



DevNet Coffee Break IOS XE RESTCONF

Cisco Connect Croatia 2017

Djordje Vulovic

Consulting Systems Engineer

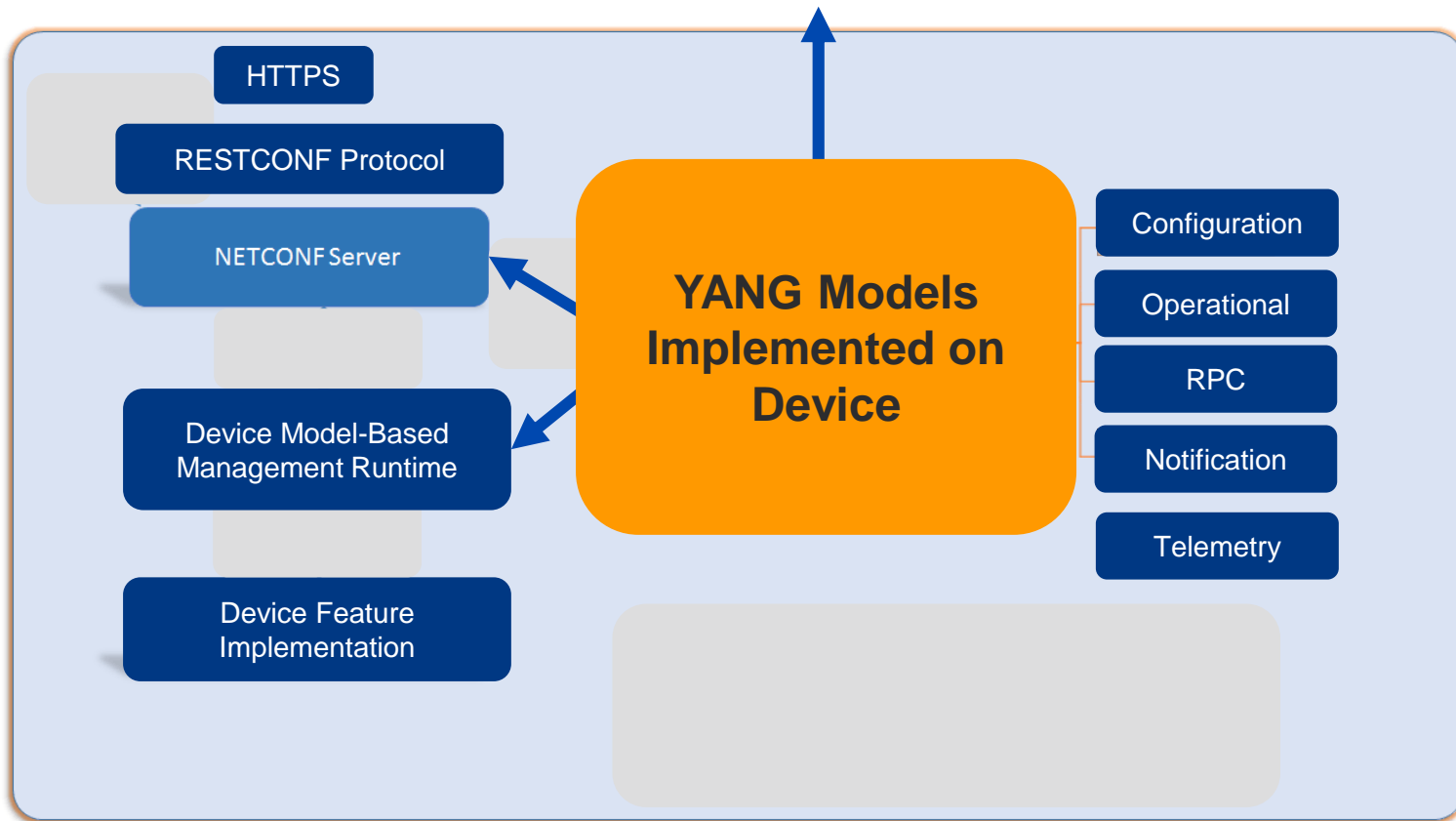
dvulovic@cisco.com

https://github.com/djordjevulovic/Cisco_Connect_HR_2017

What is RESTCONF ?

- A REST-like protocol running over HTTP for accessing data:
 - Not intended to replace NETCONF, but rather provide an additional simplified interface that follows REST-like principles
- The RESTCONF protocol operates on a conceptual datastore defined with the YANG:
 - RESTCONF standard describes how to map a YANG specification to a RESTful interface
 - On Cisco devices, RESTCONF operates on the same datastores operated on by NETCONF
- Request and response data can be in XML or JSON format.
- IETF standard (RFC 8040, January 2017)

RESTCONF and YANG



DevNet Sandbox Labs

The screenshot displays the DevNet Sandbox Labs interface in a web browser. The URL bar shows <https://devnetsandbox.cisco.com/RM/Topology>. The page features a navigation bar with the DevNet logo and links for LAB MANAGEMENT, DVULOVIC, DEVNET, HELP, and TUTORIALS. Below the navigation bar, there are several lab cards arranged in a grid. Each card includes a title, a brief description, and a button to either 'RESERVE' or 'BUILD' the lab. A red rectangle highlights the 'RESTCONF' lab card, which is labeled 'Version 16.2' and includes a 'More Info' link. Other visible lab cards include 'NX-API', 'Open NX-OS', 'OpenPlatformNFV-Ce...', 'OpenPlatformNFV-Ub...', 'OpenPnP', 'Packaged Contact Center Enterprise', 'pxGrid', 'DevNet Events Claim an Event Lab', 'Remote Expert Mobile', 'Packaged Contact Ce...', 'Registration Utility', 'Remote Expert Mobile', 'RESTCONF', 'Tropo', 'Contact Center', 'UCS', 'UCS IMC', and 'WebEx Web Conferencing'.

Lab Name	Description	Action
NX-API	Simplified switch interaction using virtualized NX-OS (NX-OS Lab with virtual switch)	ALWAYS-ON
Open NX-OS	New Standalone Only Open NX-OS Lab with virtual switch	RESERVE
OpenPlatformNFV-Ce...	CentOS 7 OPNFV Brahmputra Release	RESERVE
OpenPlatformNFV-Ub...	Ubuntu 14.04 OPNFV Brahmputra Release	RESERVE
OpenPnP	Open Plug N Play Lab	RESERVE
Packaged Contact Center Enterprise	Version 11.5	RESERVE
pxGrid	Execute pxGrid scripts on Cisco Identity Services Engine	ALWAYS-ON
DevNet Events Claim an Event Lab	Requires Event Passcode	BUILD
Remote Expert Mobile	WebRTC Collaboration	RESERVE
Packaged Contact Ce...	The Modern PCE lab for Devs!	RESERVE
Registration Utility	Acknowledgement will be sent via email in a few minutes	BUILD
Remote Expert Mobile	Be an expert from anywhere in the world!	RESERVE
RESTCONF	Version 16.2, The RESTCONF Always On Sandbox based on IOS-XE	ALWAYS-ON
Tropo	Version 10.6	RESERVE
Contact Center	Version 10.6	RESERVE
UCS	Version 3.1.2b, UCS Emulator and UCS Director	RESERVE
UCS IMC	Version 3.0	RESERVE
WebEx Web Conferencing	Version 3.0	RESERVE



<https://devnetsandbox.cisco.com>

DevNet RESTCONF SandBox

The screenshot shows the DevNet RESTCONF SandBox interface. The left sidebar contains instructions, and the main area shows a 'Sandbox Lab (Reserve to Activate)' button and a 'Try It Now' button.

INSTRUCTIONS

See the [Lab Architecture Diagram](#) for more information about the RESTCONF Always On Sandbox topology.

Type of Access:
No VPN connection is required.
Access to the RESTCONF API is provided via Telnet/SSH access to the RESTCONF Always On Sandbox using these credentials:
Username: root
Password: D_Vayl_10&

Access Details:
Developers and network engineers access the RESTCONF Always On Sandbox directly using the following information:

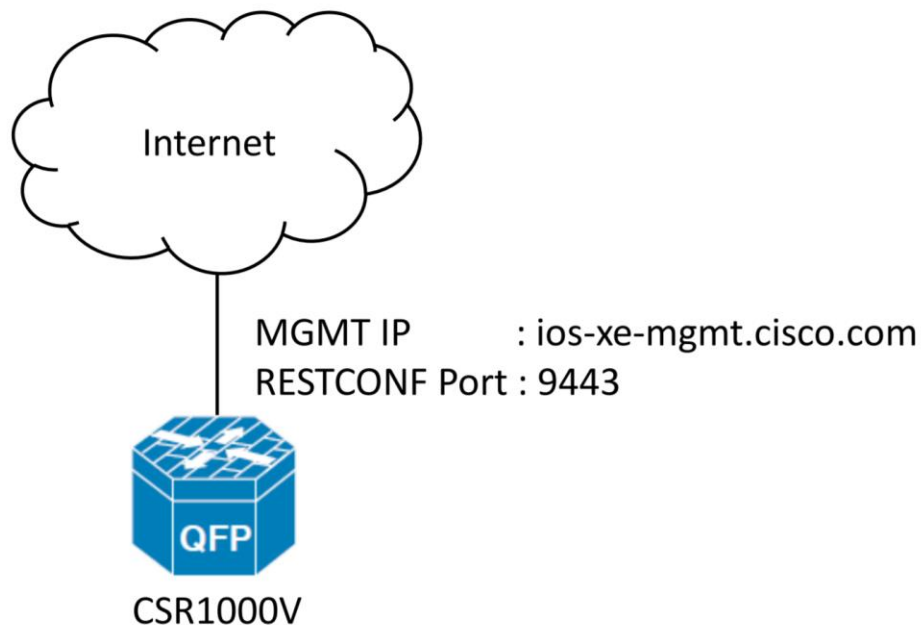
CSR1000V Host : ios-xe-mgmt.cisco.com
RESTCONF Port : 9443
Credentials:
Username: root
Password: D_Vayl_10&

Hello World:
There are various examples on the DevNet GitHub page to get up and running with RESTCONF quickly.

Sandbox Lab (Reserve to Activate)

Try It Now
No Reservation Required
← See Instructions

RESTCONF DevNet Sandbox Diagram



“Get Interface List” Request (Native Model)

http://ios-xe-mgmt.cisco.com:9443/api/config/native/interface/

```
GET /api/config/native/interface/ HTTP/1.1
Host: ios-xe-mgmt.cisco.com:9443
Authorization: Basic cm9vdDpEXlZheSFfMTAm
Accept: application/vnd.yang.data+json
```

Username: root
Password: D_Vay!_10&

Build Request in Postman Tool

The screenshot displays the Postman interface for building a request. At the top, there's a tab labeled "Get List of Interfaces (Native)" and a dropdown menu set to "No Environment". The main area shows a "GET" request to the URL "http://ios-xe-mgmt.cisco.com:9443/api/config/native/interface/". Below the URL bar, there are tabs for "Authorization", "Headers (2)", "Body", "Pre-request Script", and "Tests". The "Authorization" tab is selected, showing a "Basic Auth" dropdown. The "Username" field contains "root" and the "Password" field contains "D_Vay!_10&". A checkbox labeled "Show Password" is checked. To the right of the password field, there's a note: "The authorization header will be generated and added as a custom header" and a checkbox labeled "Save helper data to request". At the bottom right, there are "Clear" and "Update Request" buttons. The "Update Request" button is highlighted with a blue border.

Get List of Interfaces (Native) | No Environment

GET `http://ios-xe-mgmt.cisco.com:9443/api/config/native/interface/` Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

Type Basic Auth Clear Update Request

Username root

Password D_Vay!_10& ☒ Show Password

The authorization header will be generated and added as a custom header

☐ Save helper data to request

Execute Request in Postman Tool

Get List of Interfaces (Native)

GET `http://ios-xe-mgmt.cisco.com:9443/api/config/native/interface/` Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

Key	Value	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	Basic cm9vdDpEX1ZheSFfMTAm		
<input checked="" type="checkbox"/> Accept	application/vnd.yang.data+json		
New key	value		

Body Cookies Headers (8) Tests Status: 200 OK Time: 602 ms

Pretty Raw Preview JSON Save Response

```
1 {
2   "interface": {
3     "GigabitEthernet": [
4       {
5         "name": "1"
6       },
7       {
8         "name": "2"
9       },
10      {
11        "name": "3"
12      }
13    ],
14    "VirtualPortGroup": [
15      {
16        "name": 0
17      }
18    ]
19  }
20 }
```

“Get Interface List” Response (Native Model)

```
{
  "interface": {
    "GigabitEthernet": [
      {
        "name": "1"
      },
      {
        "name": "2"
      },
      {
        "name": "3"
      }
    ],
    "VirtualPortGroup": [
      {
        "name": 0
      }
    ]
  }
}
```

“Get Interface List” Request (IETF Model)

```
GET /api/config/interfaces HTTP/1.1  
Host: ios-xe-mgmt.cisco.com:9443  
Authorization: Basic cm9vdDpEXlZheSFfMTAm  
Accept: application/vnd.yang.data+json
```

“Get Interface List” Response (IETF Model)

```
{
  "interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet1"
      },
      {
        "name": "GigabitEthernet2"
      },
      {
        "name": "GigabitEthernet3"
      }
    ]
  }
}
```

“Create EFP” Request (Native Model)

```
PUT /api/running/native/interface/GigabitEthernet/2/service/instance/205/ HTTP/1.1
Host: ios-xe-mgmt.cisco.com:9443
Authorization: Basic cm9vdDpDIXNjMDEyMw==
Content-Type: application/vnd.yang.data+json

{
    "instance" : {
        "id": "205",
        "ethernet": {},
        "description": "EFP_205"
    }
}
```

“Get List of EFPs on Interface” Request

```
GET /api/config/native/interface/GigabitEthernet/2/service/ HTTP/1.1
Host: ios-xe-mgmt.cisco.com:9443
Authorization: Basic cm9vdDpDIXNjMDEyMw==
Accept: application/vnd.yang.data+json
```

“Get List of EFPs on Interface” Response

```
{
  "service": {
    "instance": [
      {
        "id": 205
      },
      {
        "id": 206
      }
    ]
  }
}
```

“Get EFP Details” Request

```
GET /api/config/native/interface/GigabitEthernet/2/service/instance/205 HTTP/1.1  
Host: ios-xe-mgmt.cisco.com:9443  
Authorization: Basic cm9vdDpDIXNjMDEyMw==  
Accept: application/vnd.yang.data+json
```


“Get EFP Details” Response

```
{
  "instance": {
    "id": 205,
    "ethernet": [
      null
    ],
    "description": "EFP_205",
    "encapsulation": {
      "dot1ad": {},
      "dot1q": {},
      "priority-tagged": {}
    },
    "rewrite": {
      "ingress": {
        "tag": {
          "translate": {}
        }
      }
    },
  },
  ...
}
```

Python: Execute Generic IOS XE RESTCONF Request

```
#####
restconf_url_prefix = "http://ios-xe-mgmt.cisco.com:9443/api/"
restconf_username = 'root'
restconf_password = 'D Vay!_10&'
#####
class HTTP_Accept_Types(Enum):
    json_data = 1
    json_collection = 2

def RESTCONF_GET(url_suffix, accept_type = HTTP_Accept_Types.json_data):

    url = restconf_url_prefix + url_suffix

    if accept_type == HTTP_Accept_Types.json_data:
        accept_string = 'application/vnd.yang.data+json'
    elif accept_type == HTTP_Accept_Types.json_collection:
        accept_string = 'application/vnd.yang.collection+json'

    header = {"Accept": accept_string}

    jsonObject = {}

    try:
        response = requests.get(url, headers=header, auth=HTTPBasicAuth(restconf_username, restconf_password), verify=False)

        if (response.status_code==200):
            jsonObject = response.json()

    except requests.exceptions.RequestException as e:
        print ("ERROR:" , e)

    return jsonObject
```

Python: Specific IOS XE RESTCONF Requests

```
def RESTONF_GET_Interface_List():
    return RESTCONF_GET('config/native/interface/', HTTP_Accept_Types.json_data)

def RESTONF_GET_Interface_EFP_List(intf_type, intf_number):
    return RESTCONF_GET("config/native/interface/"+intf_type+"/"+intf_number+"/service",
HTTP_Accept_Types.json_data)

def RESTONF_GET_EFP_Details(intf_type, intf_number, efp_number):
    return
RESTCONF_GET("config/native/interface/"+intf_type+"/"+intf_number+"/service/instance/"+efp_number,
HTTP_Accept_Types.json_data)

def RESTONF_GET_Interface_Config():
    return RESTCONF_GET("config/native/interface?deep", HTTP_Accept_Types.json_data)
```

Real-world App: Show_interface_EFP

- Current Problem: no simple way to view EVCs and their descriptions on IOS XE (e.g. ASR 903)
 - Simple operational question – on which port/EVC is specific user connected?
- It would be good to have a simple table stating:
 - Port
 - EVC ID
 - Description
 - + other config parameters (e.g. QoS policy-map)
 - + operational parameters (e.g. PW status)

Python: show_interface_EFP function

```
def show_evc():

    json1 = RESTONF_GET_Interface_List()

    intf_type = "GigabitEthernet";

    print('{:10s} {:20s}  {:30s}\n'.format("EFP ID", "Interface", "Description"))

    for intf_num in json1["interface"]["GigabitEthernet"]:

        json2 = RESTONF_GET_Interface_EFP_List(intf_type, intf_num["name"])

        if (json2):
            for efp in json2["service"]["instance"]:
                json3 = RESTONF_GET_EFP_Details(intf_type, intf_num["name"], str(efp["id"]))

                if ("description" not in json3["instance"]):
                    desc = ""
                else:
                    desc = json3["instance"]["description"]

                print('{:10s} {:20s}  {:30s}'.format(str(efp["id"]), str(intf_type +
intf_num["name"]), str(desc)))
```

Python: show_interface_EFP output

```
C:\Users\dvulovic\PycharmProjects\IOS_XE_XR_NETCONF_RESTCONF>python IOS_XE_show_interface_efp.py
```

EFP ID	Interface	Description
--------	-----------	-------------

205	GigabitEthernet2	EFP_205
-----	------------------	---------

206	GigabitEthernet2	EFP_206
-----	------------------	---------

305	GigabitEthernet3	EFP_305
-----	------------------	---------

306	GigabitEthernet3	EFP_306
-----	------------------	---------

