Konfigurieren des MPLS-L3VPN-Services auf dem PE-Router mithilfe der REST-API (IOS-XE)

Inhalt

Einleitung

In diesem Dokument wird die Verwendung der Python-Programmierung zur Bereitstellung eines MPLS-L3VPN auf einem PE-Router (Service Provider Edge) mithilfe der REST-API veranschaulicht. In diesem Beispiel werden Cisco CSR1000v (IOS-XE) Router als PE-Router verwendet.

Verfasst von: Anuradha Perera

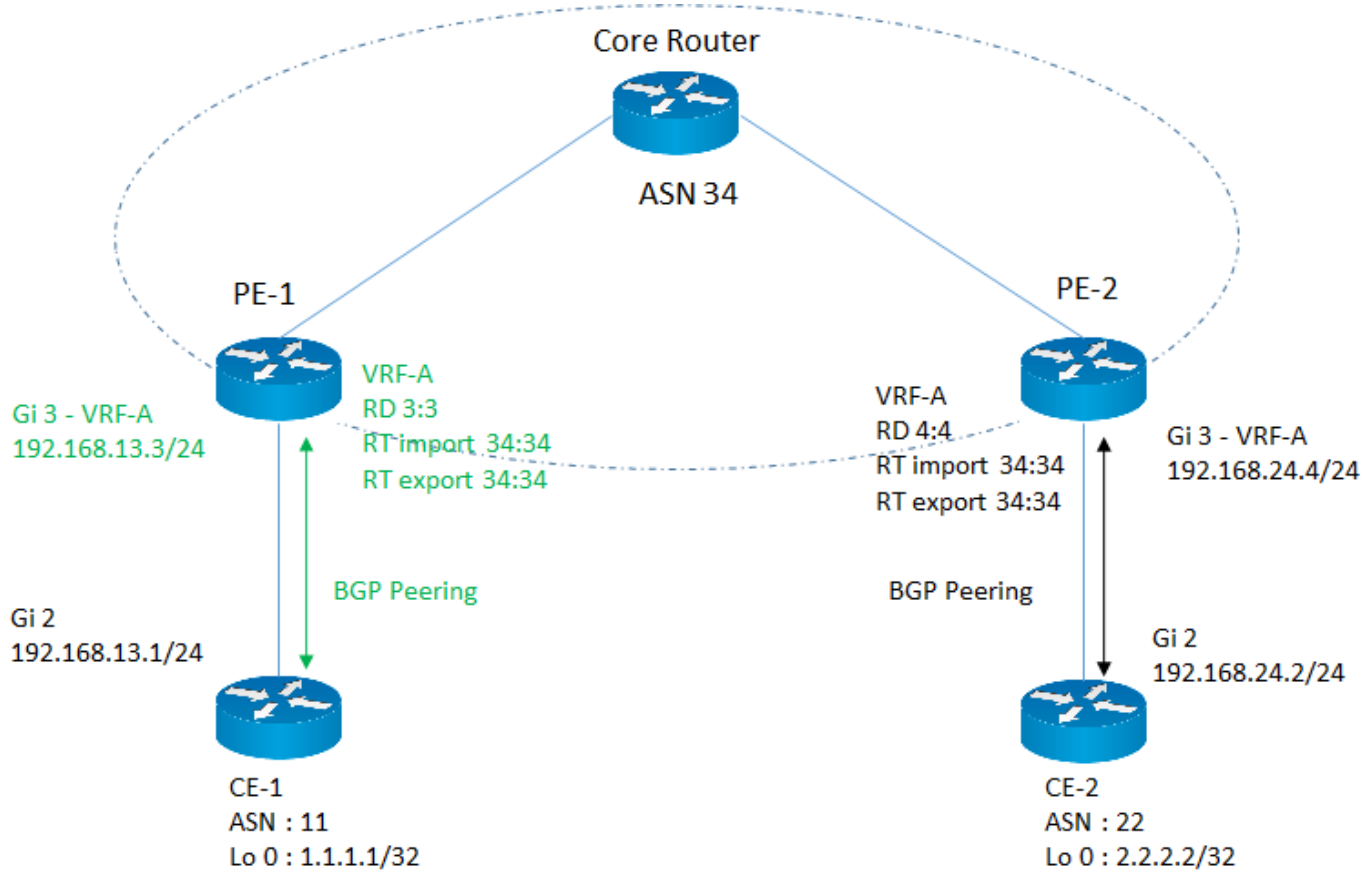
Herausgegeben von: Kumar Sridhar

Voraussetzungen

- REST API-Verwaltungszugriff auf CSR1000v-Router (weitere Informationen finden Sie im Abschnitt Referenzen am Ende dieses Dokuments).
- Python (Version 2.x oder 3.x) und die Python-Bibliothek, die auf dem für die Konfiguration der Router verwendeten Computer installiert wird, werden angefordert.
- Einige Grundkenntnisse Python-Programmierung.

Konfiguration

Netzwerkdiagramm



In diesem Beispiel liegt der Schwerpunkt auf der Konfiguration der erforderlichen MPLS-L3VPN-Serviceparameter auf dem PE-1-Router, die rosa hervorgehoben sind.

Konfigurationsverfahren

Die Konfigurationsaufgabe ist in mehrere Teilaufgaben unterteilt, und jede Teilaufgabe wird unter einer benutzerdefinierten Funktion implementiert. Auf diese Weise können Funktionen bei Bedarf wiederverwendet werden.

Alle Funktionen nutzen die Bibliothek "Requests" für den Zugriff auf REST APIs auf dem Router und das Datenformat ist JSON. Bei HTTP-Anfragen wird der Parameter "verify" auf "False" gesetzt, um die Validierung des SSL-Zertifikats zu ignorieren.

1. Token-ID abrufen

Bevor Sie mit der Konfiguration eines Routers fortfahren können, benötigen Sie eine gültige Token-ID, die Sie vom Router erhalten. Diese Funktion initiiert eine HTTP-Anforderung zur Authentifizierung und zum Abrufen einer Token-ID, sodass sie andere APIs aufrufen kann, die dieses Token verwenden. Die Antwort dieser Anforderung enthält eine Token-ID.

```
#---------------------------------

def getToken (ip, port, username, password):

Anträge auf Einfuhr

Einfuhrbasis64

 url = "https://" + ip + ":"+ port + "/api/v1/auth/token-services"

 Header = {

  'Inhaltstyp': "Anwendung/JSON",

  'Authorization': "Basic " + base64.b64encode((username + ":" + password).encode('UTF-
```

```
8')).decode('ascii'),
    'cache-control': "no-cache"
 }
 response = requirements.request("POST", url, headers=headers, verify=False )
if response.status_code == 200:
 return response.json()['token-id']
Sonstiges:
 zurückbringen "Fehlgeschlagen"
#---------------------------------
```

2. Erstellen von VRF

Diese Funktion erstellt die VRF-Instanz auf dem PE-Router mit dem erforderlichen Route Distinguisher (RD) und Import-/Export-Route Targets (RT).

```
#---------------------------------
def createVRF (IP, Port, TokenID, VRFName, RD, ImportRT, ExportRT):
Anträge auf Einfuhr
 url = "https://" + ip + ":"+ port + "/api/v1/vrf"
 Header = {
   'Inhaltstyp': "Anwendung/JSON",
   'X-auth-token': tokenID,
   'cache-control': "no-cache"
 }
 data = {
   'name': vrfName,
   'rd': RD,
   'route-target': [
      {
```

```
        'Aktion': "Import",
        'community' : importRT
    },
    {
        'Aktion': "Ausfuhr",
        'community' : exportRT
    }
  ]
}
 response = requirements.request("POST", url, headers=headers, json=data, verify=False )
if response.status_code == 201:
  zurückbringen "erfolgreich"
Sonstiges:
  zurückbringen "Fehlgeschlagen"
#---------------------------------
```

3. Schnittstelle in VRF verschieben

Diese Funktion verschiebt eine bestimmte Schnittstelle in eine VRF-Instanz.

```
#---------------------------------
def addInterfacetoVRF (IP, Port, TokenID, VRFName, Schnittstellenname, RD, ImportRT,
ExportRT):
Anträge auf Einfuhr
 url = "https://" + ip + ":"+ port + "/api/v1/vrf/" + vrfName
 Header = {
  "Inhaltstyp": "Anwendung/JSON",
  'X-auth-token': tokenID,
  'cache-control': "no-cache"
```

```
  }
 data = {
  'rd': RD,
  'forwarding': [ Schnittstellenname ],
  'route-target': [
     {
        'Aktion' : "Einfuhr",
        'community' : importRT
     },
     {
        'Aktion' : "Ausfuhr",
        'community' : exportRT
     }
  ]
 }
 response = requirements.request("PUT", url, headers=headers, json=data, verify=False )


if response.status_code == 204:
  zurückbringen "erfolgreich"
Sonstiges:
  zurückbringen "Fehlgeschlagen"
#---------------------------------
```

4. IP-Adresse Schnittstelle zuweisen

Diese Funktion weist der Schnittstelle die IP-Adresse zu.

```
#---------------------------------
def assignInterfaceIP (IP, Port, TokenID, Schnittstellenname, SchnittstelleIP,
SchnittstelleSubnetz):
```

```
Anträge auf Einfuhr
 url = "https://" + ip + ":"+ port + "/api/v1/interfaces/" + interfaceName
 Header = {
   "Inhaltstyp": "Anwendung/JSON",
   'X-auth-token': tokenID,
   'cache-control': "no-cache"
   }
 data = {
   'Typ': "Ethernet",
   'if-name': interfaceName,
   'ip-address': interfaceIP,
   'subnet mask': SchnittstelleSubnetz
   }
 response = requirements.request("PUT", url, headers=headers, json=data, verify=False )
if response.status_code == 204:
   Return "erfolgreich"
Sonstiges:
   Rückgabe "fehlgeschlagen"
#---------------------------------
```

5. Erstellen Sie ein VRF-fähiges BGP

Damit wird IPv4 für die VRF-Adressfamilie aktiviert.

```
#---------------------------------
def createVrfBGP (IP, Port, TokenID, VRFName, ASN):
Anträge auf Einfuhr
 url = "https://" + ip + ":"+ port + "/api/v1/vrf/" + vrfName + "/routing-svc/bgp"
 Header = {
   'Inhaltstyp': "Anwendung/JSON",
```

```
  'X-auth-token': tokenID,
  'cache-control': "no-cache"
  }
 data = {
  'routing-protocol-id': ASN
  }
 response = requirements.request("POST", url, headers=headers, json=data, verify=False )
if response.status_code == 201:
  zurückbringen "erfolgreich"
Sonstiges:
  zurückbringen "Fehlgeschlagen"
#---------------------------------
```

6. Definieren Sie einen BGP-Nachbarn unter der VRF-Adressfamilie.

Diese Funktion definiert einen BGP-Nachbarn unter der VRF-Adressfamilie IPV4.

```
#---------------------------------
def defineVrfBGPNachtbor (ip, port, tokenID, vrfName, ASN, neighbourIP, remoteAS):
Anträge auf Einfuhr
 url = "https://" + ip + ":"+ port + "/api/v1/vrf/" + vrfName + "/routing-svc/bgp/" + ASN +"/neighbors"
 Header = {
  "Inhaltstyp": "Anwendung/JSON",
  'X-auth-token': tokenID,
  'cache-control': "no-cache"
  }
 data = {
  'routing-protocol-id': ASN,
  'Adresse': NeighbourIP,
  'remote-as': remoteAS
```

```
  }
 response = requirements.request("POST", url, headers=headers, json=data, verify=False )
 if response.status_code == 201:
   zurückbringen "erfolgreich"
Sonstiges:
   zurückbringen "Fehlgeschlagen"
#---------------------------------
```

Beschreibung und Werte der Eingabeparameter

```
IP = "10.0.0.1"                          # IP-Adresse des Routers
Port = "55443"                           # REST API-Port auf Router
username = "cisco"                       # Benutzername für die Anmeldung.Dies sollte mit der Berechtigungsstufe 15 konfiguriert werden.
password = "cisco"                       # Kennwort zugeordnet zum Benutzernamen
tokenID = <Wert zurückgegeben># Token-ID, die vom Router mithilfe der getToken-Funktion abgerufen wurde
vrfName = "VRF-A"# Name der VRF-Instanz
RD = "3:3"# Route Distinguisher für VRF
importRT = "34:34" # Import Route Target
exportRT = "34:34" # export Route Target
interfaceName = "GigabitEthernet3"# Name der CE-Schnittstelle (Customer Edge)
interfaceIP = "192.168.13.3" # IP-Adresse der CE-seitigen Schnittstelle
interfaceSubnet = "255.255.255.0"# Subnetz der CE-seitigen Schnittstelle
ASN = "34"# BGP-AS-Nummer des PE-Routers
neighbourIP = "192.168.13.1"# BGP-Peering-IP des CE-Routers
remoteAS = "11"# AS-Nummer des CE-Routers
```

In allen oben genannten Funktionen wurden dedizierte APIs für jeden Konfigurationsschritt aufgerufen. Im folgenden Beispiel wird veranschaulicht, wie die IOS-XE-CLI im Allgemeinen im Text des REST-API-Aufrufs übergeben wird. Dies kann als Workaround für die Automatisierung genutzt werden, wenn eine

bestimmte API nicht verfügbar ist. In den obigen Funktionen ist "content-type" auf "application/json" gesetzt, im folgenden Beispiel ist "content-type" auf "text/plain" gesetzt, da es die standardmäßige CLI-Eingabe analysiert.

In diesem Beispiel wird eine Schnittstellenbeschreibung für die Schnittstelle GigabitEthernet3 definiert. Die Konfiguration kann durch Ändern des Parameters "cliInput" angepasst werden.

```
#---------------------------------

def passCLInput (ip, port, tokenID):

Anträge auf Einfuhr

 url = "https://" + ip + ":"+ port + "/api/v1/global/running-config"

 Header = {

  'Inhaltstyp': "Text/Nur",

 'X-auth-token': tokenID,

  'cache-control': "no-cache"

  }

 line1 = "Interface GigabitEthernet 3"

 line2 = "description Customer Facing Interface" (Kundenschnittstelle)

 cliInput = line1 + "\r\n" + line2

 response = requirements.request("PUT", url, headers=headers, data=cliInput, verify=False )

 print(response.text)

if response.status_code == 204:

  zurückbringen "erfolgreich"

Sonstiges:

  zurückbringen "Fehlgeschlagen"

#---------------------------------
```

Referenzen

- Software-Konfigurationsleitfaden für Cisco Cloud Services Router der Serie CSR 1000v

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 REST API Management-Referenzhandbuch

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

Verwendete Akronyme:

MPLS = Multi Protocol Label Switching

L3 - Layer 3

VPN = Virtual Private Network

VRF = Virtual Route Forwarding

BGP = Border Gateway Protocol

REST = Representational State Transfer

API - Application Program Interface

JSON - Java Script Object Notation

HTTP = Hyper Text Transfer Protocol

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.