



## Catalyst 9800 Programmability and Telemetry Deployment Guide 16.10

[Feature Overview](#) 2

[Configuration Overview](#) 2

[NETCONF](#) 2

[RESTCONF \(BETA, not supported\)](#) 5

[gRPC and gNMI \(BETA, not supported\)](#) 6

[YANG Data Models](#) 7

[YangExplorer Tool](#) 10

[Ansible Automation](#) 15

[Telemetry](#) 20

[Conclusion](#) 23

[Additional Resources](#) 23

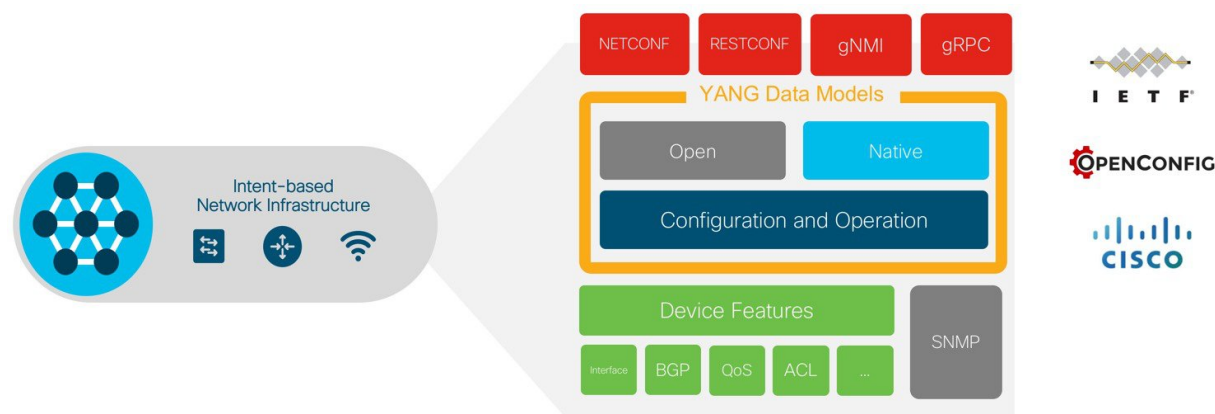
[Reference](#) 24

[Questions?](#) 25

Revised: December 17, 2024

## Feature Overview

The Catalyst 9800 IOS-XE based Wireless LAN Controller has several options for programmatic configuration. Traditional methods for configuring the WLC include the CLI or WebUI, but that has now been expanded to include the programmatic interfaces. These programmatic interfaces include NETCONF, RESTCONF, and the gNMI/gRPC protocols. YANG data models define what data is accessible over the programmatic interfaces, and they come in several varieties. Cisco IOS-XE features are defined within the native data models, while standard and vendor agnostic features are defined within the open data models. Either models can be used for many tasks, however features specific to IOS-XE are only available in the native models.



## Configuration Overview

Regardless of the interfaces used, the following configuration is required on the Catalyst 9800:

```
configure terminal
aaa new-model
aaa authentication login default local
aaa authorization exec default local
aaa session-id common
exit
write memory
```

Additionally, a user account is required. Create a new user account with username ‘netconf’ and password ‘netconf’ with the following command

```
username netconf privilege 15 password 0 netconf
username restconf privilege 15 password 0 restconf
```

Alternately the existing or pre-configure ‘admin’ user and password can be used as the dedicated NETCONF or RESTCONF user account is not required.

AAA authentication login for NETCONF is supported only with the default method.

## NETCONF

NETCONF is a protocol defined by the IETF to “install, manipulate, and delete the configuration of network devices”. NETCONF operations are realized on top of a Remote Procedure Call (RPC) layer using an XML encoding and provides a basic set of operations to edit and query configuration on a network device.

## NETCONF configuration

To enable the NETCONF interface enter the following commands:

```
configure terminal
netconf-yang
exit
```

## NETCONF YANG Models

The NETCONF capabilities exchange can be retrieved by connecting to the device on the default port 830. The capabilities exchange lists all available YANG data models supported by the device. Using tools like YangExplore and pYANG which are detailed in further sections, these YANG modules can be downloaded from the device and analyzed further.


```
user@canyon-server:~$ ssh netconf@10.10.10.223 -p 830
netconf@10.10.10.223's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
```

## Verifying NETCONF status

To verify the NETCONF interface is operational, run the command: **show platform software yang-management process**. Ensure the NETCONF SSHD “ncsshd” process is running

```
jcohoe-vewlc-2#show platform software yang-management process
confd           : Running
nesd            : Running
syncfd          : Running
ncsshd          : Running
dmiauthd        : Running
nginx           : Running
ndbmand         : Running
pubd            : Running
gnmib           : Not Running

jcohoe-vewlc-2#
```



## NETCONF XML RPC payloads examples

The XML RPC payloads below can be generated, modified, and sent from within the YangExplorer tool, or any other tool that can send XML payloads over the NETCONF interface, such as a python script. Example WLC creation, verification, and removal XML payloads are listed below. These can be used to quickly and easily create, verify, and remove a WLAN over the NETCONF interface.

### WLAN Creation

The XML RPC below can be sent over the NETCONF interface to create a WLAN with the defined parameters. In this case, the WLAN ID is 4 and the SSID is “open”.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
<edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <target>
    <running/>
  </target>
  <test-option>test-then-set</test-option>
  <error-option>rollback-on-error</error-option>
  <config>
    <wlan-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-wlan-cfg">
      <wlan-cfg-entries>
        <wlan-cfg-entry>
          <profile-name>open</profile-name>
          <vap-id>4</vap-id>
          <security-wifi-sec>false</security-wifi-sec>
          <rsn-ie-enabled>false</rsn-ie-enabled>
          <rsn-cipher-suite-aes>false</rsn-cipher-suite-aes>
          <auth-key-mgmt-suite8021x>false</auth-key-mgmt-suite8021x>
          <client-session-timeout>1800</client-session-timeout>
          <id-data>
            <profile-name>open</profile-name>
            <ssid>open</ssid>
            <status>true</status>
          </id-data>
          <is-remote-lan>false</is-remote-lan>
        </wlan-cfg-entry>
      </wlan-cfg-entries>
    </wlan-cfg-data>
  </config></edit-config></rpc>
```

### WLAN Verification

The XML RPC below can be sent over the NETCONF interface to verify a WLAN with ID 4.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <wlan-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-wlan-cfg">
        <wlan-cfg-entries>
          <wlan-cfg-entry>
            <vap-id>4</vap-id>
          </wlan-cfg-entry>
        </wlan-cfg-entries>
      </wlan-cfg-data>
    </filter>
  </get>
</rpc>
```

## WLAN Removal

The XML RPC below can be sent over the NETCONF interface to remove a WLAN with ID 4.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <wlan-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-wlan-cfg">
        <wlan-cfg-entries xc:operation="delete">
          <wlan-cfg-entry>
            <vap-id>4</vap-id>
          </wlan-cfg-entry>
        </wlan-cfg-entries>
      </wlan-cfg-data>
    </config>
  </edit-config>
</rpc>
```

## RESTCONF (BETA, not supported)

RESTCONF uses structured XML or JSON and YANG data models to provide a REST-like API which enables programmatic access to the network device. RESTCONF API uses HTTP's method. The Catalyst 9800 IOS XE implementation supports the following RESTCONF operations: GET, PATCH, PUT, POST, DELETE, HEAD.

### RESTCONF configuration

To enable the RESTCONF interface enter the following commands:

```
configure terminal
ip http secure-server
restconf
exit
```

### RESTCONF YANG Models

The list of supported YANG data models can be retrieved from the RESTCONF interface by sending GET request to the URI `restconf/data?fields=ietf-yang-library:modules-state/module`

For example:

```
curl -k -u username:password https://10.10.10.223/restconf/data?fields=ietf-yang-library:modules-state/module
```

```
user@canyon-server:~$ curl -k -u jcohoe:jcohoe https://10.10.10.223/restconf/data?fields=ietf-yang-library:modules-state/module
<data xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
    <module>
      <name>ATM-FORUM-TC-MIB</name>
      <revision></revision>
      <schema>https://10.10.10.223:443/restconf/tailf/modules/ATM-FORUM-TC-MIB</schema>
      <namespace>urn:ietf:params:xml:ns:yang:smiv2:ATM-FORUM-TC-MIB</namespace>
      <conformance-type>import</conformance-type>
    </module>
```

```
<module>
  <name>Cisco-IOS-XE-w
  <revision>2018-06-28
  <schema>https://10.1
  <namespace>http://ci
  <conformance-type>imp
</module>
<module>
  <name>Cisco-IOS-XE-w
  <revision>2018-07-09
  <schema>https://10.1
  <namespace>http://ci
  <conformance-type>imp
</module>
```

A specific YANG module can be downloaded by sending another GET request to the URI containing the model. For example to download the Cisco-IOS-XE-wireless-access-point-oper model send the GET request to /restconf/tailf/modules/Cisco-IOS-XE-wireless-access-point-oper/2018-06-28 as seen in the screenshot below:

```
curl -s -k -u jcohoe:jcohoe https://10.10.10.223/restconf/tailf/modules/Cisco-IOS-XE-wireless-access-point-oper/2018-06-28 -o Cisco-IOS-XE-wireless-access-point-oper.yang
user@canyon-server:~$ pyang -f tree Cisco-IOS-XE-wireless-access-point-oper.yang --tree-depth 2
Cisco-IOS-XE-wireless-access-point-oper.yang:6: error: module "Cisco-IOS-XE-wireless-ap-types" not found in search path
Cisco-IOS-XE-wireless-access-point-oper.yang:9: error: module "Cisco-IOS-XE-wireless-enum-types" not found in search path
Cisco-IOS-XE-wireless-access-point-oper.yang:12: error: module "Cisco-IOS-XE-wireless-types" not found in search path
Cisco-IOS-XE-wireless-access-point-oper.yang:6032 (at Cisco-IOS-XE-wireless-access-point-oper.yang:2641): error: the key "wtp-mac" does not reference an existing leaf
Cisco-IOS-XE-wireless-access-point-oper.yang:6016: error: the key "wtp-mac" does not reference an existing leaf
Cisco-IOS-XE-wireless-access-point-oper.yang:6048: error: the key "radio-mac" does not reference an existing leaf
Cisco-IOS-XE-wireless-access-point-oper.yang:6055: error: the key "radio-mac" does not reference an existing leaf
Cisco-IOS-XE-wireless-access-point-oper.yang:6062: error: the key "radio-mac" does not reference an existing leaf
module: Cisco-IOS-XE-wireless-access-point-oper
  +--ro access-point-oper-data
    +--ro ap-history* [ethernet-mac]
    |   ...
    +--ro wireless-interface-data!
    |   ...
    +--ro radio-oper-data* [wtp-mac radio-slot-id]
    |   ...
    +--ro qos-client-data* [client-mac]
    |   ...
    +--ro capwap-data* [wtp-mac]
    |   ...
    +--ro ap-name-mac-map* [wtp-name]
```

In the step above the ‘pyang’ tool is used to details some data is contained within the YANG data model. In this example ap-history, wireless-interface-data, etc, are sections within the "access-point-oper-data" container.

## Verifying RESTCONF status

To verify the RESTCONF interface is operational, run the command: **show platform software yang-management process**. Ensure the “nginx” process is running.

```
jcohoe-vewlc-2#show platform software yang-management process
confd           : Running
nesd            : Running
syncfd         : Running
ncsshd         : Running
dmiauthd       : Running
nginx           : Running
ndbmana        : Running
pubd           : Running
gnmib          : Not Running
jcohoe-vewlc-2#
```

## gRPC and gNMI (BETA, not supported)

gNMI is gRPC Network Management Interface developed by Google. gNMI provides the mechanism to install, manipulate, and delete the configuration of network devices, and also to view operational data. The content provided through gNMI can be modeled using YANG. gRPC is a remote procedure call developed by Google for low-latency, scalable distributions with mobile clients communicating to a cloud server. gRPC carries gNMI, and provides the means to formulate and transmit data and operation requests.

## gNMI configuration

To enable the gNMI interface enter the following commands:



```
configure terminal
gnmi-yang
gnmi-yang server
gnmi-yang port 50052
exit
write memory
```

## gNMI Operations

The following operations are supported over the gNMI interface:

gNMI GetRequest

gNMI SetRequest

Additional details on utilizing this interface are available from the Programmability Configuration Guide linked in the Additional Resources section.

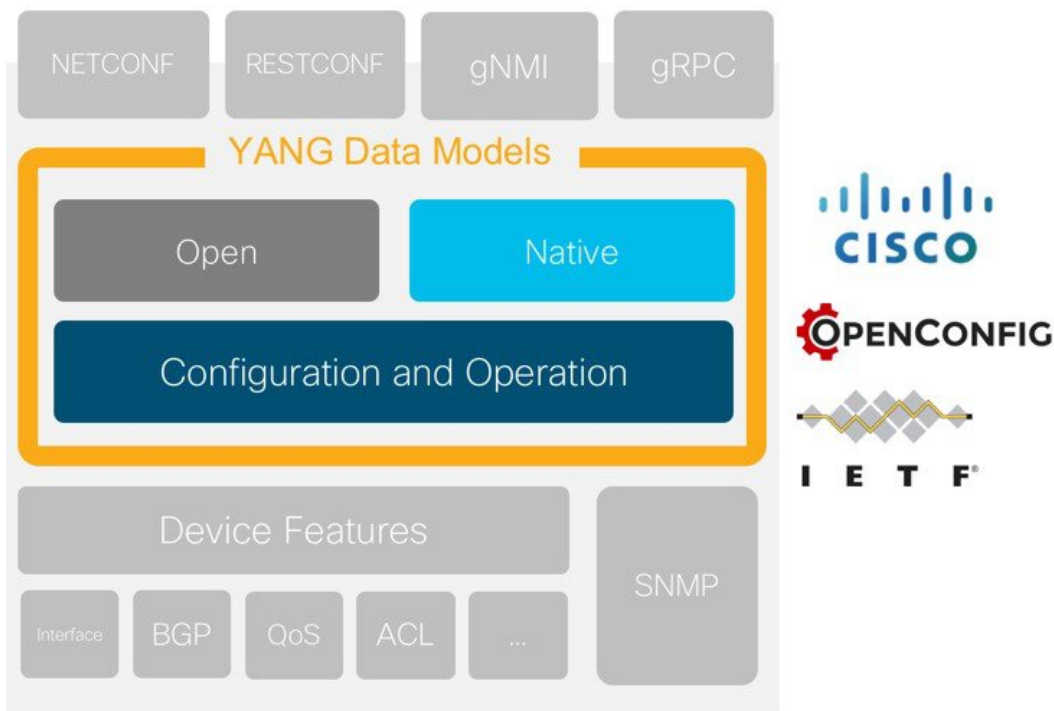
## Verifying gNMI status

To verify the gNMI interface is operational, run the command: `show gnmi-yang state`. Ensure the state is enabled and status is up.

```
WLC#show gnmi-yang state
State          Status
-----
Enabled        Up
```

## YANG Data Models

YANG is a data modelling language for NETCONF. YANG models within the IOS XE device have been defined that describe how to structure the data to send or receive. The YANG standard was defined in RFC6020. There are 2 main types of YANG models in use: Native and Open. The models are further categorized as either Configuration or Operational models. The configuration models can be used for programmatic configuration while the operational models can be used with telemetry.



## Native Models

The Native models for wireless can be grouped into two main categories: config and operational. The config modules contain configuration information for the related features, while the operational models provide run time and operational data about the feature.

## Cisco-IOS-XE-wireless: Modules

**Config:**

ap	mesh
apf	mobility
cts-sxp	mstream
dot11	rf
fabric	rfid
flex	rogue
fqdn	rrm
general	security
location	site
	wlan

**Oper:**

access-point	mobility
client	nmsp
fqdn	rf-profile
lisp-agent	rfid
mcast	rogue
mesh	rrm



## Native Configuration Models

Config Module	Data Model Details
Ap	AP configuration: tags, policy tags, rf tags, site tags
Apf	Global dot11 parameters: country, association request limit, blacklisting
Cts-sxp	CTS SXP configuration: connection information, SXP mode, password, profile name, hold times, retry periods
Dot11	802.11 configuration: TX Power, 802.11n/802.11ac features, admin state, data rates
Fabric	Fabric configuration: VNID name, SDA config, fabric profile
Flex	Flex configuration: RLAN configuration, VLAN ACL's
FQDN	FQDN configurations: url-filter, domain names, filter action
general	Misc WLC configurations: AP join configs, wireless management interfaces, multicast, WSA config
location	Location configurations: NMSP, CMX cloud, server URL, timeouts, measurement intervals
mesh	Wireless Mesh configurations: mesh profile, psk, aaa method, threshold values
mobility	Mobility configurations: mode, config, peer config, group config, guest mode
mstream	Multicast configurations: group, url, group name, stream groups, priority
rf	RF configurations: allowed channels, rf-profile, data rates config, DCA
rfid	RFID configurations: timeouts, RSSI expiry, notify thresholds
rogue	Rogue configurations: rogue global parameters, SSID rules, clients, ignore list
rrm	RRM configuration: intervals, grouping algorithm, static members
security	AP Security configurations: AP LSC config, CA trustpoint, key size
site	Site configurations: site tag, rlan ports, master AP, join profile, BLE, AP PCAP, AP config profiles
wlan	WLAN configuration: aaa, dhcp, umbrella, 802.11v, flex profiles, WLAN SSID, 802.1x security

## Native Operational Models

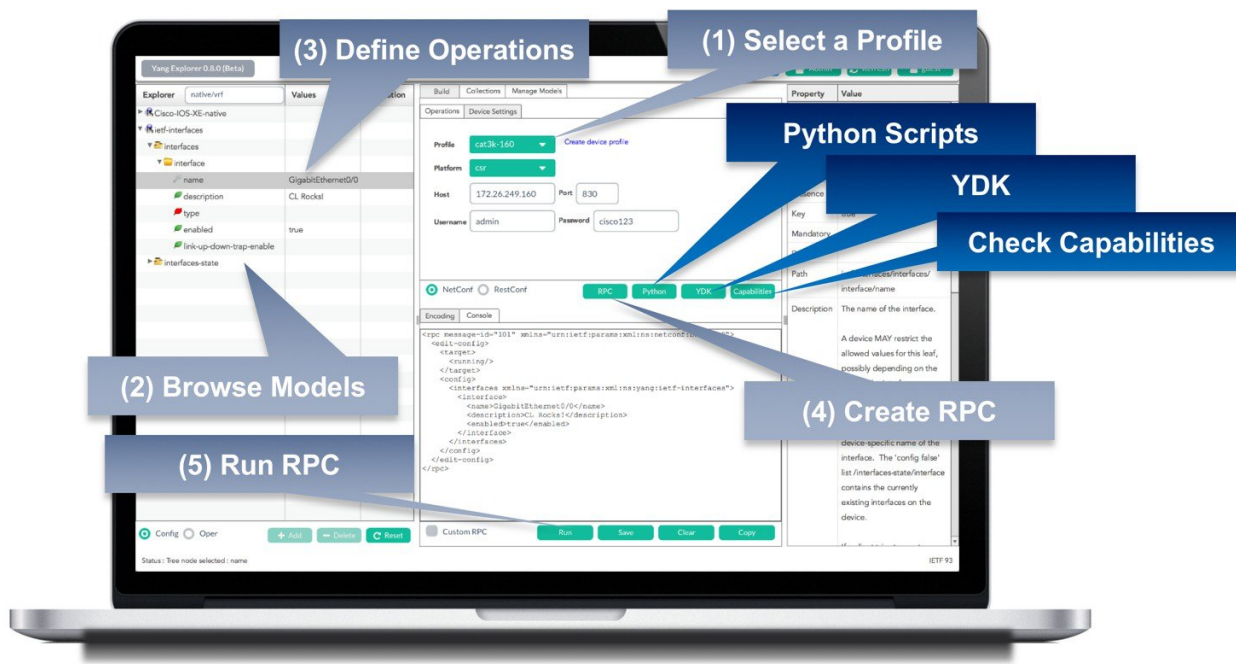
Operational Modules	Data Model Details
access-point	Join failures, discovery, certificates, country, QoS, ATF, radio power
client	Session details, eapol, wlan id, delete reasons, association details
FQDN	DNS snooping, learned IP's, timestamps, actions applied
LISP	Fabric capabilities, lisp agent, tcp and udp failure counts, map registers
mcast	Multicast
mesh	Mesh
mobility	Mobility
nmosp	NMSP Location
rf-profile	RF Profiles
rfid	RFID
rogue	Rogues
rrm	Radio Resource Management

## Open Models

In addition to the Cisco native configuration and operational models, there are several additional YANG models that are supported on the device. The capabilities exchange via NETCONF and RESTCONF's yang-library list all of these models. There are models for SNMP MIB's, IETF, and OpenConfig. These models can be used in the same way as the Native models, however, they offer a limited or subset of the capabilities available on the device. For this reason it is recommended to utilize the Cisco Native models whenever possible.

## YangExplorer Tool

YangExplorer is a tool used for interfacing with the NETCONF and RESTCONF interfaces. It is available on Github and can be used to generate and send XML RPC payloads to the device.



<https://github.com/CiscoDevNet/yang-explorer>

## YangExplorer Installation

Follow the instructions from the Github site at <https://github.com/CiscoDevNet/yang-explorer> to complete the installation. Detailed instructions are available for Mac and Linux.

### ###2. Installation #####2.1 First time installation #####Prerequisite:

- MAC, Linux (not supported on Windows)
- python 2.7
- pip package manager (<https://pip.pypa.io/en/stable/installing/>)

If already installed, make sure that pip / setuptools are upto date (commands may vary)

```
pip install --upgrade pip
```

```
Ubuntu: sudo pip install --upgrade setuptools
```

- virtualenv (recommended)

```
Ubuntu: sudo apt-get install python-virtualenv
```

```
Fedora: sudo dnf install python-virtualenv
```

```
MAC: sudo pip install virtualenv
```

- graphviz (<http://www.graphviz.org/Download.php>)

```
Ubuntu: sudo apt-get install graphviz
```

```
Fedora: sudo dnf install graphviz
```

```
MAC: brew install graphviz
```

- Browser with latest flash plugin (tested with google chrome)

### #####Download and install:

```
git clone https://github.com/CiscoDevNet/yang-explorer.git
cd yang-explorer
bash setup.sh
```

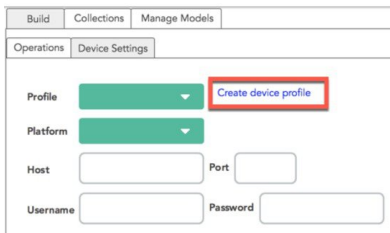
Note: sudo may be required if you do not use virtualenv.

## YangExplorer Day 0 configuration

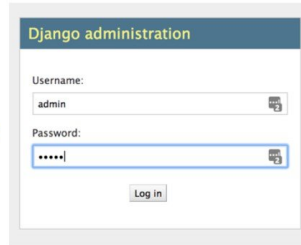
Once YangExplorer is installed and running, a few tasks must be completed before interacting with the Catalyst 9800. Select “Create device profile” and enter the Django administration page with the username and password of ‘admin’. The “Add Device Profile” page loads, where information about the device must be added. Enter the profile name, select the device type, and enter the NETCONF IP, RESTCONF IP, username, and password.

# YangExplorer – Initial Configuration

- Select “Create Device Profile” from main Yang Explorer page
- Use “admin” for both username and password for the Django administration

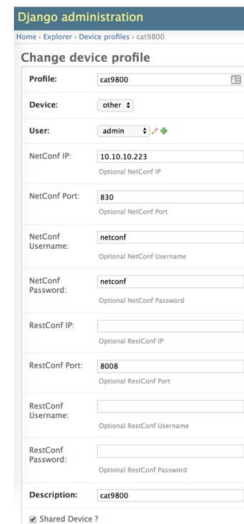


The image shows the main Yang Explorer interface with tabs for 'Build', 'Collections', and 'Manage Models'. Under the 'Operations' tab, there is a 'Device Settings' sub-tab. The 'Profile' dropdown is set to 'cat9800', and the 'Create device profile' button is highlighted with a red rectangle. Other fields like 'Platform', 'Host', 'Port', 'Username', and 'Password' are also visible.



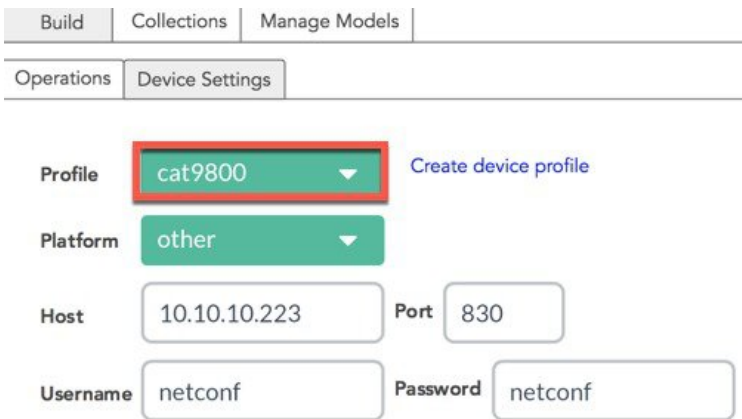
The image shows the Django administration login page. It has a title 'Django administration' and fields for 'Username' (with 'admin' entered) and 'Password' (with masked characters). A 'Log in' button is at the bottom.

Enter details for the cat9800 device including:  
Profile name, device type, user type, NETCONF IP,  
username, password, and description



The image shows the 'Change device profile' page in Django administration for the 'cat9800' device. It includes fields for 'Profile' (cat9800), 'Device' (other), 'User' (admin), 'NetConf IP' (10.10.10.223), 'NetConf Port' (830), 'NetConf Username' (netconf), 'NetConf Password' (netconf), 'RestConf IP', 'RestConf Port' (8008), 'RestConf Username', 'RestConf Password', and 'Description' (cat9800). There is also a 'Shared Device?' checkbox.

Once the device is added, refreshing the main YangExplorer window will show the available device when selecting the “Profile” drop-down.



The image shows the main Yang Explorer interface after the device has been added. The 'Profile' dropdown is now set to 'cat9800' and is highlighted with a red rectangle. The 'Create device profile' button is still visible. Other fields like 'Platform' (other), 'Host' (10.10.10.223), 'Port' (830), 'Username' (netconf), and 'Password' (netconf) are also visible.

To ensure the setup is successful select the “Capabilities” button and YangExplorer will connect to the selected device and return the list of YANG modules that are supported on the device.

Profile **cat9800** [Create device profile](#)

Platform **other**

Host  Port

Username  Password

---

☒ NetConf
 ☐ RestConf
 RPC
Python
YDK
Capabilities

Encoding

```

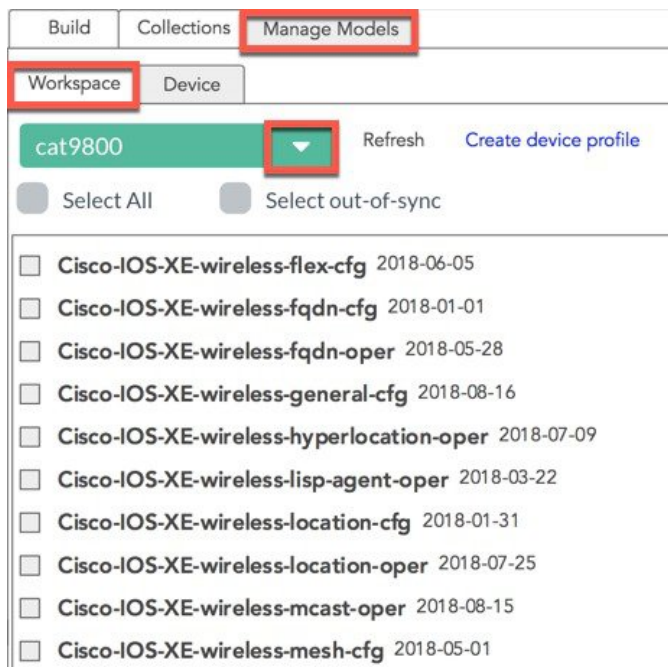
http://cisco.com/ns/yang/Cisco-IOS-XE-wccp?module=Cisco-IOS-XE-wccp&revision=2017-11-27
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-access-point-oper?module=Cisco-IOS-XE-wireless-access-point-oper&revision=2018-06-28
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-ap-cfg?module=Cisco-IOS-XE-wireless-ap-cfg&revision=2018-05-10
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-ap-types?module=Cisco-IOS-XE-wireless-ap-types&revision=2018-06-15
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-apf-cfg?module=Cisco-IOS-XE-wireless-apf-cfg&revision=2018-08-13
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-client-oper?module=Cisco-IOS-XE-wireless-client-oper&revision=2018-09-06
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-client-types?module=Cisco-IOS-XE-wireless-client-types&revision=2018-08-08
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-cts-sxp-cfg?module=Cisco-IOS-XE-wireless-cts-sxp-cfg&revision=2018-04-09
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-cts-sxp-oper?module=Cisco-IOS-XE-wireless-cts-sxp-oper&revision=2018-08-07
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-dot11-cfg?module=Cisco-IOS-XE-wireless-dot11-cfg&revision=2018-06-13
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-enum-types?module=Cisco-IOS-XE-wireless-enum-types&revision=2018-05-18
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-events-oper?module=Cisco-IOS-XE-wireless-events-oper&revision=2018-06-11
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-fabric-cfg?module=Cisco-IOS-XE-wireless-fabric-cfg&revision=2018-03-08
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-flex-cfg?module=Cisco-IOS-XE-wireless-flex-cfg&revision=2018-06-05
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-fqdn-cfg?module=Cisco-IOS-XE-wireless-fqdn-cfg&revision=2018-01-01
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-fqdn-oper?module=Cisco-IOS-XE-wireless-fqdn-oper&revision=2018-05-28
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-general-cfg?module=Cisco-IOS-XE-wireless-general-cfg&revision=2018-08-16
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-hyperlocation-oper?module=Cisco-IOS-XE-wireless-hyperlocation-oper&revision=2018-07-09
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-lisp-agent-oper?module=Cisco-IOS-XE-wireless-lisp-agent-oper&revision=2018-03-22
http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-location-cfg?module=Cisco-IOS-XE-wireless-location-cfg&revision=2018-01-31
  
```

## Downloading the YANG Models from YangExplorer

From within YangExplorer the YANG modules can be downloaded from the device. Follow the steps to sync and subscribe to the models:

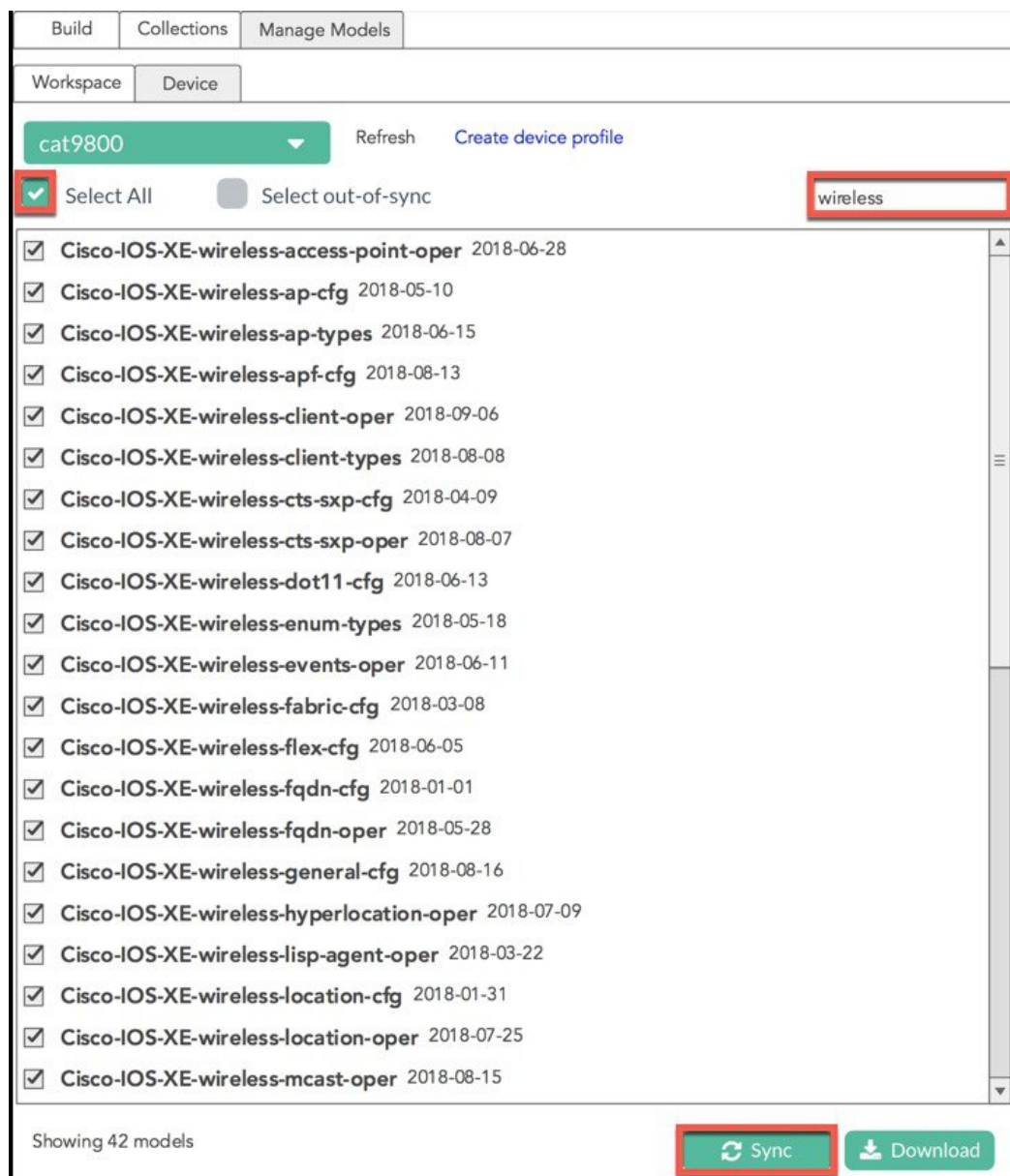
1. Select “Manage Models” tab.
2. Select “Device” tab.
3. Select the device profile by selecting the down arrow.





4. To see only the wireless modules, enter “wireless” into the search box.
5. Enable the “Select All” tick-box to select all wireless modules
6. Select “Sync” to download the modules from the device





- Once the download is completed, select “Subscribe” and the models will now be available in the Explorer tab of YangExplorer.



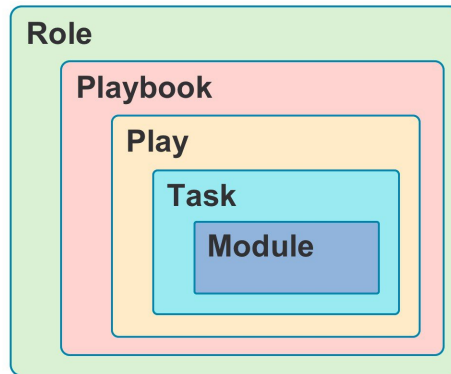
## Ansible Automation

Ansible is a popular and easy to use open-source software suite that automates software provisioning, configuration, and management. It connects to and controls devices via SSH, NETCONF, and a variety of other protocols as well. Ansible is agentless, meaning there is no installation or requirements on the target device, other than having an accessible API or interface. It is minimal in nature and provides a secure and reliable way to interact with remote devices. It is commonly used with other automation tools to accomplish complex workflows as it is highly adaptable. Below are some examples of using Ansible to complete basic day0 configuration tasks.

Ansible has several components that work together to provide a holistic solution. There are modules that are reusable and standalone script that can be called. Tasks call upon modules to perform an action. When there are multiple tasks, a play can be used to call the tasks in order. A playbook is then used when there are multiple plays. Finally, a role is a set of playbooks.

## Ansible Taxonomy

- **Role:** a set of Playbooks ( )
- **Playbook:** repeatable standard config
- **Play:** a set of tasks
- **Task:** single action that references a module
- **Module:** reusable, standalone scripts



## Ansible Example: Enable NETCONF-YANG and RESTCONF

The hosts file has connection details and device specific information including credentials. In this example it is assumed that the Catalyst 9800 has already been configured to allow SSH logins, and that the enable password has also been set. The hosts files and YAML file are used together to accomplish a task, in this case, to connect to the CLI over SSH, enter enable mode, and execute the requires IOS commands to enable netconf-yang and setup the AAA requirements.

## Ansible hosts file for Day 0 configuration

This example hosts.txt file contains the variables needed to successfully establish a connection to the device.

```
[all:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_user=admin
ansible_password=Cisco123
ansible_become=yes
ansible_become_method=enable
ansible_become_pass=Cisco123
[cat9800]
cat9800-1 ansible_host=10.10.10.224
```

```
cat hosts.txt
[all:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_user=admin
ansible_password=Cisco123
ansible_become=yes
ansible_become_method=enable
ansible_become_pass=Cisco123

[cat9800]
cat9800-1 ansible_host=10.10.10.223
```

## Ansible YAML configuration file to enable NETCONF and RESTCONF

This example enable\_netconf\_yang.yaml YAML file that can be used to enable the NETCONF interface on the device and configure the authentication prerequisites including adding a user.

```
---
- hosts: cat9800
  gather_facts: no

  tasks:
    - ios_config:
        commands:
          - aaa new-model
          - aaa authorization exec default local
          - aaa authentication login default local
          - aaa session-id common
          - username netconf privilege 15 password 0 netconf
          - netconf-yang
          - ip http secure-server
          - restconf
        save_when: modified
```

```

cat enable_netconf_yang.yaml
---
- hosts: cat9800
  gather_facts: no

  tasks:
    - ios_config:
      commands:
        - aaa new-model
        - aaa authorization exec default local
        - aaa authentication login default local
        - aaa session-id common
        - username netconf privilege 15 password 0 netconf
        - netconf-yang
        - ip http secure-server
        - restconf
      save_when: modified

```

## Ansible YAML show file to verify NETCONF and RESTCONF configuration

This example cat9800\_verify.yaml YAML file will run 2 ios SHOW commands to verify that netconf-yang and restconf is enabled, and the output will be registered and displayed on the screen once executed:

```

- hosts: cat9800
  gather_facts: no

  tasks:
    - ios_command:
      commands:
        - show run | i netconf-yang
        - show run | i restconf
      register: show
    - debug: var=show.stdout_lines

```

```

cat cat9800_verify.yaml
---
- hosts: cat9800
  gather_facts: no

  tasks:
    - ios_command:
      commands:
        - show run | i netconf-yang
        - show run | i restconf
      register: show
    - debug: var=show.stdout_lines

```

## Executing the task to enable and verify NETCONF and RESTCONF

The ‘ansible-playbook’ command can be used to execute the task that we define above to enable the NETCONF-YANG interface on the device. In this example we will define a variable to set the Host Key Checking to false, so that the SSH host key is not validated. In production environments it is important to verify the authenticity of the device being accessed, however in this lab example the check is set to false for ease of use.

From the command line, execute the following command to run the enable\_netconf\_yang.yaml configuration:

```
$ ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -i ./hosts.txt ./enable_netconf_yang.yaml
```

```
ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -i ./hosts.txt ./enable_netconf_yang.yaml

PLAY [cat9800] *****

TASK [ios_config] *****
changed: [cat9800-1]

PLAY RECAP *****
cat9800-1                : ok=1    changed=1    unreachable=0    failed=0
```

From the command line, execute the following command to run the cat9800\_verify.yaml configuration

```
$ ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -i ./hosts.txt ./cat9800_verify.yaml
```

```
ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -i ./hosts.txt ./cat9800_verify.yaml

PLAY [cat9800] *****

TASK [ios_command] *****
ok: [cat9800-1]

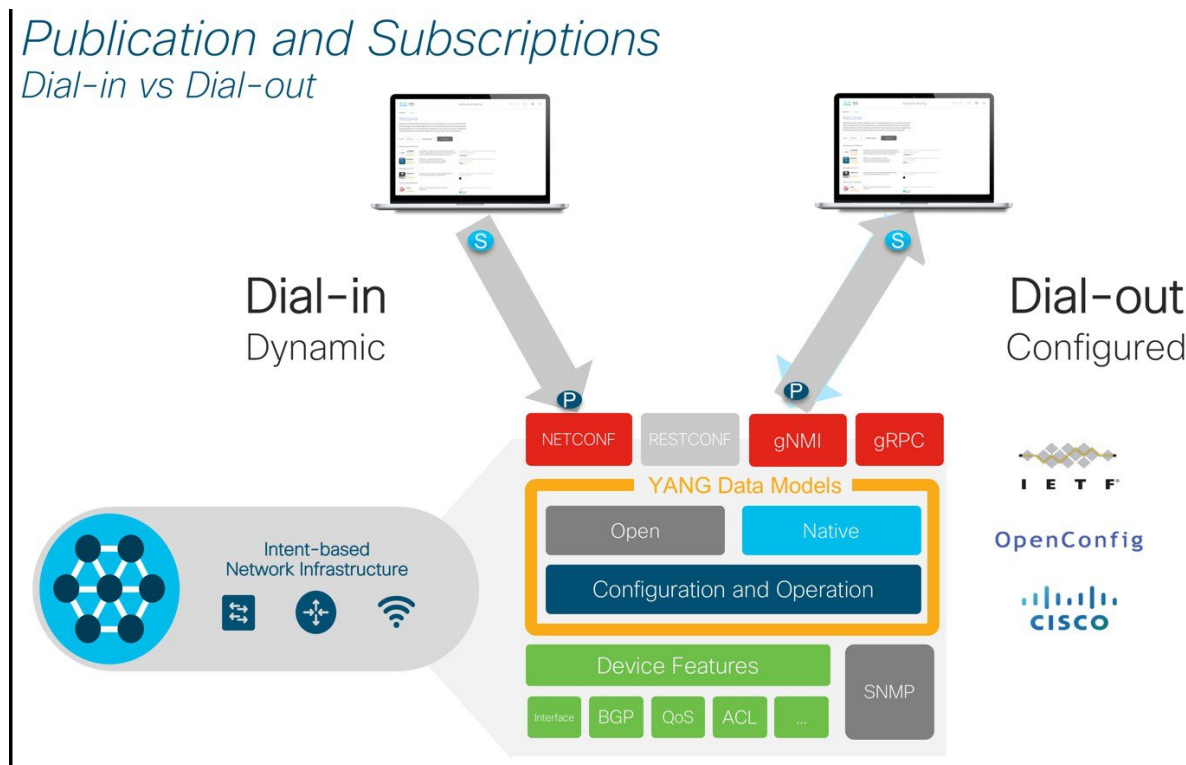
TASK [debug] *****
ok: [cat9800-1] => {
  "show.stdout_lines": [
    [
      "netconf-yang"
    ],
    [
      "restconf"
    ]
  ]
}

PLAY RECAP *****
cat9800-1                : ok=2    changed=0    unreachable=0    failed=0
```

# Telemetry

## Dial-in dynamic vs Dial-out configured subscriptions

Both the NETCONF and gNMI interfaces support telemetry subscriptions in either a dial-in or dial-out configuration. IOS XE supports NETCONF Dial-in and gRPC supports dial-out style configuration. With a dial-in or “dynamic” subscription, the subscriber must first establish session a connection to the device and then subscribe to the data models. The NETCONF session must remain established in order for telemetry data to remain streaming. If the session is disconnected then the telemetry subscription must be manually re-established. With dial-out or “configured” subscriptions, once the configuration is setup by the user, the device will maintain the subscription configuration and sends telemetry to the subscriber without needing an active session to the collector.



## Dial-in NETCONF dynamic telemetry subscription with ncc

Dial-in telemetry subscriptions can be easily created by using the open source NCC tools available from <https://github.com/CiscoDevNet/ncc>. The repository can be simply cloned from Github with the git clone command. An additional requirement is to use a patched ncclient tool that works with the ncc-establish-subscription.py tool. This can be installed by following the directions on the ncc Github page or by executing the two commands below:

```
$ git clone https://github.com/CiscoDevNet/ncc
$ sudo pip install --upgrade git+https://github.com/CiscoDevNet/ncclient.git
```

Once downloaded a subscription can be established by running the following command:

```
$ python ncc-establish-subscription.py --host 10.10.10.223 --port 830 -x
"/wireless-access-point-oper:access-point-oper-data/wireless-interface-data" --period 5000 -u netconf -p netconf
```



```
python ncc-establish-subscription.py --host 10.10.10.223 --port 830 -x "/wireless-access-point-oper:access-point-oper-data/wireless-interface-data" --period 1000 -u netconf -p netconf
No handlers could be found for logger "ncclient.operations.subscribe"
Subscription Result : notif-bis:ok
Subscription Id : 2147483655
-->>
(Default Callback)
Event time : 2018-10-25 21:52:55.450000+00:00
Subscription Id : 2147483655
Type : 1
Data :
<datastore-contents-xml xmlns="urn:iETF:params:xml:ns:yang:ietf-yang-push">
  <access-point-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-access-point-oper">
    <wireless-interface-data>
      <intf-name>GigabitEthernet1</intf-name>
      <intf-type>Management</intf-type>
      <intf-id>0</intf-id>
      <mgmt-ip>10.10.10.223</mgmt-ip>
      <net-mask>255.255.255.0</net-mask>
      <mgmt-mac>52:54:00:02:35:FA</mgmt-mac>
    </wireless-interface-data>
  </access-point-oper-data>
</datastore-contents-xml>
```

In the example above a telemetry subscription has been created using the Cisco-IOS-XE-wireless-access-point-oper YANG model, which has a xpath of “/wireless-access-point-oper:access-point-oper-data/wireless-interface-data”. YangExplorer is used to determine the correct XPath Filter for this YANG model as seen in the screenshot below:

Property	Value
Name	wireless-interface-data
Node Type	container
Data Type	
Access	read-only
Presence	true
Key	
Mandatory	
Default	
Path	Cisco-IOS-XE-wireless-access-point-oper/access-point-oper-data/wireless-interface-data
Description	Contains wireless interface data
XPath Filter	/wireless-access-point-oper:access-point-oper-data/wireless-interface-data

## Dial-out gRPC configured subscription

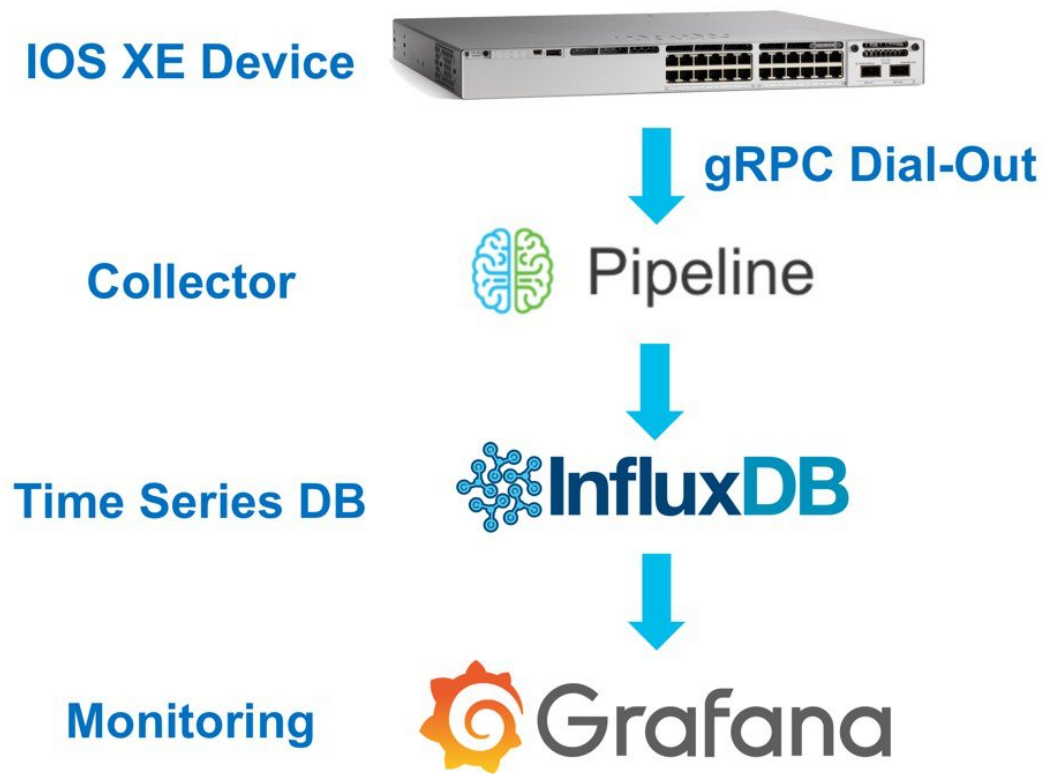
Instead of having the subscriber or collector establish the telemetry subscription to the device, the device can be configured with a dial-out subscription. In this mode the collector’s role is only to collect and process the data and it is not responsible for establishing a session into the device in order to receive the telemetry data. Dial out subscriptions only support encoding with key-value google protocol buffers (kv-gpb) over a gRPC-TCP connection.

The following configuration can be used to establish a dial-out gRPC telemetry subscription. This configuration creates a new subscription with an ID of 101. The encoding is set to kv-gpb, and the xpath filter defines the KPI to subscribe to, in this case wireless-interface-data. The xpath filter is defined within the YANG model and YangExplorer is used to determine the exact xpath for this model. The source address and VRF to use from the device is set, as well as the receiver IP, port, and protocol. The yang-push stream is set to 1000 centiseconds, so data will be published every 10 seconds.

```
telemetry ietf subscription 101
encoding encode-kvgpb
filter xpath /wireless-access-point-oper:access-point-oper-data/wireless-interface-data
source-address 10.10.10.223
source-vrf Mgmt-vrf
stream yang-push
update-policy periodic 1000
receiver ip address 10.10.10.49 57500 protocol grpc-tcp
```

## Receiving events with Pipeline

The key-value encoded google protocol buffer (kv-gpb) telemetry data that is sent over the gRPC interface can be received with many tools and in many different configurations, depending on the business needs and use-cases. Pipeline is an open source set of tools that can be used to receive the data and is available on Github at <https://github.com/cisco/bigmuddy-network-telemetry-pipeline>. Pipeline works by acting as the gRPC receiver, where it processes the Google Protocol Buffers and sends the data into the time series database InfluxDB. From there Grafana can be used to visualize the data. Pipeline is highly configurable and can receive a variety of telemetry sources as well as output data to a variety of data source, including Kafka, InfluxDB, and Prometheus. Follow the documentation on the Github page to setup Pipeline to receive telemetry as required. Additional documentation is available from <https://xrdocs.io/telemetry/tutorials/2018-06-04-ios-xr-telemetry-collection-stack-intro/>



## Verifying telemetry subscriptions

There are several show commands available to verify the status of the telemetry subscription configurations:

```
show telemetry ietf subscription all
show telemetry ietf subscription <ID> receiver
show telemetry ietf subscription all <ID> detail
```

Examples of each are below:

```
WLC#show telemetry ietf subscription all
Telemetry subscription brief
```

ID	Type	State	Filter type
2147483653	Dynamic	Valid	xpath

```
WLC#show telemetry ietf subscription 2147483654 detail
Telemetry subscription detail:

Subscription ID: 2147483654
Type: Dynamic
State: Valid
Stream: yang-push
Filter:
  Filter type: xpath
  XPath: /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper-data
Update policy:
  Update Trigger: periodic
  Period: 1000
Encoding: encode-xml
Source VRF:
Source Address:
Notes:

Receivers:
  Address      Port      Protocol      Protocol Prof
-----
10.10.10.49    43824     netconf
```

## Conclusion

Using the details within the guide the programmatic interfaces can be enabled and configured for use. Configured and Dynamic telemetry subscriptions can be established using open source tools. Example XML payloads can be used to create, verify, and remove a WLAN programmatically using the YangExplorer tools. An example Ansible configuration can be used to enable the programmatic interfaces on the Catalyst 9800 device.

## Additional Resources

Cisco IOS-XE Programmability Book <https://www.cisco.com/c/dam/en/us/products/collateral/enterprise-networks/nb-06-ios-xe-prog-ebook-cte-en.pdf>

# Cisco IOS XE Programmability Automating Device Lifecycle Management



## Programmability Configuration Guide

[https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/169/b\\_169\\_programmability\\_cg.html](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/169/b_169_programmability_cg.html)

## Reference

- YangExplorer: <https://github.com/CiscoDevNet/yang-explorer>
- pYang: : <https://github.com/mbj4668/pyang>
- Pipeline on DevNet: <https://developer.cisco.com/codeexchange/github/repo/cisco/bigmuddy-network-telemetry-pipeline/>

- Pipeline on Github: <https://github.com/cisco/bigmuddy-network-telemetry-pipeline>
- Pipeline on XR Docs: <https://xrdocs.io/telemetry/tutorials/2018-06-04-ios-xr-telemetry-collection-stack-intro/>
- NCC: <https://github.com/CiscoDevNet/ncc>

## Questions?

- Cisco DevNet: <https://developer.cisco.com/>
- Cisco Communities: <https://community.cisco.com/>



**Americas Headquarters**  
Cisco Systems, Inc.  
San Jose, CA 95134-1706  
USA

**Asia Pacific Headquarters**  
CiscoSystems(USA)Pte.Ltd.  
Singapore

**Europe Headquarters**  
CiscoSystemsInternationalBV  
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).