LES TP CISCO

TP1

7.0.3-LAB---INSTALL-THE-CSR1000V-VM_FR-FR.PDF

Ici tout vas bien, c'est normal il est en anglais.

TP2

7.6.3-LAB---AUTOMATED-TESTING-USING-PYATS-AND-GENIE_FR-FR.PDF

Ici de meme, tout vas bien le seul probléme etant le code est mal interpreter par le pdf ce qui cause la creation d'espace lors du copier, coller.

TP3

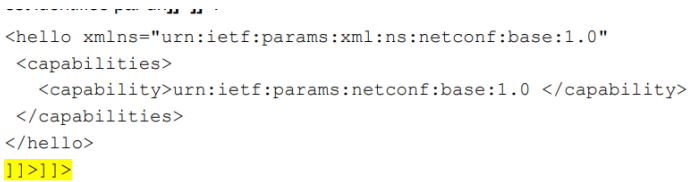
8.3.5-LAB---EXPLORE-YANG-MODELS_FR-FR.PDF

Ici tout vas bien.

TP4

8.3.6-LAB---USE-NETCONF-TO-ACCESS-AN-IOS-XE-DEVICE_FR-FR.PDF

Probleme a la page 4, il manque une ">".



```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
 <capabilities>
   <capability>urn:ietf:params:netconf:base:1.0 </capability>
 </capabilities>
</hello>
]]>]]>
```

TP5

8.3.7-LAB---USE-RESTCONF-TO-ACCESS-AN-IOS-XE-DEVICE_FR-FR.PDF

Premier probleme page 7, des simples quotes a la place des doubles sont utiliser.

f. Retournez à Postman et envoyez à nouveau votre demande GET. Vous devriez maintenant voir les informations d'adressage IPv4 dans la réponse JSON, comme indiqué ci-dessous. Dans la partie suivante, vous allez copier ce format JSON pour créer une nouvelle interface.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "description": "VBox",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "192.168.56.101",
          ''netmask'': ''255.255.255.0''
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

Partie 5 : Utiliser Postman pour envoyer une demande de PUT

De même page 8.

pouvez également copier ce qui suit dans la section Corps de votre demande PUT. Notez que le type d'interface doit être défini sur **SoftwareLoopback**.

```
{
  "ietf-interfaces:interface": {
    "name": "Loopback1",
    "description": "My first RESTCONF loopback",
    "type": "iana-if-type:softwareLoopback",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "10.1.1.1",
          ''netmask'': ''255.255.255.0''
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

c. Cliquez sur **Send** pour envoyer la demande PUT au CSR1kv. Sous la section Corps, vous devriez voir le code de réponse HTTP **Status: 201 Created**. Cela indique que la ressource a été créée avec succès.

De même pages 12.

```
basicauth = ("cisco", "cisco123!")
```

d. Créez une variable de dictionnaire Python **YangConfig** qui contiendra les données YANG requises pour créer la nouvelle interface Loopback2. Vous pouvez utiliser le même dictionnaire que vous avez utilisé précédemment dans Postman. Cependant, modifiez le numéro et l'adresse de l'interface. Aussi, sachez que les valeurs booléennes doivent être capitalisées en Python. Par conséquent, assurez-vous que le **T** est mis en majuscule dans la paire clé/valeur pour **"enabled": True**.

```
YangConfig = {
    "ietf-interfaces:interface": {
        "name": "Loopback2",
        "description": "My second RESTCONF loopback",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "10.2.1.1",
                    ''netmask'': ''255.255.255.0''
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}
```

Ici, pages 12, headers a été traduit.

a. Avant d'entrer des instructions, veuillez noter que cette spécification de variable ne doit être que sur une seule ligne de votre script. Entrez les instructions suivantes :

Remarque : Cette spécification de variable doit se trouver sur une ligne dans votre script.

```
resp = requests.put (api_url, data=json.dumps (YangConfig), auth=basicauth, en-têtes
=, verify=false)
```

Ici, encore pages 12, il manque des "\".

b. Entrez le code ci-dessous pour gérer la réponse. Si la réponse est l'un des messages de succès HTTP, le premier message sera imprimé. Toute autre valeur de code est considérée comme une erreur. Le code de réponse et le message d'erreur seront imprimés dans le cas où une erreur a été détectée.

```
if(resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print('Error. Code d'état : {} \ nMessage d'erreur : {} '.format
(resp.status_code, resp.json ()))
```

Correction :

```
if(resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print('Error. Code d\'état : {} \n Message d\'erreur : {} '.format(resp.status_code, resp.json()))
```