ご利用のコンピュータを設定する方法

このラボの作業を行うには、事前設定済みの dCloud ラボを使用するか、自身のコンピュータを セットアップしてください。詳細については、<u>イベントの準備</u>[英語]モジュールと<u>ラボの設定</u> [英語]モジュールを確認してください。

APIC-EM REST API

APIC-EM のノースバウンド(NB) REST ベースの API により、ネットワークエンジニアとオペレータは、カスタムアプリケーションを使用してコントローラと接続し、ネットワーク状況の変化や関連するビジネス チャンスに迅速に対応できるようになります。

APIC-EM サウスバウンド インターフェイスはネットワークの制御プレーンに接続するもので、 直接のアクセスはできません。これは、ノースバウンド(NB) REST API を通じて特定されたネッ トワーク デバイスでコマンドを実行するために使用されます。したがって、APIC-EM と通信して 動的な SDN 機能をネットワークに直接追加するように、アプリケーションを設計できます。

このラボでは、Python プログラミング言語によって Cisco APIC-EM ノースバウンド REST API を 使用する方法を示します。

目標

所要時間:35分

- Postman REST クライアントを使用して Python コードを生成する方法を学習する
- Python を使用してカスタム スクリプトを記述する

前提条件

このモジュールでは、REST API コールを発信する Google Chrome ブラウザのアプリケーション である <u>Postman</u> を使用します。

また、プログラミング言語として Python3 を使用します。

背景:このラボを開始する前に、ラーニングラボのモジュールを完了しておくことを強くお勧めします。

- <u>ラボの設定</u>[英語]
- <u>REST API および Python</u> [英語]

詳細については、「自身のコンピュータを設定する方法」の上のボックスをクリックしてください。

APIC-EM コントローラへのアクセス

- これらのサンプルコードを実行するためには、APIC-EMコントローラにアクセスできる 必要があります。
- APIC-EM コントローラをまだ導入していない場合は、Cisco dCloud から提供された APIC-EM コントローラを使用できます。
 - 。 APIC-EM コントローラを利用できる dCloud に接続する方法については、<u>ラボの</u> セットアップ [英語] モジュールを参照してください。
 - 。 接続すると、IP アドレス <u>https://198.18.129.100/</u>を使用して APIC-EM コントロー ラにアクセスできるようになります。コントローラのログイン クレデンシャルは、 **ユーザ名:**admin、パスワード:C1sco12345 です。

ステップ 1:APIC-EM に接続する

Web ブラウザを開き、<u>https://198.18.129.100/</u>の Web ページに移動します。SSL 証明書の警告 ページが表示されます。続行するには、[詳細(Advanced)] ボタンをクリックして、[198.18.129.100 に進む(Proceed to 198.18.129.100)] リンクをクリックします。



ログインページで、ユーザ名 admin、パスワード C1sco12345 を使用して認証します。

← → C		☆ 〓
،،۱۱،،۱۱، cısco		
	APIC-EM	
Cisco Ap	plication Policy Infrastructure Controller Enterprise Module	
	admin	
© 2009-2015 Cisco Systems, Inc., Cisco, Cisco Systems, and C owned by other third parties and used and distributed under	Tisco Systems logo are registered trademarks of Cisco Systems, inc. and/or its affiliates in the U.S. and certain other countries. The copyrights to certain works contained in this license. Certain components of this software are licensed under the GNU <u>GPL 2.0, GPL 3.0, LGPL 3.0</u> and <u>AGPL 3.0</u> .	s software are

Mac で Postman を使用する場合は、SSL 検証がオフになっていることを確認してください。

1. Postman でスパナ アイコンをクリックして、ドロップダウン メニューから [設定 (Settings)] を選択します。



2. [全般(General)] タブのポップアップ ウィンドウで、SSL 証明書がオフになっていることを 確認します。このオプションが表示されない場合は、そのままウィンドウを閉じてください。

	PC	ost 🗸	https://198	3.18.129.100/ap	i/v1/ticket		
	1.	SETTINGS					
:	2 3 4	General	Themes	Shortcuts	Data	Add-ons	Sync
/		REQUES Trim key	T ys and values	y C	OFF	HEADERS Send no-	
,		SSL certificate verification				OFF	Send Pos

次に、Postman アプリケーションを使用して Python コードを生成します。

ステップ 2: Postman を使用してコードを生成する

ラーニング ラボの <u>REST API および Python</u> [英語] モジュールで、Postman アプリケーションを 使用して API リクエストを行う方法を説明しました。ここでは、Postman を使用して、各種のソフト ウェア プログラミング言語でコードを生成する方法を示します。このステップでは、アプリケーシ ョンの Python 言語オプションを使用します。

ここでは、Postman アプリケーションと Python がワークステーションにインストール済みで、使用 方法も理解していることを前提としています。システムをセットアップしていない場合は、このペー ジの最初の「自分のコンピュータを設定する方法」セクションを参照してください。

コードを生成するには、Postman で関連するフィールドに入力する必要があります。それでは始めましょう。

New Tab	+					No er	nvironm	nent V 💿
get \vee	Enter req	uest URL			Params	Send	~	Save 🗸
Authorization	Headers		Pre-request Scrip	t Tests				Generate Code
Туре		No Auth	\sim					

1. ワークステーションで Postman アプリケーションを開きます。

2. 最初に、APIC-EMから認証トークンを取得して、APIコールを可能にします。それには、 メソッドを POST に設定し、URL を https://198.18.129.100/api/v1/ticket に設定します。

	POST 🗸	https://198.18.129.100/api/v1/ticket	Params	Send	~	Save ~	~
--	--------	--------------------------------------	--------	------	---	--------	---

- 3. 次に、リクエスト ヘッダー情報を定義する必要があります。 ヘッダー タブをクリックして、 1 つのキー値ペアを入力します。
 - [Content-Type] および [application/json]

post ∨	https://198.18.129.100/api/v1/ticket	Params	Send 💙	Save ~
Authorization	Headers (1) Body Pre-request Script Tests			Generate Code
Content-Typ	application/json	≡ ×	Bulk Edit	Presets 🗸
key	value			

- 4. 最後に、認証トークンを取得するために、ユーザ名とパスワードを入力します。[ボディ (Body)] タブをクリックして、[raw] オプションを選択します。下のウィンドウに次の情報を 貼り付けます(この情報は POST のデータとして送信されます)。
- 5. {
- 6. "username" : "admin",
- 7. "password" : "C1sco12345"
- 8. }



- 9. [送信(Send)] ボタンをクリックして、すべてが正常に機能することと、入力した情報が正しいことを確認します。その結果、有効なトークンと 200/OK が得られます。
- これで、Postman を使用してコードを生成する準備ができました。[保存(Save)] ボタンの下に、[コードの生成(Generate Code)] リンクがあります。リンクをクリックします。 [コードスニペットの生成(Generate Code Snippets)] ウィンドウで、[Python] -> [リクエスト(Requests)]の順に選択します。

https://198.18.129.100/-			No Environment	~ © ‡
POST V	GENERATE CODE SNIPPET	S	×	Save ~
Authorization	Python Requests V		Copy to Clipboard	Generate Code
Content-Typ	C (LibCurl)	198.18.129.100/api/v1/ticket"		Presets 🗸
Rey Colling	cURL C# (RestSharp)	<pre>sername\" : \"admin\",\r\n\"password\" : \"Clscol2345\"\r pe': "application/json", rol': "no-cache", ren': "rog89697-78412-sabr-d7ff-dh3a206fa299"</pre>	r\n}"	Time: 1307 mr
Pretty Raw	Go Java 🕨	<pre>ests.request("POST", url, data=payload, headers=headers)</pre>		
1 * [{] 2 * "respons 3 "servi 4 "idle"	NodeJS	text)		
5 "sessi 6 }, 7 "version	OCaml (Cohttp)			
	PHP Python	http.client (Python 3)		
	Ruby (NET::Http) Shell	Requests		
	Swift (NSURL)			

11. requests モジュールによってエントリが Python コードに変換され、API コールが可能 になります。[クリップボードにコピー(Copy to Clipboard)] ボタンをクリックしてコードをコ ピーします。

GENERA	TE CODE SNIPPETS	×
Pythor	n Requests 🗸	Copy to Clipboard
1 2 3 4 5 6 7 8 9 10 11 12 13 14	<pre>import requests url = "https://198.18.129.100/api/v1/ticket" payload = "{\"username\" : \"admin\",\r\n\"password\" : \"Clsco12345\"\r\n headers = { 'content-type': "application/json", 'cache-control': "no-cache", 'postman-token': "be7a2a10-84a6-3884-5878-aa3138c37f72" } response = requests.request("POST", url, data=payload, headers=headers) print@response.text)</pre>	}"

12. クリップボードの内容を任意のテキストエディタに貼り付け、作業ディレクトリに Python ファイルとして保存します。

File Ec	dit Selectio	n Find View Goto	Tools Project Preferences	Help						
4 Þ	import req	uests •								
1	import	requests								-
2								~		
3	url =	Save As								
5	pavloa	$\leftarrow \rightarrow \land \uparrow$	« Users > armartirosyan >	DevNet Express → Iab05	v ⊙	Search lab05		<i>م</i>		
6	header	Organize 🔻 Ne	ew folder					•		
7	.c	A Name	^	Date modified	Туре	Size				
8	°с 'n	-		No items match you	ur search.					
10	۲ {	-								
11										
12	respon	-								
14	print(a								
~ .	[p. 2	-4								
		~								
		File name:	code_from_postman.py					~		
		Save as type:	: All Files (*.*)					~		
						Sava	Cancel			
		A Hide Folders				Jave	Cancer			
	e 14, Column	21							Tab Size: 4	Plain Text

13. 正しく処理できるか試してみてください。コマンドライン ターミナルを開き、作業ディレクト リに移動します。py -3 <FILE-NAME.py> コマンドを実行します。APIC-EM コントローラ からの応答が確認できるはずです。



14. SSL エラーが表示された場合は、リクエスト関数コールに "Verify=False" を追加し、 スクリプトを再度実行します。



以上で、Postmanを使用してコードを生成する方法を学習しました。このラボの次のセクションでは、自身でコードを記述する方法について示します。

ステップ 3: Python スクリプトを記述する

ここでは自身でコードを記述してみましょう。手順に従ってコードを記述すれば、API リクエストが APIC-EM に送信され、認証トークンが取得されます。

Python を使用して API コールを行うには、requests と json の 2 つのモジュールをインポートする必要があります。

#Import necessary modules
import requests
import json

注:JSON モジュールの機能は requests モジュールにすでに含まれているため、技術的には JSON モジュールは必須ではありません。たとえば、JSON データは、data=json.dumps(variable) ではなくjson=variable パラメータを使用して requests.post を呼び出すことで、POST メソッド に渡すことができます。ただし、受講者が json モジュールと loads() および dumps() メソッドの 機能を理解することには意義があります。

モジュールに含まれている必要な機能は、すべて呼び出すことができなければなりません。オプションで、SSL証明書に関連する警告メッセージをオフにすることができます。
 それには次のコードを使用します。

```
#Import necessary modules
import requests
import json
#Disable warnings
requests.packages.urllib3.disable warnings()

    次に、後でコード内で使用する変数を定義します。

#Import necessary modules
import requests
import json
#Disable warnings
requests.packages.urllib3.disable warnings()
# Variables
apic em ip = "https://198.18.129.100/api/v1"
api call ="/ticket"

    次に、ペイロード、ヘッダー、パラメータ情報を定義します。

#Import necessary modules
import requests
import json
```

```
#Disable warnings
requests.packages.urllib3.disable warnings()
# Variables
apic em ip = "https://198.18.129.100/api/v1"
api call ="/ticket"
#Payload contains authentication information
payload = {"username":"admin", "password":"Clsco12345"}
# Header information
headers = {"content-type" : "application/json"}

    この呼び出しの結果を、response 変数に割り当てます。

#Import necessary modules
import requests
import json
#Disable warnings
requests.packages.urllib3.disable warnings()
# Variables
apic em ip = "https://198.18.129.100/api/v1"
api call ="/ticket"
#Payload contains authentication information
payload = {"username":"admin", "password":"Clscol2345"}
# Header information
headers = {"content-type" : "application/json"}
# Combine apic em ip and api call variables into one variable call url
url = apic em ip + api call
response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False)
# Print the respond body
print(response.text)
```

これでコードを使用できるようになりました。コードは、Python ファイルとして作業ディレクトリにコピーして保存できます。

4 b	untitled •					•
1	# Importing necessary modules					
2	import requests					Martin Constant and a second s
3	import json					
4						
5	#Disabl			×		
6	request			^		
7	← → ✓ ↑ 📙 « Users → armartirosyan → DevNet Express → Iab	05 ~ ඊ	Search lab05	م		
8 9	# Varia Organize - New folder			== - ?		
10	url = " A Name Date modified	Туре	Size			
11	api cal code from postman.py 07-Sep-16 02:4	6 Python File	1 KB			
12	access_	·			access	
	token.					
13						
14	# Heade					
15	headers					
16						
17						
18	• •					
19	File <u>n</u> ame: our_code.py			~		
20	# Comb1 Save as type: Python (*.py;*.rpy;*.pyw;*.cpy;*.SConstruct;*.Sconstruct	ct;*.sconstruct;*.SConscript;	*.gyp;*.gypi)	~		
21	url +=a					
22	∧ Hide Folders		<u>S</u> ave	Cancel		
23	respons .			.:.		
24						
25	# Print the respond body					
26	print(response.text)					
	e 4 Column 1				Tah Size: 4	Markdowp
22 23 24 25 26 □ Lin	<pre># Hide Folders # Print the respond body print(response.text) e4,Column 1</pre>		Save		Tab Size: 4	Markdow

どのように機能するかを確認するには、コマンドライン インターフェイスを開き、作業ディレクトリに移動します。次に py -3 our_code.py コマンドを実行します。すべて正しく記述されていれば、端末画面に出力が表示されます。



ご覧のように、出力は読みやすいものではありません。コードを変更して、必要な情報だけが出力されるようにしてみましょう。次のようにします。

- response = requests.post(url, data=json.dumps(payload), headers=headers, verify=False) 行の最後に、.json()を追加します。
- print(response.text) 行をprint("Authenticaton Token: " + response["response"]["serviceTicket"]) に置き換えます。
- 3. これで、serviceTicketキーに保存されている値のみが出力されるようになります。
- コードは最終的に次のようになります。

```
#Import necessary modules
import requests
import json
#Disable warnings
requests.packages.urllib3.disable_warnings()
```



おめでとうございます。REST API コールを行い、APIC-EM から情報を取得する Python コード を記述しました。次のステップでは、ネットワークデバイスを APIC-EM のインベントリリストから 選択し、そのデバイスで実行されている設定を取得した上でファイルに保存するアプリケーショ ンを作成します。

ステップ 4: Python アプリケーション

このステップでは、ネットワークデバイスから設定を取得してファイルに保存するアプリケーションを作成します。デバイスは APIC-EM のインベントリリストから選択します。

APIコールを正しく行うには、どのメソッドを使用するか、またどのような情報をリクエストに含めるかを理解する必要があります。それには、APIC-EMのAPIドキュメント (<u>http://devnetapic.cisco.com/</u>[英語])を参照してください。

ここで作成するアプリケーションでは、[インベントリ(Inventory)] セクションでグループ化されている API コールを主に使用します。ここで内容を確認しておいてください。

APIC-EM-SWAGGER-UI × ← → C				☆ 🛛 🔿	
APIC-EM GA2 Re	lease				-0
Services File Flow Analysis IP Geolocation IP Pool Manager Identity-Manager Inventory Network Discovery	Inventory APIC-EM Service API based on the Swagger™ 1.2 specification				
Network Plug and Play PKI Broker Service Policy Administration	global-credential	Show/Hide	List Operations	Expand Opera	ations
Role Based Access Control Scheduler	host	Show/Hide	List Operations	Expand Opera	ations
Task Topology Visibility	location	Show/Hide	List Operations	Expand Opera	ations
	interface	Show/Hide	List Operations	Expand Opera	ations
	discovery	Show/Hide	List Operations	Expand Opera	ations
	license	Show/Hide	List Operations	Expand Opera	ations
	segment	Show/Hide	List Operations	Expand Opera	ations
	tag	Show/Hide	List Operations	Expand Opera	ations
	network-device	Show/Hide	List Operations	Expand Opera	ations
	network-device/vlan	Show/Hide	List Operations	Expand Opera	ations

アプリケーションを作成する準備ができました。

- 最初に、前のステップのコードをコピーして、関数に変換します。最初の関数では、 APIC-EMのURLアドレスを1つの引数として使用することで、APIC-EMコントローラ から取得した認証トークンが返されます。返されたトークンは、他の関数で使用され、必 要なAPIコールが行われます。
- def get_token(url):
- ٠
- api_call ="/ticket"

```
#Payload contains authentication information
    payload = { "username": "admin", "password": "Clscol2345" }
•
    # Header information
    headers = {"content-type" : "application/json"}
•
•
    # Combine URL, API call variables
•
    url +=api call
•
    response = requests.post(url, data=json.dumps(payload),
•
  headers=headers, verify=False).json()
•
    # Return authentication token from respond body
    return response["response"]["serviceTicket"]
•

    次に、2つの引数、認証トークン、APIC-EMのURLアドレスを使用して、APIC-EMか

  ら(GET)インベントリ情報を取得する新しい関数を作成します。取得した情報は、定義
  した基準によってフィルタリングされます。最初の照合では、一致したデバイスの ID が
  返されます。
• def get device id(token, url):
    #Define API Call
    api call = "/network-device"
•
    #Header information
    headers = {X-AUTH-TOKEN" : token}
•
    # Combine URL, API call variables
•
    url +=api call
•
    response = requests.get(url, headers=headers, verify=False).json()
•
    #Iterate over the response and find first device with access role.
•
    #Return ID number of the first device matching the the `if` statement
•
•
    for item in response['response']:
        if item['role'] == 'ACCESS':
•
•
           return item['id']

    最後に、前の2つの関数によって返された値を使用して、特定のデバイスから設定を取

  得し、ファイルに書き込みます。そのために、3つの引数(認証トークン、APIC-EMの
  URL アドレス、ネットワーク デバイスの ID)を取る新しい関数を作成し、APIC-EM にク
  エリを行ってデバイスの設定を取得する必要があります。
• def get config(token, url, id):
•
    #Define API Call.To get specific device's configuration
    #we will need to add device's ID in the API call
•
    api call = "/network-device/" + id +"/config"
•
•
    #Header information
```

```
headers = {"X-AUTH-TOKEN" : token}
   ٠
   •
        # Combine URL, API call variables
       url +=api call
   •
       response = requests.get(url, headers=headers, verify=False).json()
   ٠
   •
        #Create a file in present working directory
   •
        file = open('access host 1.txt', 'w')
   •
   •
   •
        #Write response body to the file
       file.write(response['response'])
   ٠
       #Close the file when writing is complete
   •
   •
       file.close()
     以上で、すべての関数が作成されたので、それらをまとめます。コードは次のようになり
   •
      ます。
#import modules
import requests
import json
#Disable warnings
requests.packages.urllib3.disable warnings()
# Variables
apic em ip = "https://198.18.129.100/api/v1"
def get token(url):
    #Define API Call
    api call ="/ticket"
    #Payload contains authentication information
   payload = { "username": "admin", "password": "Clscol2345" }
    #Header information
    headers = {"content-type" : "application/json"}
    #Combine URL, API call and parameters variables
   url +=api call
    response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False).json()
    # Return authentication token from respond body
    return response["response"]["serviceTicket"]
def get device id(token,url):
    #Define API Call
    api call = "/network-device"
```

```
#Header information
    headers = {"X-AUTH-TOKEN" : token}
    # Combine URL, API call and parameters variables
    url +=api call
    response = requests.get(url, headers=headers, verify=False).json()
    #Iterate over the response and find first device with access role.
    #Return ID number of the first device matching the criteria
    for item in response['response']:
        if item['role'] == 'ACCESS':
            return item['id']
def get config(token, url, id):
    #Define API Call.To get specific device's configuration
    #we will need to add device's ID in the API call
    api call = "/network-device/" + id +"/config"
    #Header information
    headers = {"X-AUTH-TOKEN" : token}
    # Combine URL, API call variables
    url +=api call
    response = requests.get(url, headers=headers, verify=False).json()
    #Create a file in present working directory
    file = open('access host 1.txt', 'w')
    #Write response body to the file
    file.write(response['response'])
    #Close the file when writing is complete
    file.close()
#Assign obtained authentication token to a variable.Provide APIC-EM's URL
address
auth token = get token(apic em ip)
#Assign obtained ID to a variable.Provide authentication token and APIC-EM's
URL address
device_id = get_device_id(auth_token, apic_em_ip)
#Call get config() function to obtain and write device's configuration to a
file.
#Provide authentication token, APIC-EM's URL address and device's ID
get config(auth token, apic em ip, device id)
```

• 上記のコードをコピーして、Python ファイルとして保存します。

< >)	untitled								
63	"content	-type" : '	'application/json"	•					
64	"X-AUTH	TOKEN"	-olcon					-	
65	}	Save As						×	
66	←	→ ^ ↑	« Users > armartirosyan > [DevNet Express → Iab05	ٽ ~	Search lab05		<i>P</i>	
67	# Combine URL, AP	anniza z Na	w folder				8== -	0	
68	url +=api_call	ganize + ive	^					•	And Contrast of Address
69		Name		Date modified	Туре	Size			
70	response = reques	🕴 📑 code	_from_postman.py	07-Sep-16 02:46	Python File	1 KB			
71		🖌 🛛 🛃	ode.py	08-Sep-16 14:26	Python File	3 KB			
/2	#Create a file in	•							
73	file = open(acce								
74	##Ttopata aven th	E.							
75	fon itom in norm	41.00							
70	file write(it	1							
78	1116.00106(10	_ ~							
79	#Close the file w	File name:	our ann ny					~	
80	file.close()	Courses to an				•			
81		Save as type:	Python (".py;".rpy;".pyw;".cpy;".	SConstruct;".Sconstruct;".sci	onstruct; Sconscript;	.gyp;".gypi)			
82	#Assign obtained auth	11.1.5.11				Save	Cancel		
83	auth token = get toke	Hide Folders				3070	cuncer		
84									
85									
86	<pre>device_id = get_device_i</pre>	d(auth_tol	ken, apic_em)						
87									
88						ile. <mark>-</mark>			
89									
90	<pre>get_config(auth_token, a</pre>	pic_em, de	evice_id)						
🛛 🗌 Line	75, Column 64							Spaces: 4	Python

 ターミナルウィンドウを開き、Pythonコードが保存されているディレクトリに移動します。
 py -3 our_app.py または python our_app.pyコマンドを実行して、アプリケーションを 起動します。すべてが正しく実行されると、端末画面に新しいプロンプトが表示されます。これは、コードの実行中にエラーが発生しなかったことを示します。

C:\Windows\System32\cmd.exe	_	\times
F:\>cd "Users\armartirosyan\DevNet Express\lab05"		^
F:\Users\armartirosyan\DevNet Express\lab05>dir Volume in drive F is Personal Drive Volume Serial Number is 52E1-8A76		
Directory of F:\Users\armartirosyan\DevNet Express\lab05		
08-Sep-16 14:36 <dir> 08-Sep-16 14:36 <dir> 07-Sep-16 02:46 407 code_from_postman.py 08-Sep-16 14:30 2,643 our_app.py 08-Sep-16 14:26 2,643 our_code.py 3 File(s) 5,693 bytes 2 Dir(s) 112,129,646,592 bytes free</dir></dir>		
F:\Users\armartirosyan\DevNet Express\lab05>py -3 our_app.py		
F:\Users\armartirosyan\DevNet Express\lab05>		
		\checkmark

次にターミナルウィンドウでフォルダの内容を再度確認します。access_host_1.txtという名前の新しいファイルができているはずです。このファイルはアプリケーションによって作成されたものです。

G C:\Windows\System32\cmd.exe	_	\times
F:\>cd "Users\armartirosyan\DevNet Express\lab05"		
F:\Users\armartirosyan\DevNet Express\lab05>dir Volume in drive F is Personal Drive Volume Serial Number is 52E1-8A76		
Directory of F:\Users\armartirosyan\DevNet Express\lab05		
08-Sep-16 14:36 <dir> 08-Sep-16 14:36 <dir> 07-Sep-16 02:46 407 code_from_postman.py 08-Sep-16 14:30 2,643 our_app.py 08-Sep-16 14:26 2,643 our_code.py 3 File(s) 5,693 bytes 2 Dir(s) 112,129,646,592 bytes free</dir></dir>		
F:\Users\armartirosyan\DevNet Express\lab05>py -3 our_app.py		
F:\Users\armartirosyan\DevNet Express\lab05>dir Volume in drive F is Personal Drive Volume Serial Number is 52E1-8A76		
Directory of F:\Users\armartirosyan\DevNet Express\lab05		
08-Sep-16 14:38 <dir> 08-Sep-16 14:38 <dir> 08-Sep-16 14:38 <dir> 07-Sep-16 14:38 4,611 access_host_1.txt 07-Sep-16 02:46 407 code_from_postman.py 08-Sep-16 14:30 2,643 our_app.py 08-Sep-16 14:26 2,643 our_code.py 4 File(s) 10,304 bytes 2 Dir(s) 112,129,613,824 bytes free</dir></dir></dir>		
F:\Users\armartirosyan\DevNet Express\lab05>		~

 ファイルの内容を確認するには、任意のテキストエディタを使用してテキストファイルを 開くか、ファイルの内容をターミナルウィンドウに出力します。ここではテキストエディタ を使用してファイルの内容を確認します。

📙 🕑 🛄 🖛 lab05	F-\Users\armartirosyan\DevNet Express\lab05\access_host_1.bt - Sublime Text (UNREGISTERED)	- 🗆 X
File Home Share View	File Edit Selection Find View Goto Tools Project Preferences Help	
← → ~ ↑ 🔤 « Personal Drive (F:) → Users → a	mart ◀ ▶ access host 1.bt ×	
File Home Share View Image: Share View Image: Share View Image: Share Image: Share View Image: Share View Image: Share Image	<pre>File Edd Selection Find View Geto Tools Project Preferences Help arcess.host.lbd 2 Building configuration 3 4 Current configuration : 8228 bytes 5 4 Current configuration change at 18:29:47 UTC Tue Jul 5 2016 7 ! NVRAM config last updated at 18:29:53 UTC Tue Jul 5 2016 8 9 version 15.2 10 service config 11 no service pad 12 service timestamps debug datetime msec 13 service timestamps log datetime msec 14 no service pad 15 ! 16 hostname AHEC-2960C1 17 ! 18 boot-start-marker 19 boot-end-marker 20 ! 21 ! 22 username xxxxxx 24 no aaa new-model 25 system mtu routing 1500 26 ! 27 ! 28 ! </pre>	
	29 ! 30 !	
• ~	31 1	
4 items 1 item selected 4.50 KB	Eine 1, Column 1	Tab Size: 4 Plain Text

• アプリケーションが正しく機能していることがわかります。

アプリケーションの改善

出力ファイルには、access_host_1.txtという名前を付けるよりも、設定を取得したデバイスの ホスト名を付けるほうがわかりやすくなります。さらにファイル名に日付と時刻を加えることで、設 定がいつ保存されたが明らかになります。この方法で、ネットワーク全体の設定が含まれたバッ クアップファイルを数秒で作成し、それを変更管理メカニズムとして使用して、特定の時点で特 定のデバイスにどのような変更があったかを確認できます。

この方法を実施するには、コードを少し変更する必要があります。

- 1. datetimeとreの2つの新しいモジュールをインポートします。re モジュールはデバイ スのホスト名の取得に役立ち、datetime モジュールでは現在の日付と時刻が得られ、 ファイル名にタイムスタンプを付加することができます。
- #Import necessary modules
 import requests
 - 4. import json
 - 5. import datetime
 - 6. import re
- 7. get_config() 関数を変更して、デバイスのホスト名を取得し、ファイル名を目的の形式 に変更します。デバイスのホスト名を取得する方法は複数ありますが、正規表現を作成 する方法が最も効率的です。変更された関数は次のようになります。

```
8. def get config(token, url, id):
9.
10.
    #Define API Call.To get specific device's configuration
11. #we will need to add device's ID in the API call
     api call = "/network-device/" + id +"/config"
12.
13.
14.
     #Header information
15. headers = {"X-AUTH-TOKEN" : token}
16.
17. # Combine URL, API call variables
18. url +=api call
19.
20.
     response = requests.get(url, headers=headers, verify=False).json()
21.
22.
     #Find the hostname in the response body and save it to a hostname
  variable
23. hostname = re.findall('hostname\s(.+?)\s', response['response'])[0]
24.
25. #Create a date time variable which will hold current time
26. date time = datetime.datetime.now()
27.
     #Create a variable which will hold the hostname combined with the
28.
   date and time
29. #The format will be hostname year month day hour.minute.second
30. file name = hostname + ' ' + str(date time.year) + ' ' +
  str(date time.month) + ' ' + str(date time.day) + ' ' +
   str(date time.hour) + '.'+ str(date time.minute) + '.'+
   str(date time.second)
31.
32. file = open(file name+'.txt', 'w')
```

33.
34. #Write response body to the file
35. file.write(response['response'])
36.
37. #Close the file when writing is complete
38. file.close()
39. ここで Python スクリプトに必要な変更を加えて、コードを再度実行します。

おめでとうございます。デバイスの設定をバックアップしてファイルに保存するアプリケーションを記述しました。