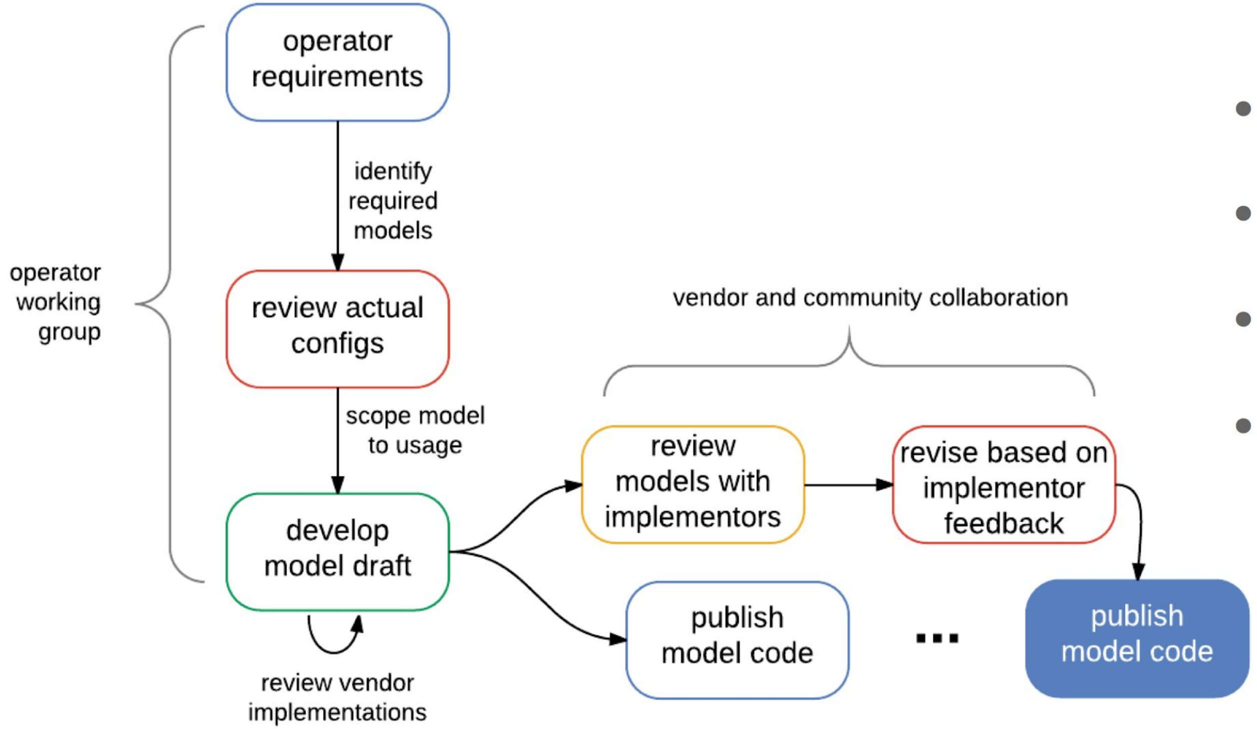
OpenConfig - Example configuration pipeline



*Source - Joshua George, Anees Shaikh - Google Network Operations

OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

Current OpenConfig "process"



- initial models developed by OpenConfig
- extensive collaboration with vendors
- leverage existing work where possible
- publish models and docs

*Source - Joshua George, Anees Shaikh - Google Network Operations

Support for OpenConfig models in ODL

Configuration and monitoring APIs based on OpenConfig models

- BGP and routing policy
  - interface to ODL BGP implementation
  - some progress underway (e.g., IETF 93 hackathon)
- MPLS / TE
  - integration with PCEP, segment routing

Agenda

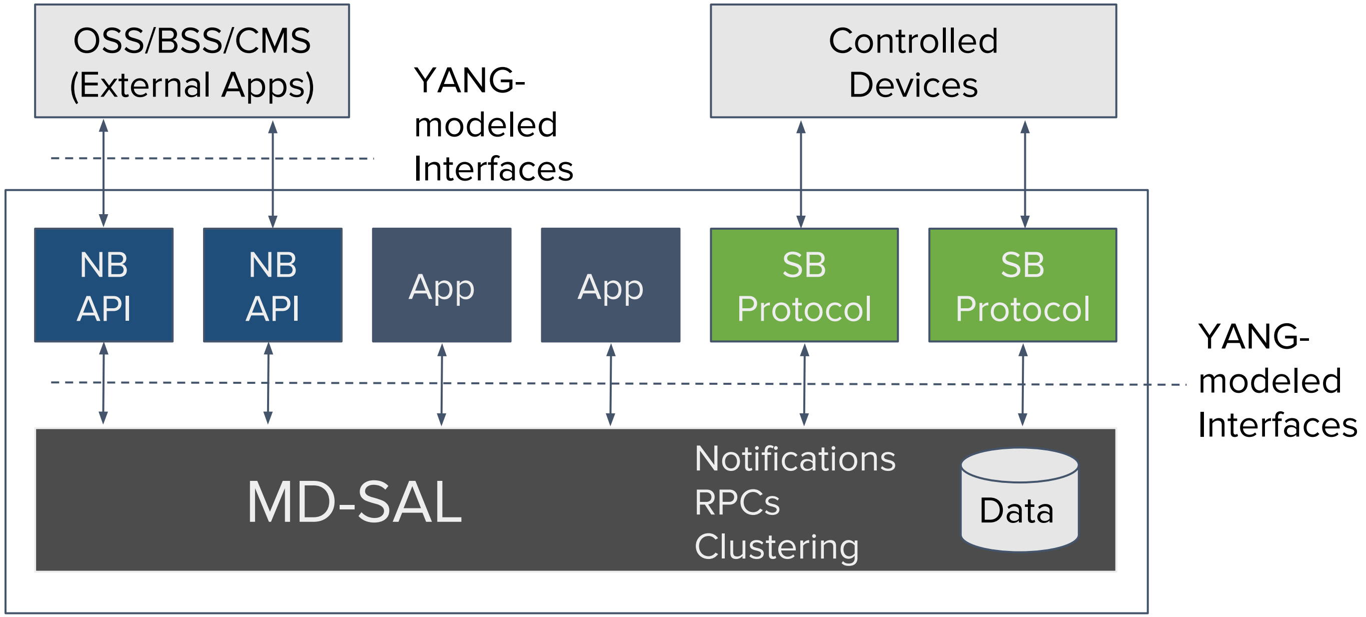
- Orchestration problem statement
- Netconf overview
- Restconf overview
- Yang overview
- **OpenDaylight overview**
- DEMO

What is OpenDaylight?

- Open Source programmable platform hosted by the Linux Foundation
- ~4 Years Old
- Mature, Open Governance
- Mature code base
- ~1000 Individual Contributors from ~140 organizations
- Dozens of OpenDaylight-based solutions
- Over 100 deployments

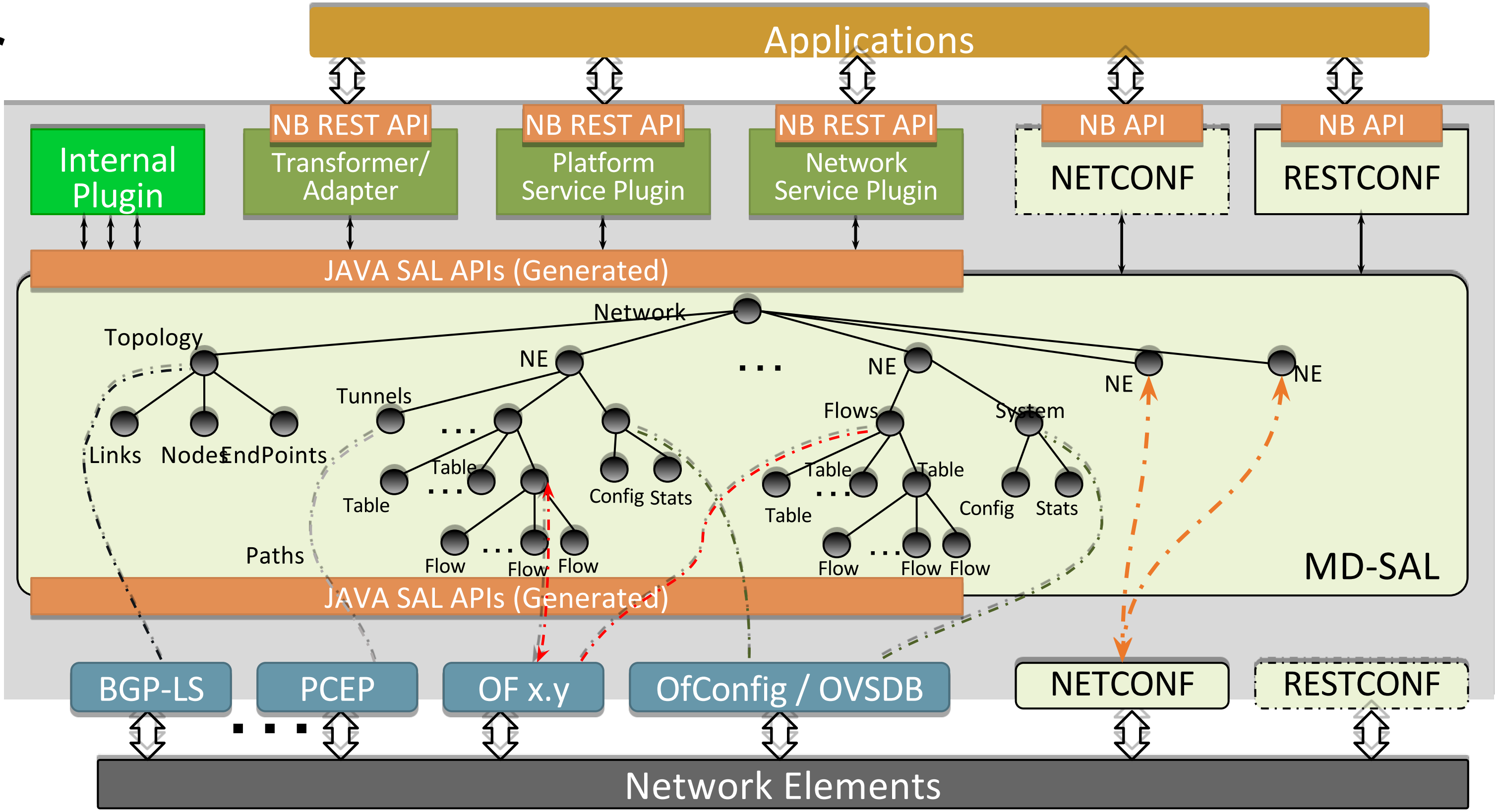
OpenDaylight: a Yang-based microservices Platform

- Based on Model-Driven Service Abstraction Layer (MD-SAL)
- Yang
- Data Modeling Language for NETCONF
- Creates well-defined APIs
- Java and RESTCONF APIs auto-generated from Yang

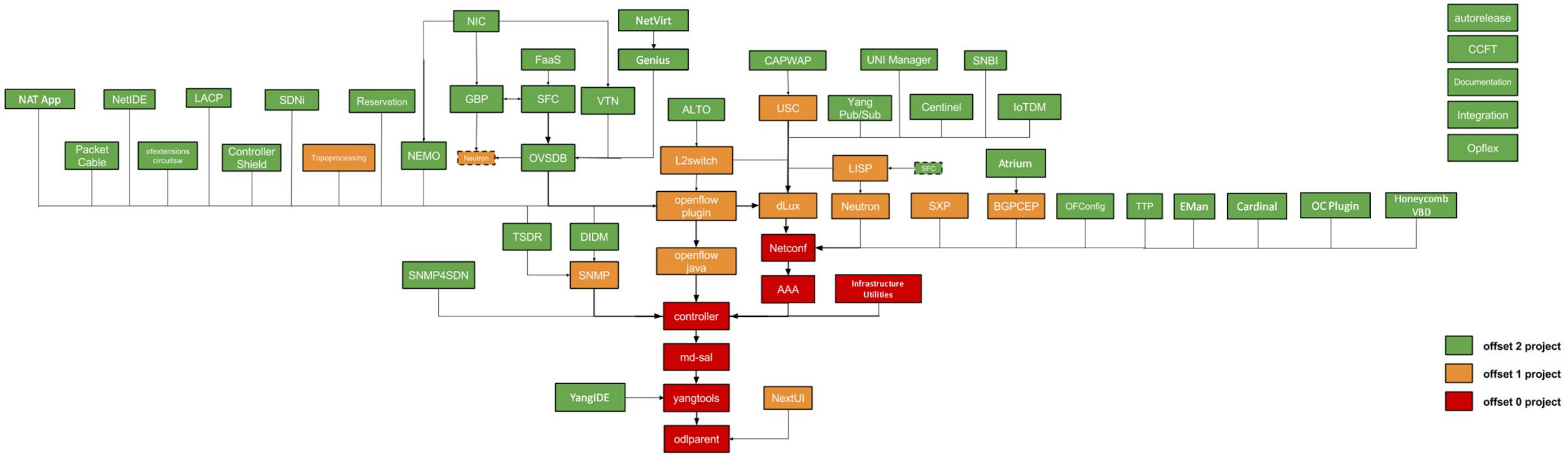


OPENDAYLIGHT

MD-SAL = Model-Driven Service Abstraction Layer

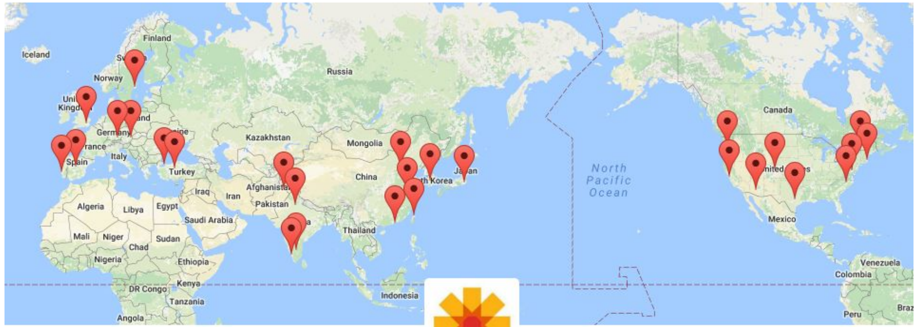


Opendaylight Boron Project Dependencies



The Sun Never Sets on OpenDaylight

- 3580 members of OpenDaylight User Group
  - 27 Meetup Cities
  - Top 5 communities: SF Bay Area, Bengaluru, Delhi, London, Tokyo
- 1782 people in China QQ group
- India Forum – 2 Days 300+ attendees
- > 700 people have taken the LF online ODL course ($150/person) in first 2 months!
- Coming in 2017:
  - OpenDaylight Days



OpenDaylight Meetups

We are **3580** members across **27** Meetups

OpenDaylight is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multi-vendor networks. It provides a model-driven service abstraction platform that allows users to write apps that easily work across a wide variety of hardware and southbound protocols.

OpenDaylight User Groups (ODLUGs) are regional, self-organized, informal associations that meet globally to discuss OpenDaylight. The participants in these groups share knowledge, recruit and onboard new developers, discuss best practices and technical challenges, as well as create awareness. ODLUG hold meetups, run small hackathons, and host social events. While ODLUG do not have a formal relationship with the OpenDaylight Project, they are a key part of the vast and growing ecosystem.

https://www.opendaylight.org/ODLUG

OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

Join ODL Russia Meetup Group !

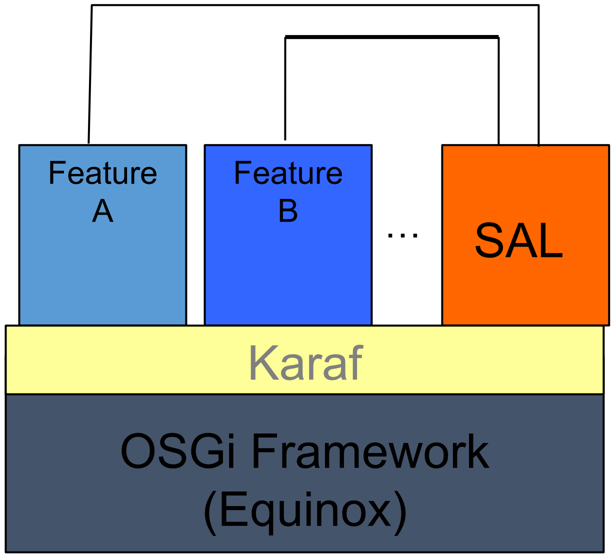
Assign to group, we will meet soon!

https://www.meetup.com/OpenDaylight-Russia/

OPENDAYLIGHT FACTOR GROUP INOCYBE TECHNOLOGIES

Java, Interface, Maven, OSGi, Karaf

- Java chosen as an enterprise-grade, cross-platform compatible language
- Java Interfaces are used for event listening, specifications and forming patterns
- Maven – build system for Java
- OSGi:
  - Allows dynamically loading bundles
  - Allows registering dependencies and services exported
  - For exchanging information across bundles
- Karaf: Light-weight Runtime for loading modules/bundles
  - OSGi based. Primary distribution mechanism for Helium

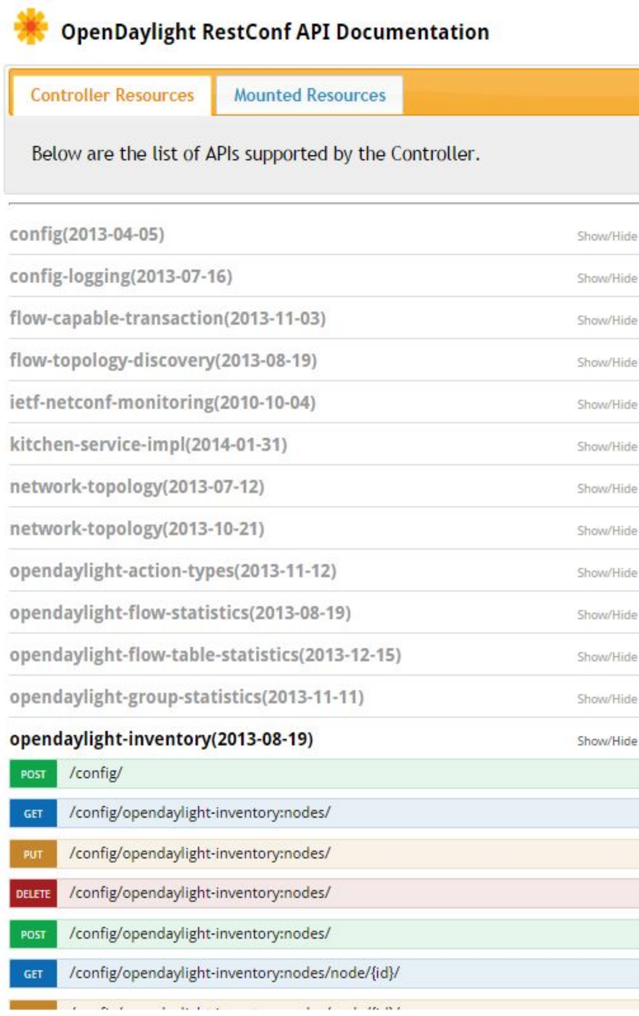


REST APIs

- OpenDaylight has significant support for REST APIs
- Restconf allows for checking config and operational state
  - feature:install odl-restconf
  - http://localhost:8181/restconf/....
- List of exposed Northbound APIs are auto-generated using swagger.
  - feature:install odl-mdsal-apidocs
  - http://localhost:8181/apidoc/explorer

OPENDAYLIGHT

FACTOR GROUP


OpenDaylight RestConf API Documentation

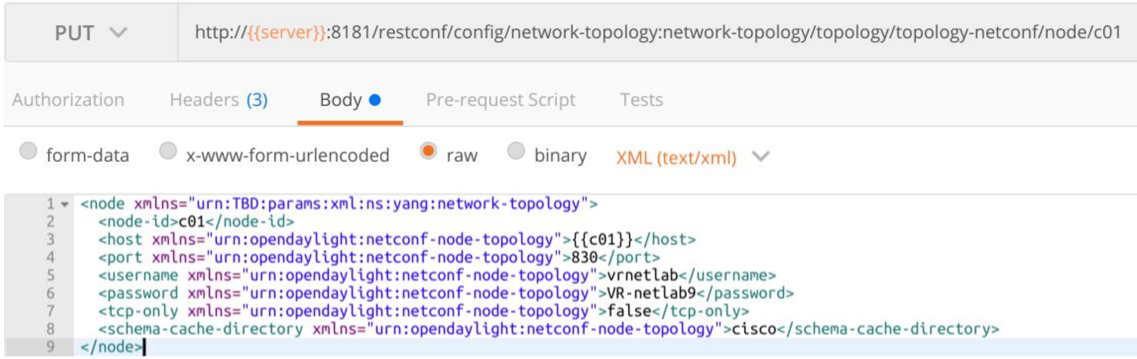
Controller Resources | Mounted Resources

Below are the list of APIs supported by the Controller.

config(2013-04-05)	Show/Hide
config-logging(2013-07-16)	Show/Hide
flow-capable-transaction(2013-11-03)	Show/Hide
flow-topology-discovery(2013-08-19)	Show/Hide
ietf-netconf-monitoring(2010-10-04)	Show/Hide
kitchen-service-impl(2014-01-31)	Show/Hide
network-topology(2013-07-12)	Show/Hide
network-topology(2013-10-21)	Show/Hide
opendaylight-action-types(2013-11-12)	Show/Hide
opendaylight-flow-statistics(2013-08-19)	Show/Hide
opendaylight-flow-table-statistics(2013-12-15)	Show/Hide
opendaylight-group-statistics(2013-11-11)	Show/Hide
opendaylight-inventory(2013-08-19)	Show/Hide
POST	/config/
GET	/config/opendaylight-inventory:nodes/
PUT	/config/opendaylight-inventory:nodes/
DELETE	/config/opendaylight-inventory:nodes/
POST	/config/opendaylight-inventory:nodes/
GET	/config/opendaylight-inventory:nodes/node/{id}/

NETCONF connector

- To connect the NETCONF/YANG NE to ODL - use network-topology feature:
  - feature:install odl-netconf-topology odl-restconf
- Fully capable (full support for Netconf/Yang) device to mount, we need only:
  - Name
  - Host
  - Port
  - Username/Password (for Netconf session)

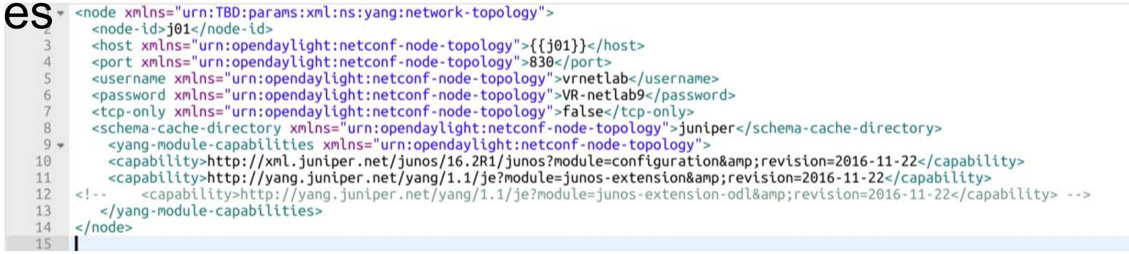


```
http://{{server}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/c01
```

```
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
  <node-id>c01</node-id>
  <host xmlns="urn:opendaylight:netconf-node-topology">{{c01}}</host>
  <port xmlns="urn:opendaylight:netconf-node-topology">830</port>
  <username xmlns="urn:opendaylight:netconf-node-topology">vrnetlab</username>
  <password xmlns="urn:opendaylight:netconf-node-topology">VR-netlab9</password>
  <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
  <schema-cache-directory xmlns="urn:opendaylight:netconf-node-topology">cisco</schema-cache-directory>
</node>
```

NETCONF connector (advanced config)

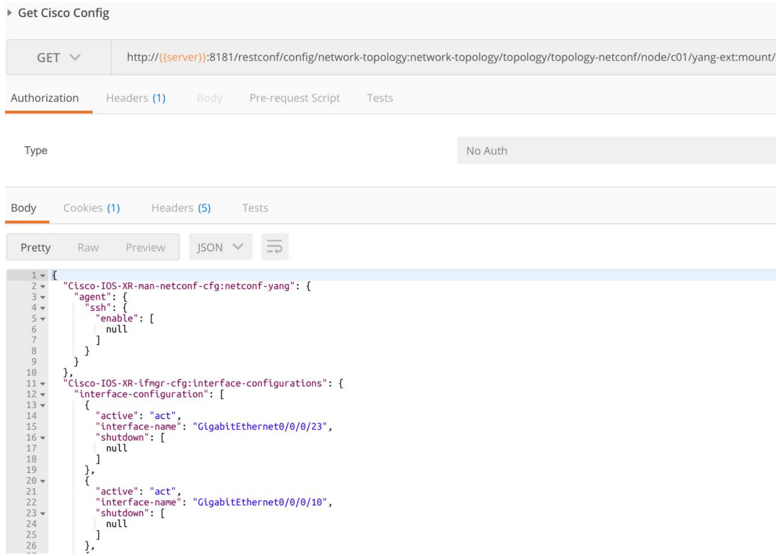
- Advanced configuration
  - Schema-cache-directory
  - Reconnect-on-changed-schema
  - Connection-timeout-millis
  - Default-request-timeout-millis
  - Max-connection-attempts
  - Between-attempts-timeout-millis
  - Sleep-factor
  - Keepalive-delay
  - Yang-module-capabilities
  - Yang library
  - Concurrent rpc limit



```xml
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
  <node-id>j01</node-id>
  <host xmlns="urn:opendaylight:netconf-node-topology">{{j01}}</host>
  <port xmlns="urn:opendaylight:netconf-node-topology">830</port>
  <username xmlns="urn:opendaylight:netconf-node-topology">vrnetlab</username>
  <password xmlns="urn:opendaylight:netconf-node-topology">VR-netlab9</password>
  <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
  <schema-cache-directory xmlns="urn:opendaylight:netconf-node-topology">juniper</schema-cache-directory>
    <yang-module-capabilities xmlns="urn:opendaylight:netconf-node-topology">
    <capability>http://xml.juniper.net/junos/16.2R1/junos?module=configuration&amp;revision=2016-11-22</capability>
    <capability>http://yang.juniper.net/yang/1.1/je?module=junos-extension&amp;revision=2016-11-22</capability>
<!--    <capability>http://yang.juniper.net/yang/1.1/je?module=junos-extension-odl&amp;revision=2016-11-22</capability> -->
   </yang-module-capabilities>
</node>
```

NETCONF connector - what available

- Get datastore
  - yang-ext:mount
- Invoke RPC (POST)
  - yang-ext:/mount/<module>:<operation>
- Update/Delete a netconf-connector
- Get/Set/Modify mounted NEs configuration
- Get the operational datastore
- Receive notifications from NETCONF NE


Get Cisco Config

GET http://{{server}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/c01/yang-ext:mount/

Type: No Auth

```
{
  "Cisco-IOS-XR-man-netconf-cfg:netconf-yang": {
    "agent": {
      "ssh": {
        "enable": [
          null
        ]
      }
    }
  },
  "Cisco-IOS-XR-ifmgr-cfg:interface-configurations": {
    "interface-configuration": [
      {
        "active": "act",
        "interface-name": "GigabitEthernet0/0/0/23",
        "shutdown": [
          null
        ]
      },
      {
        "active": "act",
        "interface-name": "GigabitEthernet0/0/0/10",
        "shutdown": [
          null
        ]
      },
```

Agenda

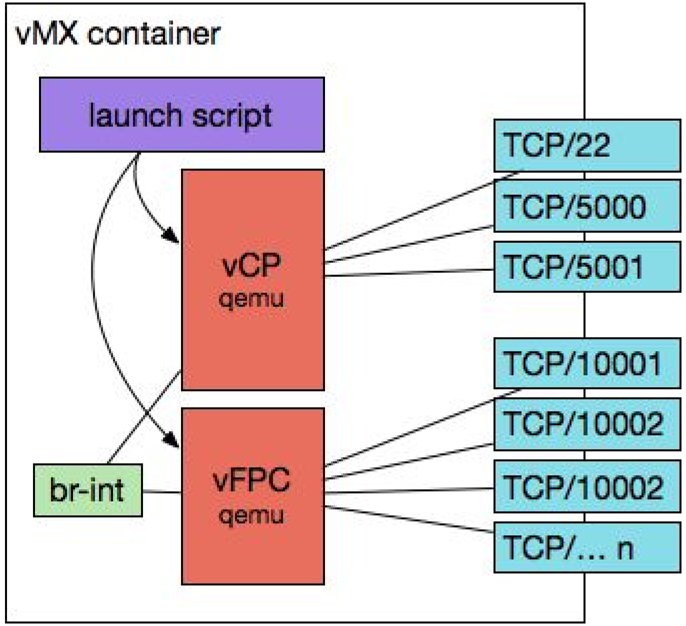
- Orchestration problem statement
- Netconf overview
- Restconf overview
- Yang overview
- OpenDaylight overview
- **DEMO**

vrnetlab – tool for service models CI/CT/CD

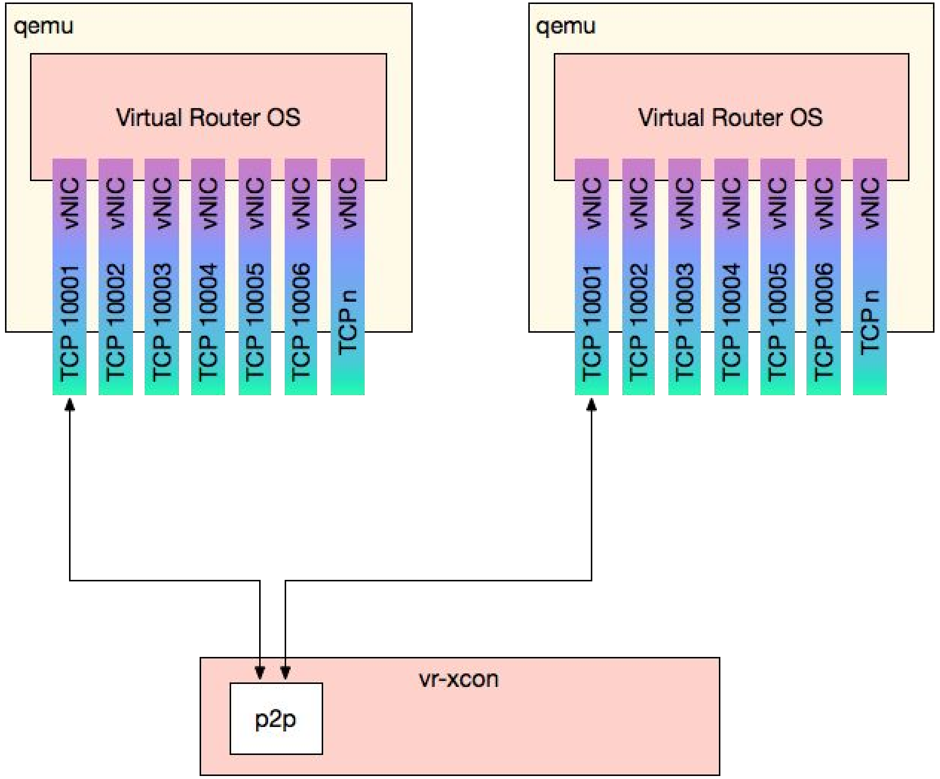
- Run your favourite virtual routers in docker for convenient labbing, development and testing.
- vrnetlab is being developed for the TeraStream project at Deutsche Telekom as part of an automated CI/CT/CD environment for testing our network provisioning system.
- It supports: Arista vEOS, Cisco CSR1000v,Cisco Nexus NX-OS, Cisco XRv, Juniper vMX, Nokia VSR
- Features:
  - Use docker and KVM
  - Ship as single unit
  - Bootstrap
  - Flexible networking
  - Simple: docker run —privileged -d vr-xrv:5.3.3

vrnetlab – tool for service models CI/CT/CD

- vMX runs two VMS:
- control plane
- forwarding plane
- vMX startup scripts is a mess
- vrnetlab makes it simple!



vrnetlab – tool for service models CI/CT/CD



NETCONF RFC Overview

- RFC 3535 Informational: Background
- RFC 6244 NETCONF+Yang Architectural Overview
- RFC 6241 Base NETCONF Protocol
- RFC 6242, 5539 Transport Mappings
- RFC 5277 Notifications
- RFC 5717 Partial Locking
- RFC 6022 NETCONF Monitoring
- RFC 6243 With defaults
- RFC 6470 Base Notifications
- RFC 6536 NETCONF Access Control Model

  – https://datatracker.ietf.org/wg/netconf/charter/
  – www.rfc-editor.org/rfc/rfcXXXX.txt

RESTCONF RFC Overview

- RFC 8040 - RESTCONF

Yang RFC Overview

- RFC 6020/7950 YANG Base Specification
- RFC 6087 Guidelines for YANG Authors and Reviewers
- RFC 6110 Mapping YANG and Validating NETCONF Content
- RFC 6244 NETCONF+Yang Architectural Overview
- RFC 6643 Translation of SMIv2 MIBs to YANG
- RFC 6991 Common Yang Data Types
- RFC 7223 YANG Module for Interface Management
- RFC 7224 YANG Module for Interface Types
- RFC 6022 YANG Module for NETCONF Monitoring

  - https://datatracker.ietf.org/wg/netmod/charter/
  - https://www.ietf.org/iesg/directorate/yang-doctors.html
  - http://www.yang-central.org/

Thank You

Evgeny Zobnitsev
e-mail: e@zobnitsev.ru
twitter: @ezobn