DevNet Coffee Break
IOS XE RESTCONF

Cisco Connect Croatia 2017

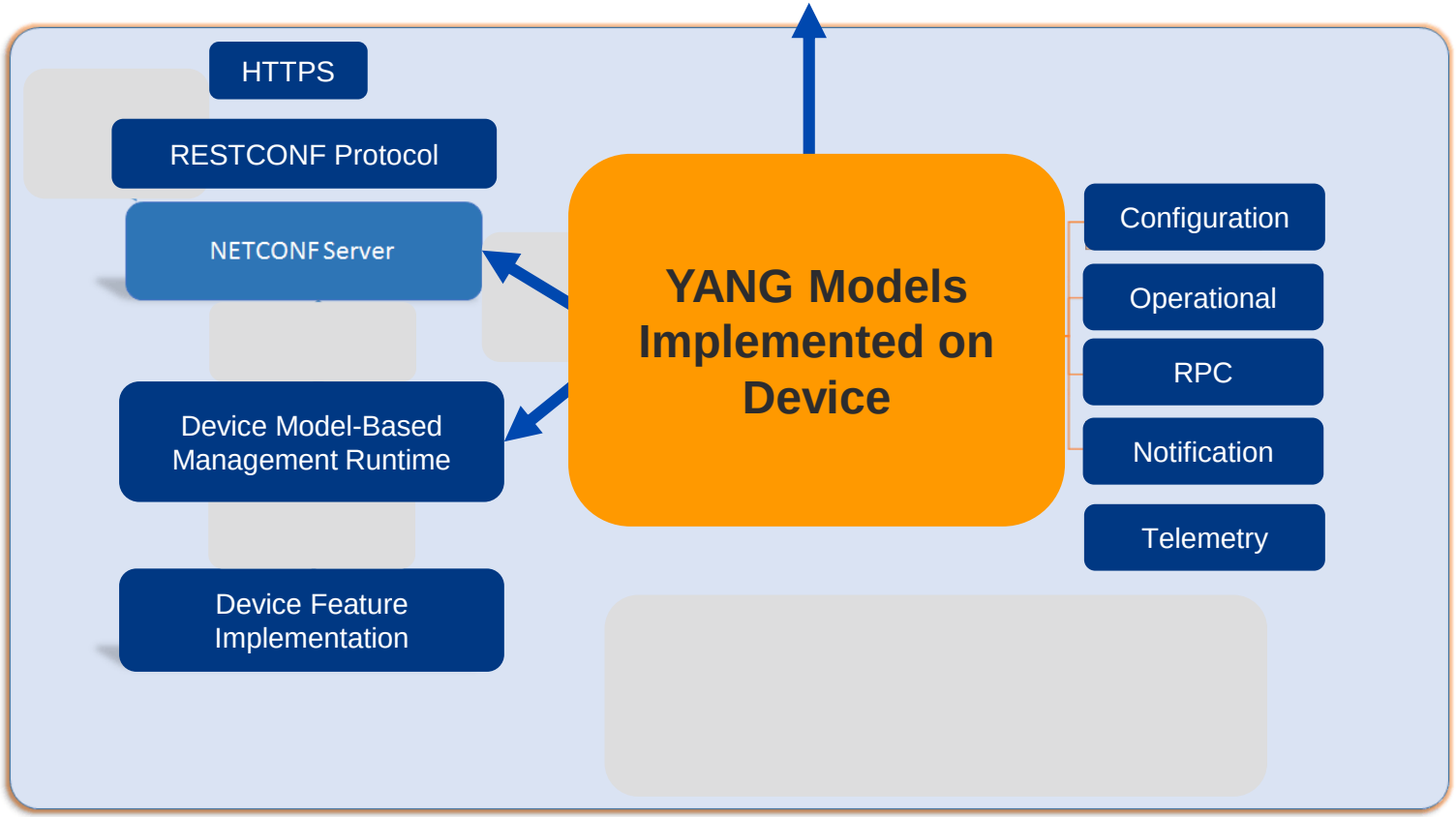
Djordje Vulovic
Consulting Systems Engineer
dvulovic@cisco.com

https://github.com/djordjevulovic/Cisco_Connect_HR_2017

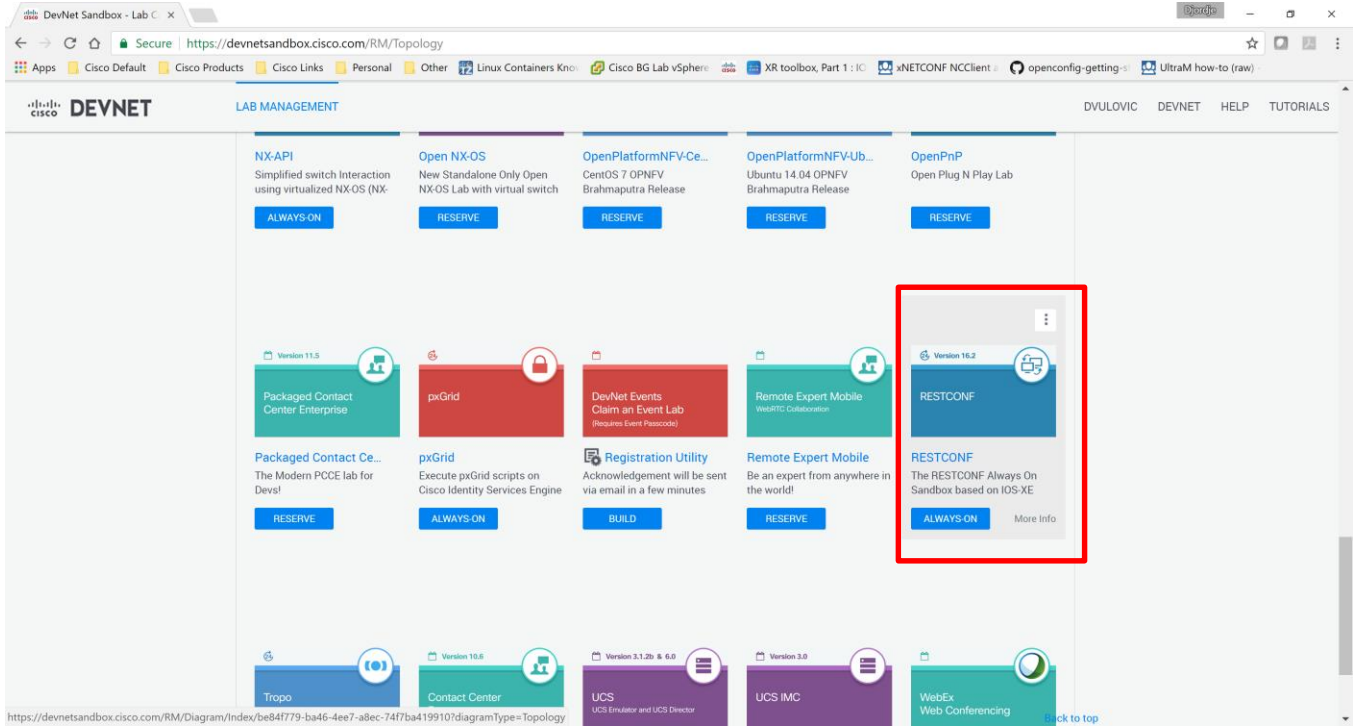
What is RESTCONF ?

- 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

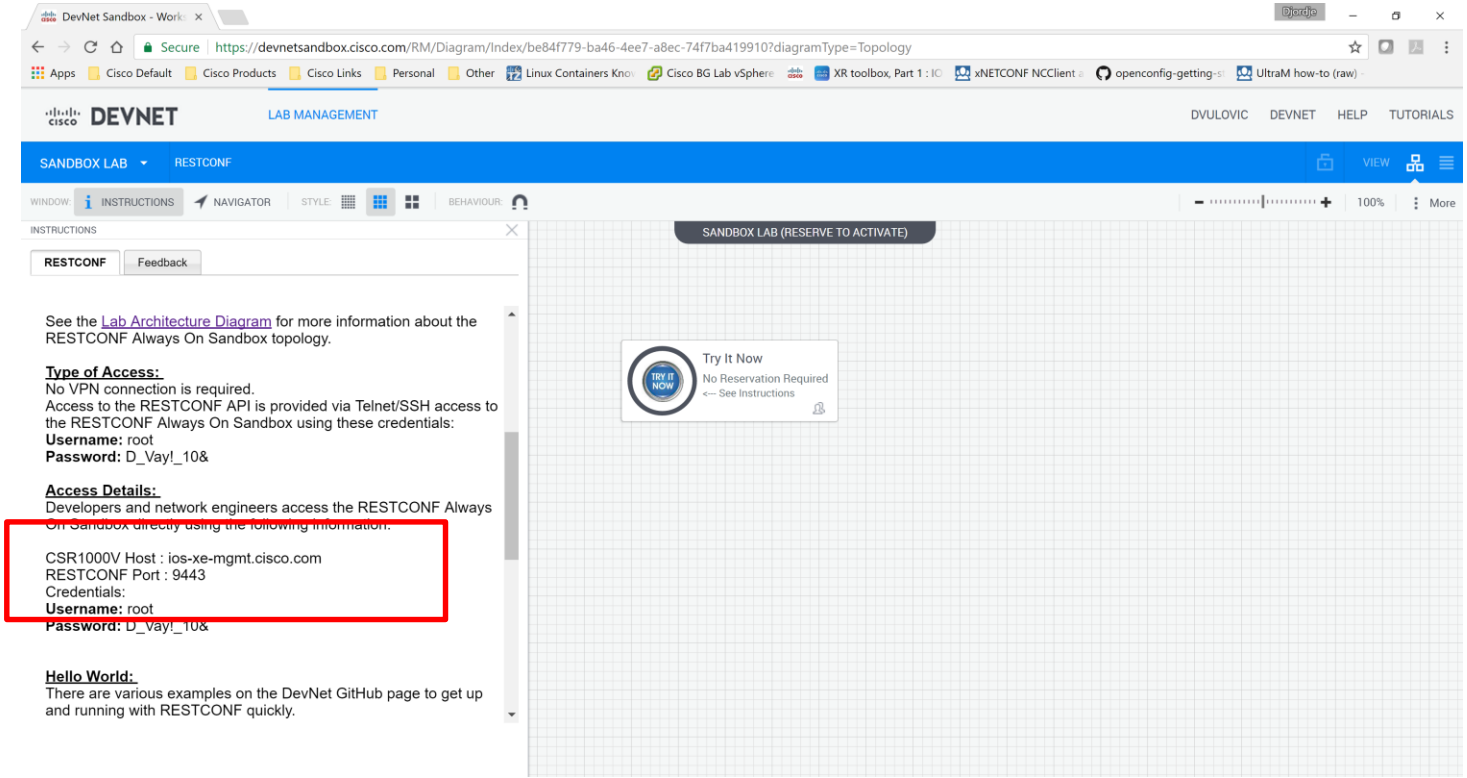


DevNet Sandbox Labs

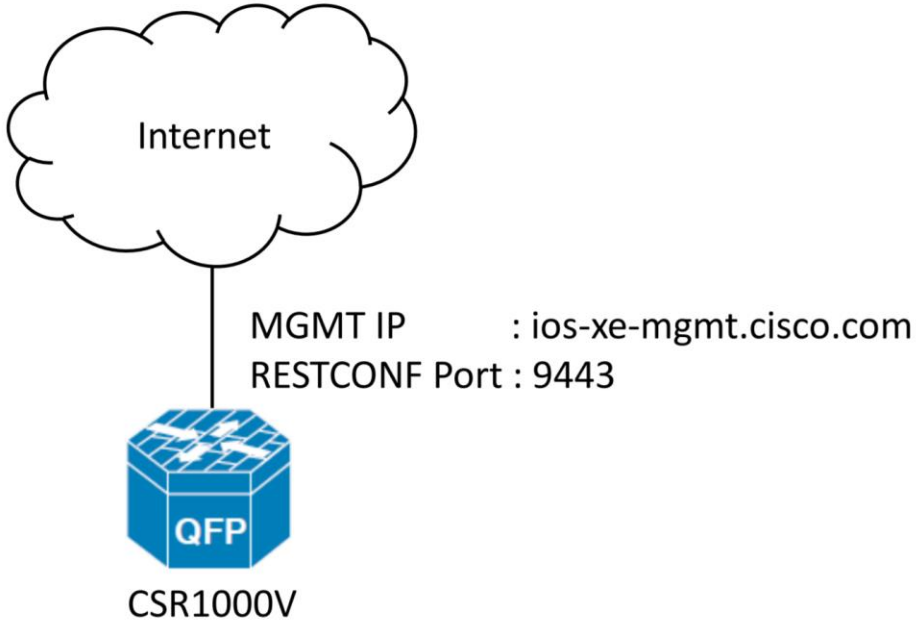


https://devnetsandbox.cisco.com

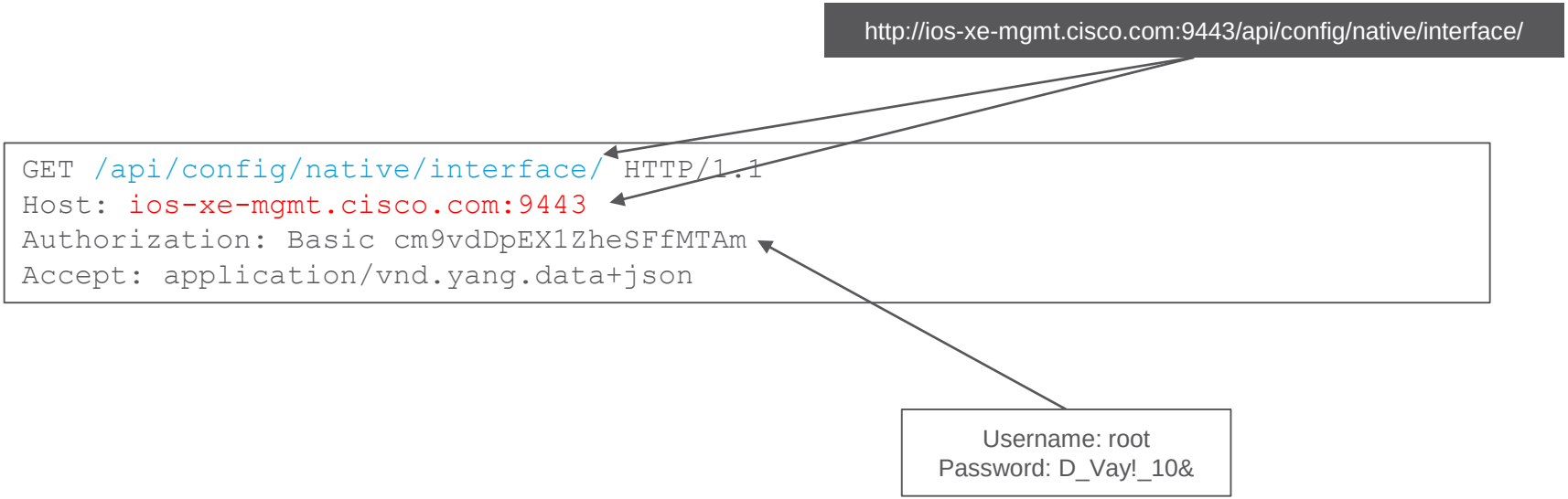
DevNet RESTCONF SandBox



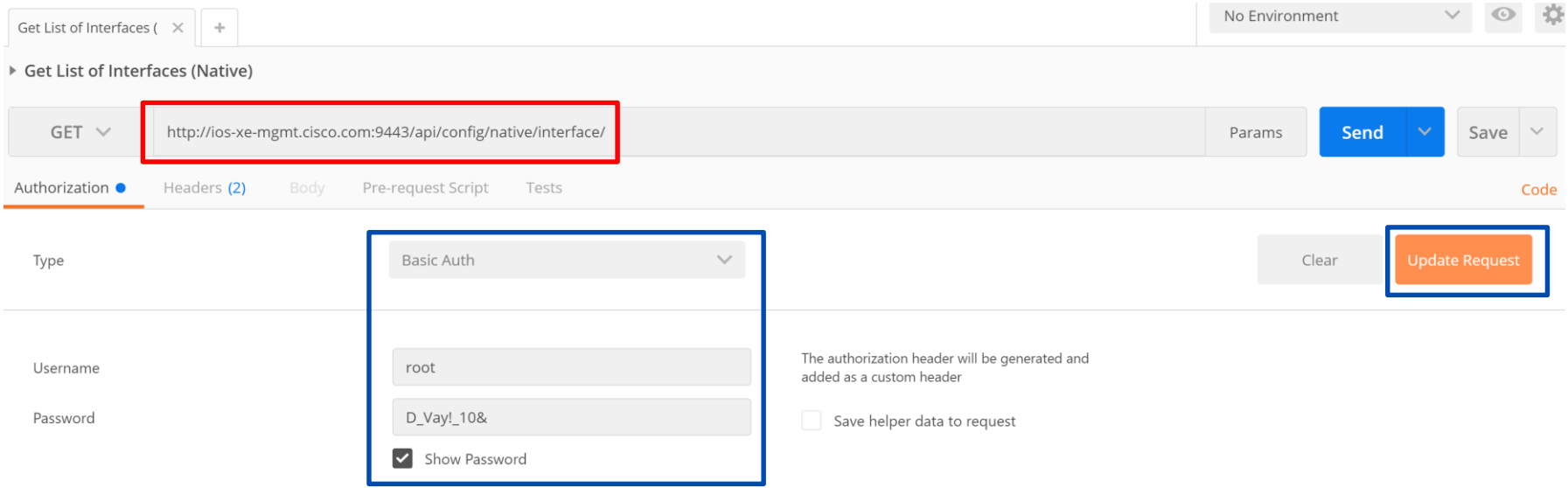
RESTCONF DevNet Sandbox Diagram



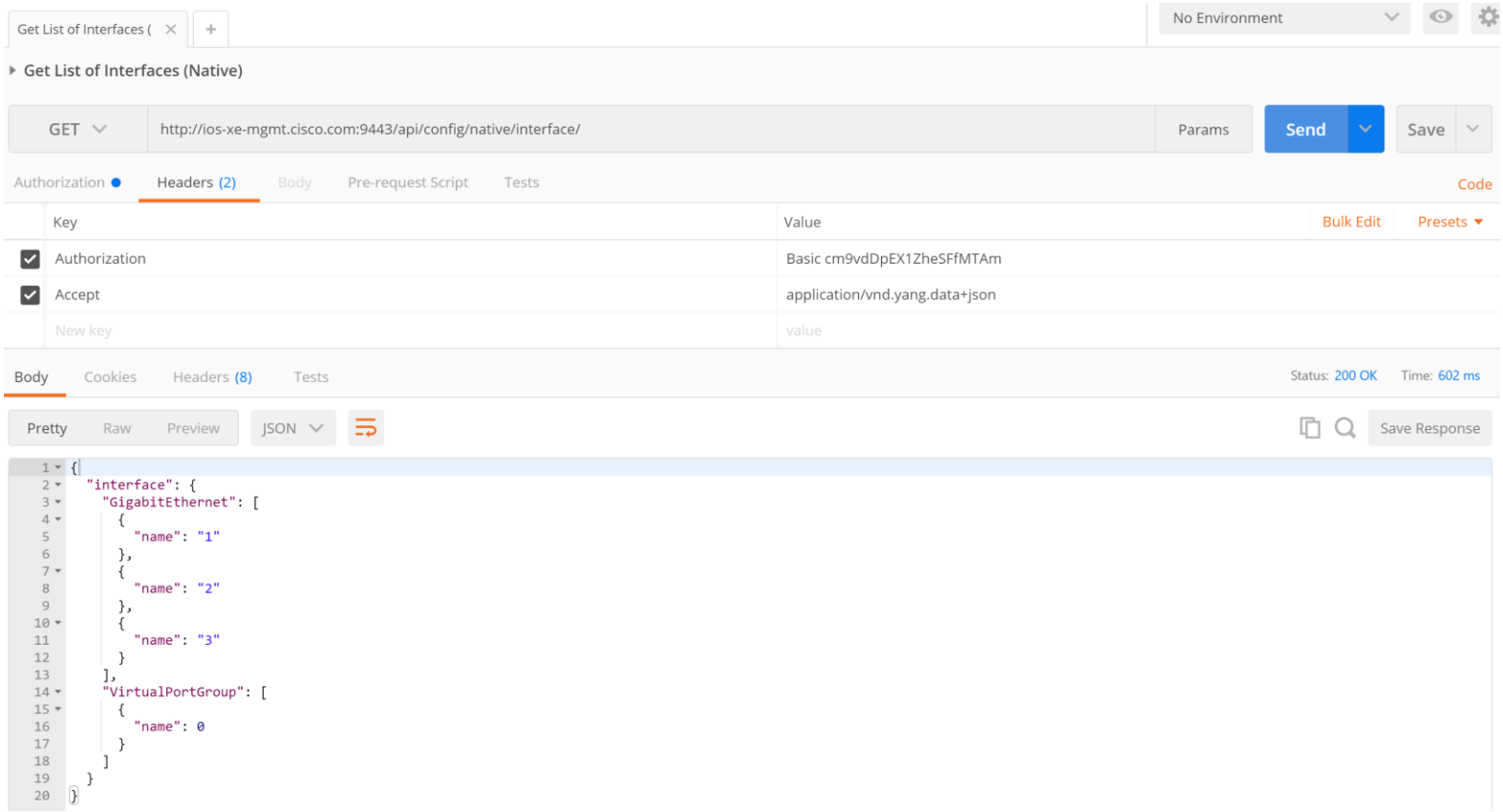
“Get Interface List” Request (Native Model)



Build Request in Postman Tool



Execute Request in Postman Tool



“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 cm9vdDpEX1ZheSFfMTAm
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
```


CISCO