配置MPLS在PE路由器的L3VPN服务使用REST-API (IOS-XE)

目录

简介

本文展示编程的使用Python设置在服务运营商边缘(PE)使用其余API，路由器的一个MPLS L3VPN。此示例使用思科CSR1000v (IOS-XE)路由器作为PE路由器。

Anuradha Perera

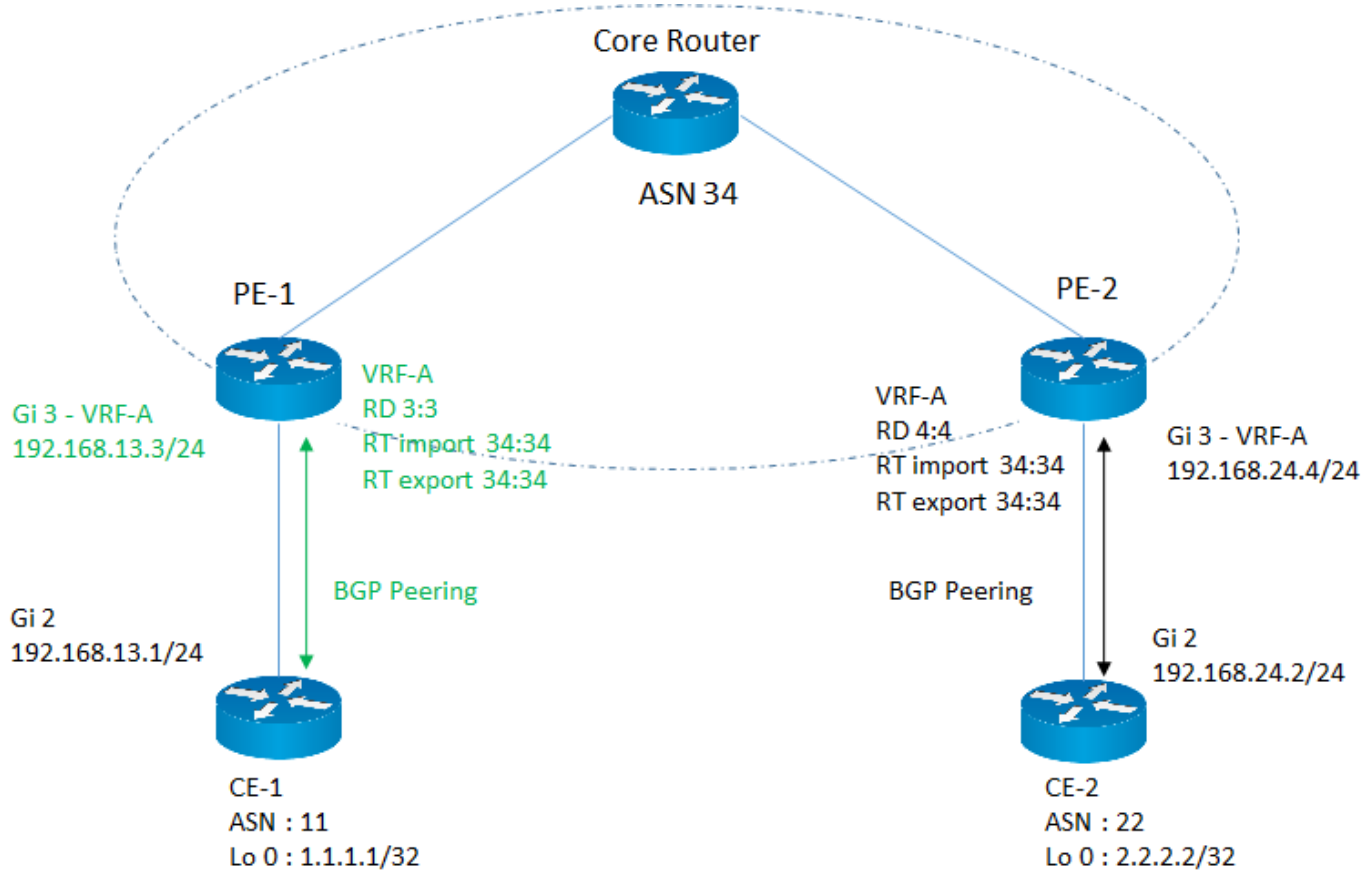
Sridhar

先决条件

- 其余对CSR1000v路由器的API管理访问(pleae参考参考在本文结束时)。
- 在计算机安装的Python (版本2.x或3.x)和“请求” Python库使用配置路由器。
- 若干基础知识Python编程。

配置

网络图



在此示例重点在配置在PE-1路由器的需要的MPLS L3VPN服务参数，用粉红色颜色突出显示。

配置过程

配置任务分开对一定数量的子任务，并且每个子任务实现在用户定义的功能下。这样可以重新使用功能，当要求。

所有功能使用“请求”库访问路由器的基于API，并且数据格式是JSON。 在HTTP请求“请验证”参数设置对“”忽略验证的错误SSL证书。

1. 获取标记id

在继续进行在路由器前的所有配置您需要有从路由器获取的有效标记id。此功能启动HTTP请求验证和获取标记id使用此标记，以便能调用其他API。此请求答复包括标记id。

```
#~#----------------------------------

def getToken (ip，端口、用户名，密码）：

导入请求

导入base64

 URL = “https://” + ip +” ：“+端口+ "/api/v1/auth/token-services"

 报头= {

  ‘内容类型’：“应用程序/json”，

  ‘授权’：“基本” + base64.b64encode((username + “ ：” +密码) .encode ('UTF-8')).decode
(‘ascii’)，
```

‘缓存控制’：“no-cache”

}

答复= requests.request (“POST”， URL，错误headers=headers的verify=)

如果response.status_code == 200 ：

回归response.json () ['token-id']

：

返回“失败”

#~#----------

2. 创建VRF

此功能将创建在PE路由器的VRF与需要的路由辨别器(RD)并且导入/输出路由目标(RT)

#~#----------

def createVRF (ip，端口、tokenID， vrfname， RD， importRT， exportRT) ：

导入请求

URL = “https://” + ip +” ：“+端口+ "/api/v1/vrf"

报头= {

‘内容类型’：“应用程序/json”，

‘X验证标记’：tokenID，

‘缓存控制’：“no-cache”

}

数据= {

‘名称’：vrfname，

‘rd’：RD，

‘route-target’：[

{

‘操作’：“导入”，

'社区'：importRT

},

{

'操作'："出口"，

'社区'：exportRT

}

]

}

答复= requests.request ("POST"， URL， headers=headers、错误json=data的verify=)

如果response.status_code == 201 ：

"成功"的返回

：

返回"失败"

#~#----------

3.移动接口到VRF里

此功能将搬入指定接口VRF。

#~#----------

def addInterfacetoVRF (ip，端口、tokenID、vrfname， interfaceName， RD， importRT， exportRT) ：

导入请求

URL = "https://" + ip +" ："+端口+ "/api/v1/vrf/" + vrfname

报头= {

'内容类型'："应用程序/json"，

'X验证标记'：tokenID，

'缓存控制'："no-cache"

}

数据= {

‘rd’：RD，

‘转发’：[interfaceName]，

‘route-target’：[

{

‘操作’：“导入”，

‘社区’：importRT

}，

{

‘操作’：“出口”，

‘社区’：exportRT

}

]

}

答复= requests.request (“PUT”， URL， headers=headers、错误json=data的verify=)

如果response.status_code == 204 ：

“成功”的返回

：

返回“失败”

#~#----------

4. 分配IP地址建立接口

此功能将分配IP地址到接口。

#~#----------

def assignInterfaceIP (ip，端口、tokenID、interfaceName、interfaceIP， interfaceSubnet) ：

导入请求

URL = “https://” + ip +” ：“+端口+ "/api/v1/interfaces/" + interfaceName

报头= {

‘内容类型’：“应用程序/json”，

‘X验证标记’：tokenID，

‘缓存控制’：“no-cache”

}

数据= {

‘类型’：“以太网”，

‘如果NAME’：interfaceName，

‘IP地址’：interfaceIP，

‘子网掩码’：interfaceSubnet

}

答复= requests.request (“PUT”， URL， headers=headers、错误json=data的verify=)

如果response.status_code == 204 ：

“成功”的返回

：

返回“失败”

#~#----------

5. 创建VRF意识bgp

这将启用VRF地址家族ipv4。

#~#----------

def createVrfBGP (ip，端口、tokenID， vrfname， ASN) ：

导入请求

URL = “https://” + ip +” ：“+端口+ "/api/v1/vrf/" + vrfname + “/routing-svc/bgp”

报头= {

‘内容类型’：“应用程序/json”，

‘X验证标记’：tokenID，

‘缓存控制’：“no-cache”

}

数据= {

'路由协议id'：ASN

}

答复= requests.request ("POST"， URL， headers=headers、错误json=data的verify=)

如果response.status_code == 201 ：

"成功"的返回

：

返回"失败"

#~#----------

6. 定义BGP邻居在VRF地址家族下

此功能将定义BGP邻居在VRF地址家族IPV4下。

#~#----------

def defineVrfBGPNeighbour (ip，端口、tokenID， vrfname， ASN， neighbourIP， remoteAS) ：

导入请求

URL = "https://" + ip +" ："+端口+ "/api/v1/vrf/" + vrfname + "/routing-svc/bgp/ "+ ASN +" /neighbors"

报头= {

'内容类型'："应用程序/json"，

'X验证标记'：tokenID，

'缓存控制'："no-cache"

}

数据= {

'路由协议id'：ASN，

'地址的：neighbourIP，

'远程AS'：remoteAS

}

答复= requests.request (“POST”， URL， headers=headers、错误json=data的verify=)

如果response.status_code == 201 ：

“成功”的返回

：

返回“失败”

#~#----------

说明和值输入参数

ip = "10.0.0.1" #路由器的IP地址

port= "55443" #其余路由器的API端口

username= “cisco” #登陆的用户名。应该配置这与权限级别15。

密码= “cisco” #密码关联与用户名

tokenID = <value returned> #从路由器获取的标记ID使用getToken功能

vrfname = “VRF-A” # VRF的名称

RD = "3:3" # VRF的路由辨别器

importRT = "34:34" #导入路由目标

exportRT = "34:34" #输出路由目标

interfaceName = "GigabitEthernet3" #面对接口的用户边缘(CE)的名称

interfaceIP = "192.168.13.3" #面对接口的CE的IP地址

interfaceSubnet = "255.255.255.0" #面对接口的CE子网

ASN = "34" # PE路由器BGP AS编号

neighbourIP = "192.168.13.1" # BGP CE路由器对等体IP

remoteAS = "11" # CE路由器AS编号

在所有上述功能，专用的API为每配置setp呼叫。下面的示例展示如何通过IOS-XE CLI，一般来说，在其余API呼叫正文。如果特定API不是可用的，这可以用于作为应急方案自动化。在上述功能的内容类型'设置为'应用程序/json'，但是在下面的示例， '内容类型'设置为'文本/纯文本'，当解析标准的CLI输入。

此示例定义了接口的GigabitEthernet3接口说明。 配置可以通过更改“cliInput”参数定制。

#~#----------

```
def passCLIInput (ip，端口， tokenID) ：

导入请求

 URL = “https://” + ip +” ：“+端口+ "/api/v1/global/running-config"

 报头= {

  ‘内容类型’：“文本/纯文本”，

 ‘X验证标记’：tokenID，

  ‘缓存控制’：“no-cache”

  }

 line1 = “Interface gigabitethernet 3"

 line2 = “面对接口的说明客户”

 cliInput = line1 + “\ r \ n” + line2

 答复= requests.request (“PUT”， URL， headers=headers、错误data=cliInput的verify=)

 print(response.text)

如果response.status_code == 204 ：

  “成功”的返回

 :

  返回“失败”

#~#----------------------------------
```

参考

- Cisco CSR 1000v系列Cloud服务路由器软件配置指南

https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/configuration/b_CSR1000v_Configuration_Guide /b_CSR1000v_Configuration_Guide_chapter_01101.html

- Cisco IOS XE其余API管理参考指南

https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/restapi/restapi.html

使用的缩略语：

MPLS -多协议标签交换

L3 -第3层

VPN -虚拟专用网络

VRF -虚拟路由转发

BGP -边界网关协议

其余-代表状态转移

API -应用程序结口

JSON -Java脚本对象符号

HTTP -超文本传输协议