cisco Live!

#CiscoLive

The bridge to possible

Automating Cisco FTD Deployments

Rafael Leiva-Ochoa

BRKCRT-2301

#CiscoLive

Agenda

- Introduction
- Overview on REST API
- Overview on Ansible
- FTD to VMware install using Ansible
- Registering FTD using Python REST API
- Managing FTD using Ansible Modules
- Conclusion

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

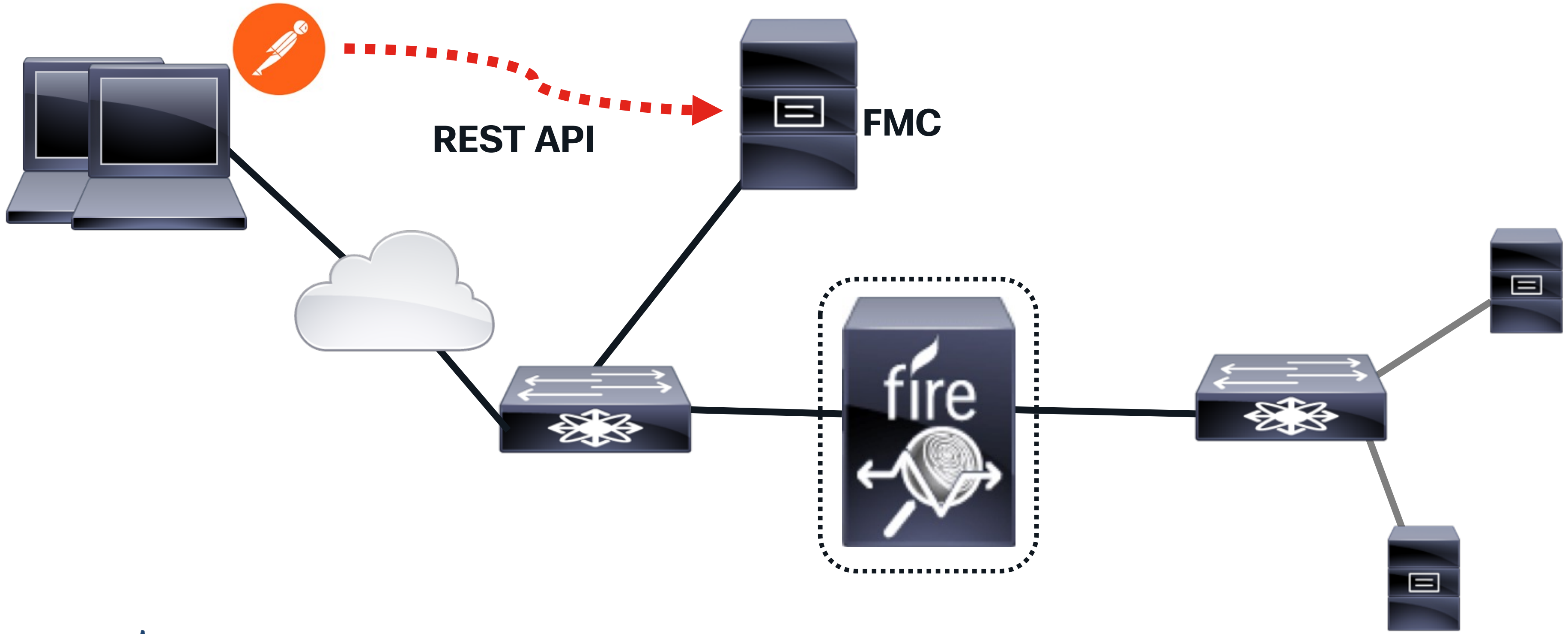
Introduction

Introduction

- Rafael Leiva-Ochoa
- @Cisco since Oct 2000
- Works in the CX Training Group (Part of Learning@Cisco)
- Delivers courses on Security to Global TAC Centers
- CCIE 19322 Security since 2007

Overview on REST API

REST API Overview



REST API Requirements

- FTD/FMC version 6.2.3 and later
- REST API Client (Postman)
- REST API versions 1, 2, 3, and 4 (effects what you can do)
- Works with Firepower Device Manager (FDM), and FMC

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

REST API versions

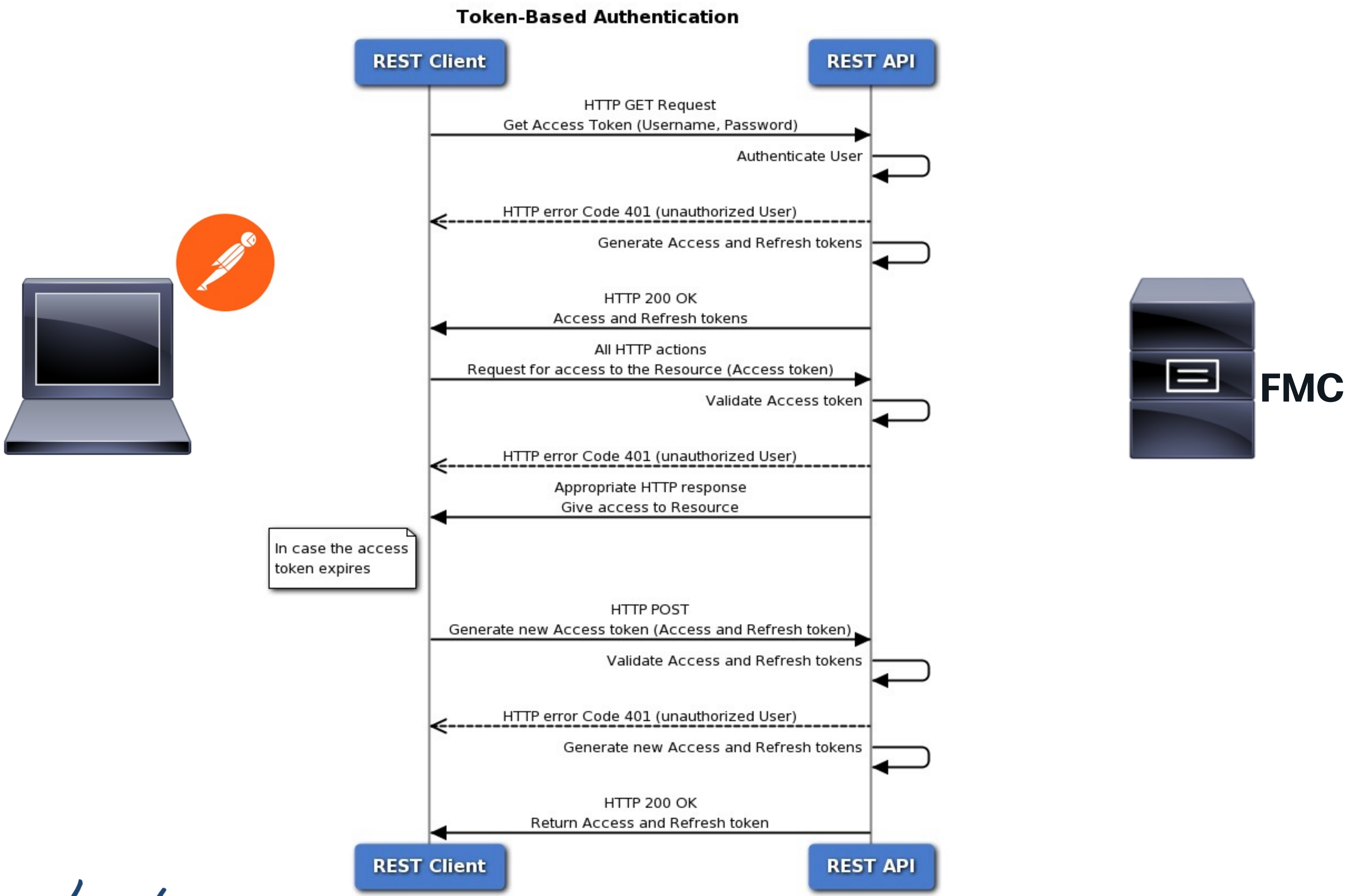
REST API Version	FTD Version	Changes
v1	6.2.3	This is the initial release of the FTD REST API.
v2	6.3.0	Version 2 adds resources for all new features available in FTD 6.3.0. You can now configure external authorization for API access using a RADIUS server. For this version, you must change v1 in the API URLs to v2.

REST API versions (Cont.)

v3	6.4.0	Version 3 adds resources for all new features available in FTD 6.4.0. New in this release is the GET /api/versions (ApiVersions) method, which you can use to determine the API versions you can use on the device; do not include the version number in the GET call. For this version, you can use v3 or latest in the API URLs. The use of latest as a version alias is new in this release.
v4	6.5.0	Version 4 adds resources for all new features available in FTD 6.5.0. Significant changes include the resources and methods for the following, but this is not an exhaustive list: •ConfigurationImportExport, for exporting and importing the device configuration (/action/configexport, /jobs/configexportstatus, /action/configimport, /jobs/configimportstatus). •FileAndMalwarePolicies, for the creation of custom file policies, including filepolicies, filetypes, filetypecategories, ampcloudconfig, ampservers, and ampcloudconnections. •Security Intelligence DNS policies, adding the following SecurityIntelligence resources: domainnamefeeds, securityintelligencednspolicies. •LDAP attribute maps for use with remote access VPN. We added or modified the following FTD For this version, you can use v4 or latest in the API URLs.

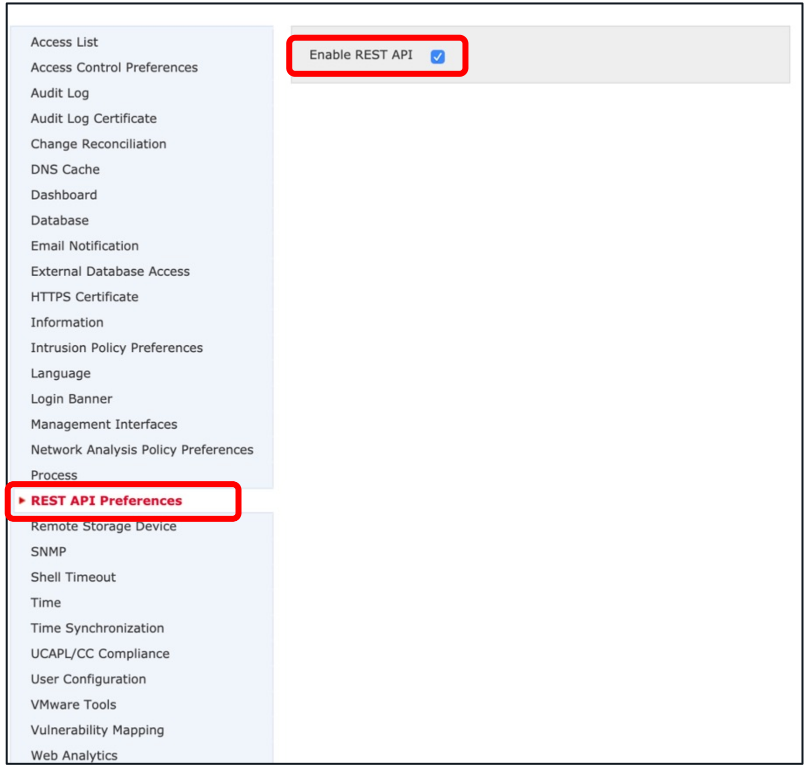
REST API Explorer Access Token

- Tokens are used to access the HTTP service for a limited time period without the need for the username and password with every request
- In order eliminate the need for authenticating with your username and password with each request, you replace user credentials with a uniquely generated access token
- Tokens are only good for 30 minutes and can refresh up to three times.


Token-Based Authentication
REST Client
REST API
HTTP GET Request
Get Access Token (Username, Password)
Authenticate User
HTTP error Code 401 (unauthorized User)
Generate Access and Refresh tokens
HTTP 200 OK
Access and Refresh tokens
All HTTP actions
Request for access to the Resource (Access token)
Validate Access token
HTTP error Code 401 (unauthorized User)
Appropriate HTTP response
Give access to Resource
In case the access token expires
HTTP POST
Generate new Access token (Access and Refresh token)
Validate Access and Refresh tokens
HTTP error Code 401 (unauthorized User)
Generate new Access and Refresh tokens
HTTP 200 OK
Return Access and Refresh token
REST Client
REST API
FMC

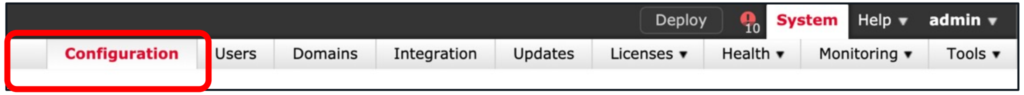
© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Enable REST API on FMC



Access List
Access Control Preferences
Audit Log
Audit Log Certificate
Change Reconciliation
DNS Cache
Dashboard
Database
Email Notification
External Database Access
HTTPS Certificate
Information
Intrusion Policy Preferences
Language
Login Banner
Management Interfaces
Network Analysis Policy Preferences
Process
REST API Preferences
Remote Storage Device
SNMP
Shell Timeout
Time
Time Synchronization
UCAPL/CC Compliance
User Configuration
VMware Tools
Vulnerability Mapping
Web Analytics

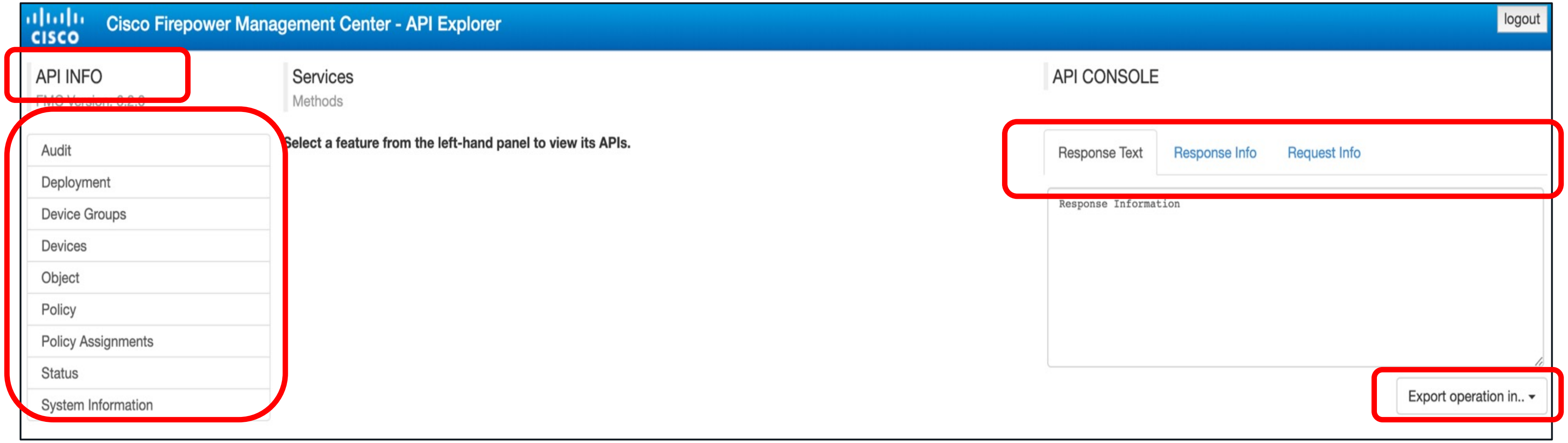
Enable REST API ✓



- The REST API is enabled by default
- Base URL:

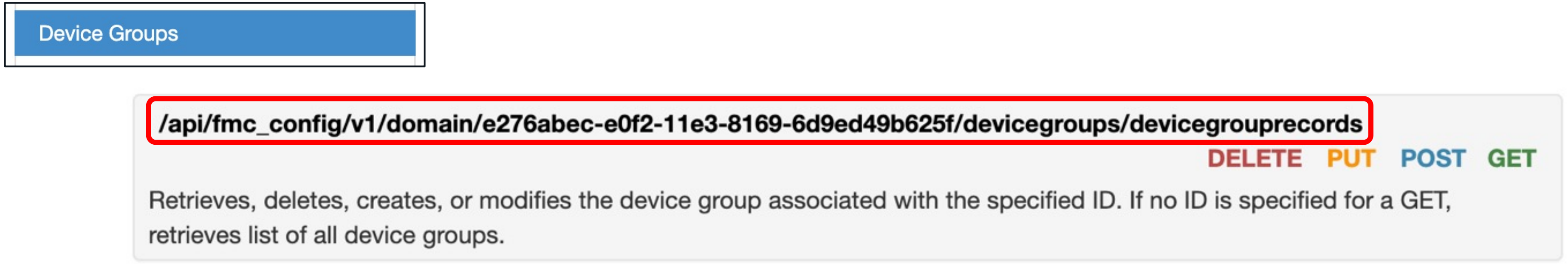
https://<management_center_IP_or_name>:<https_port>/api/api-explorer

Access API API-Explorer



- Any FMC user can login but is still limited to the functions that the user can perform.

Access API API-Explorer (Cont.)

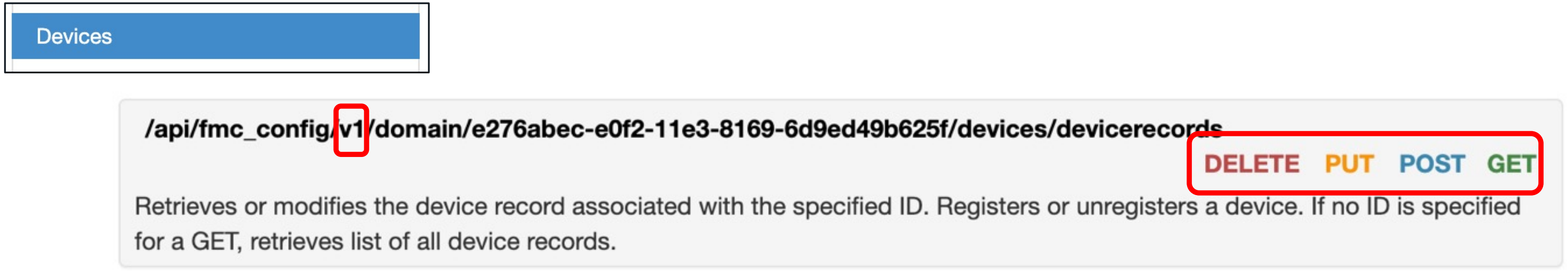


Device Groups

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devicegroups/devicegrouprecords

DELETE PUT POST GET

Retrieves, deletes, creates, or modifies the device group associated with the specified ID. If no ID is specified for a GET, retrieves list of all device groups.



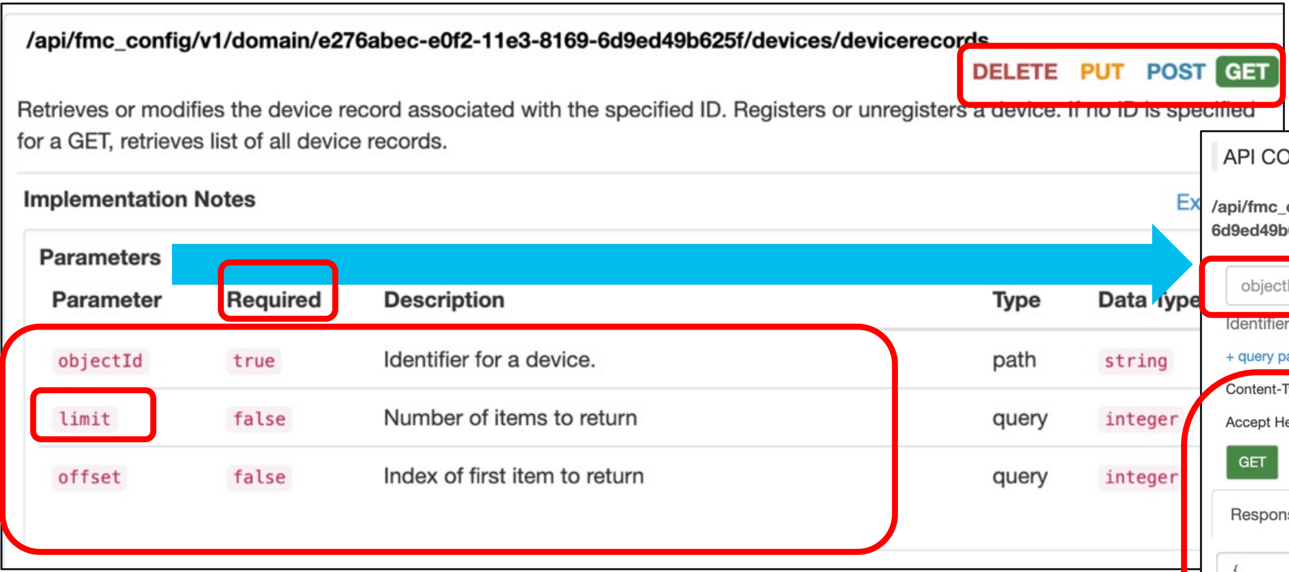
Devices

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

DELETE PUT POST GET

Retrieves or modifies the device record associated with the specified ID. Registers or unregisters a device. If no ID is specified for a GET, retrieves list of all device records.

Access API API-Explorer (Cont.)



/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

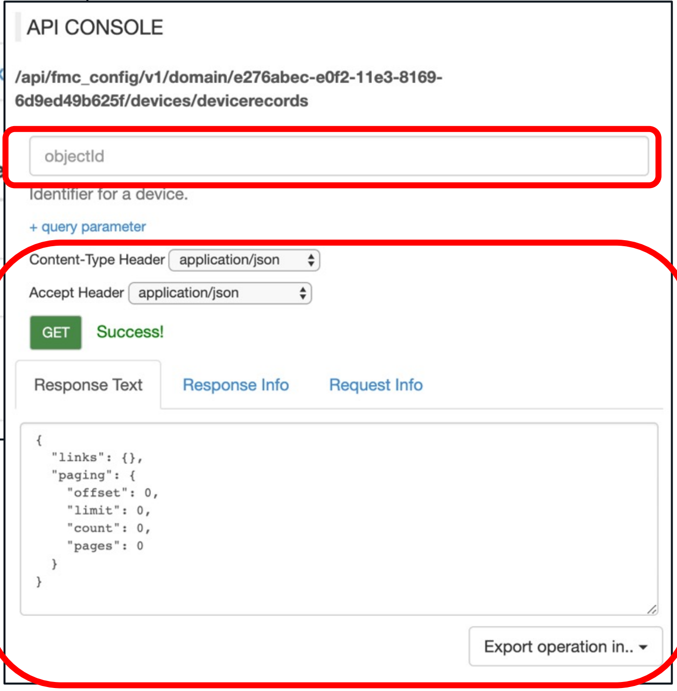
DELETE PUT POST GET

Retrieves or modifies the device record associated with the specified ID. Registers or unregisters a device. If no ID is specified for a GET, retrieves list of all device records.

Implementation Notes

Parameters

Parameter	Required	Description	Type	Data Type
`objectId`	`true`	Identifier for a device.	path	`string`
`limit`	`false`	Number of items to return	query	`integer`
`offset`	`false`	Index of first item to return	query	`integer`



API CONSOLE

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

objectId

Identifier for a device.

+ query parameter

Content-Type Header: application/json

Accept Header: application/json

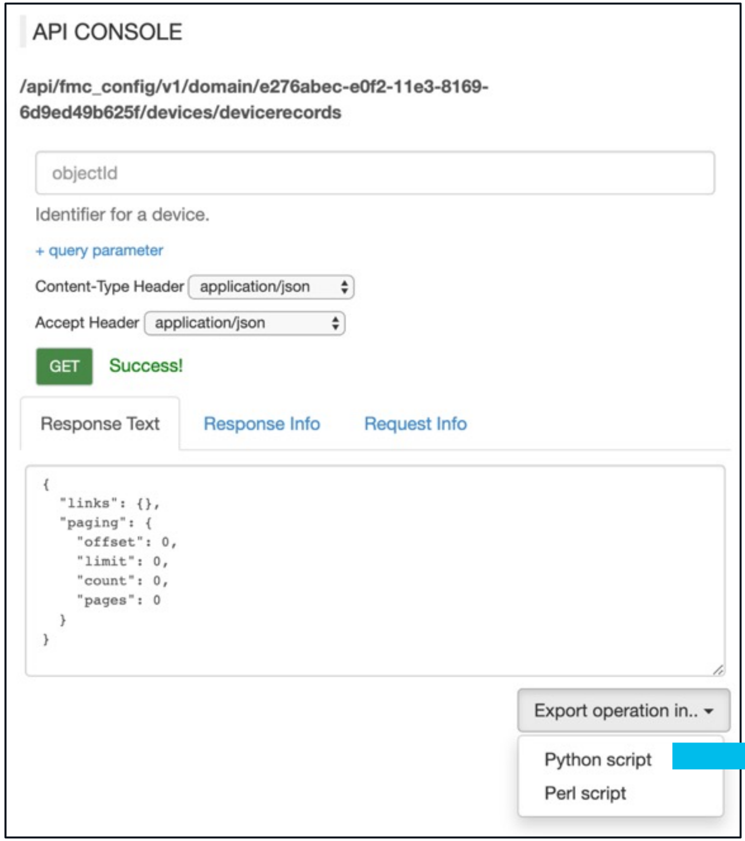
GET Success!

Response Text | Response Info | Request Info

```
{
  "links": {},
  "paging": {
    "offset": 0,
    "limit": 0,
    "count": 0,
    "pages": 0
  }
}
```

Export operation in..

Access API API-Explorer (Cont.)


API CONSOLE

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

objectId

Identifier for a device.

+ query parameter

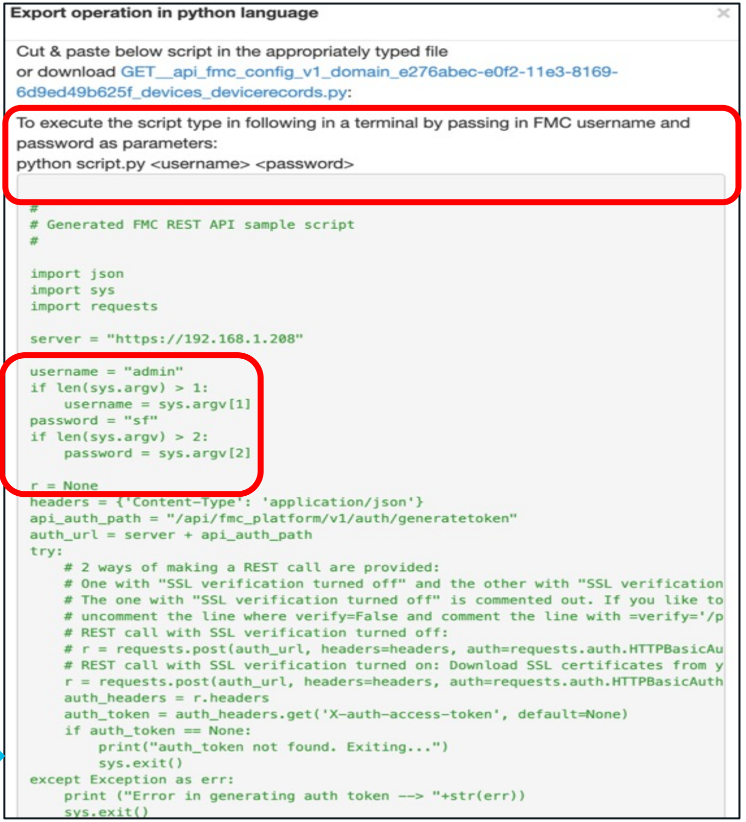
Content-Type Header: application/json

Accept Header: application/json

GET Success!

Response Text | Response Info | Request Info

```
{
  "links": {},
  "paging": {
    "offset": 0,
    "limit": 0,
    "count": 0,
    "pages": 0
  }
}
```

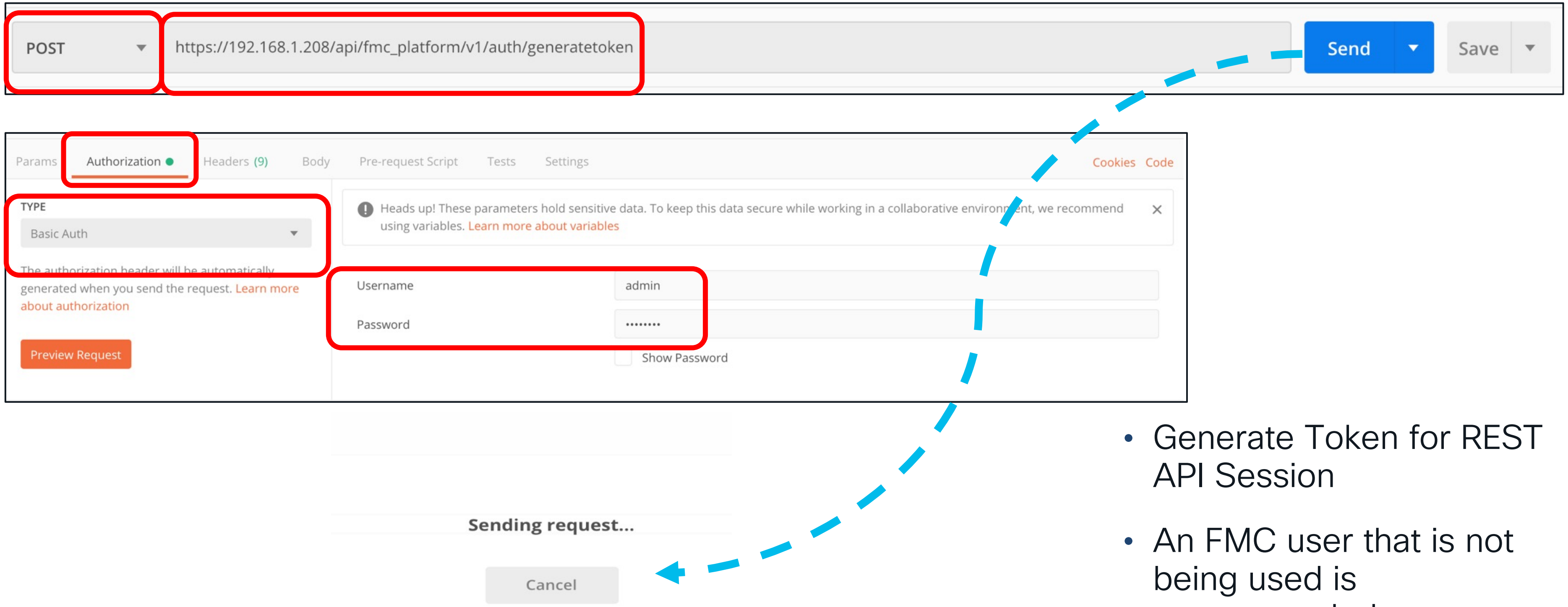

Export operation in python language

Cut & paste below script in the appropriately typed file or download GET__api_fmc_config_v1_domain_e276abec-e0f2-11e3-8169-6d9ed49b625f_devices_devicerecords.py:

To execute the script type in following in a terminal by passing in FMC username and password as parameters:
python script.py <username> <password>

```
#
# Generated FMC REST API sample script
#

import json
import sys
import requests

server = "https://192.168.1.208"

username = "admin"
if len(sys.argv) > 1:
    username = sys.argv[1]
password = "sf"
if len(sys.argv) > 2:
    password = sys.argv[2]

r = None
headers = {'Content-Type': 'application/json'}
api_auth_path = "/api/fmc_platform/v1/auth/generatetoken"
auth_url = server + api_auth_path
try:
    # 2 ways of making a REST call are provided:
    # One with "SSL verification turned off" and the other with "SSL verification
    # The one with "SSL verification turned off" is commented out. If you like to
    # uncomment the line where verify=False and comment the line with =verify='/p
    # REST call with SSL verification turned off:
    # r = requests.post(auth_url, headers=headers, auth=requests.auth.HTTPBasicAu
    # REST call with SSL verification turned on: Download SSL certificates from y
    r = requests.post(auth_url, headers=headers, auth=requests.auth.HTTPBasicAuth
    auth_headers = r.headers
    auth_token = auth_headers.get('X-auth-access-token', default=None)
    if auth_token == None:
        print("auth_token not found. Exiting...")
        sys.exit()
except Exception as err:
    print ("Error in generating auth token --> "+str(err))
    sys.exit()
```

Postman REST API Client



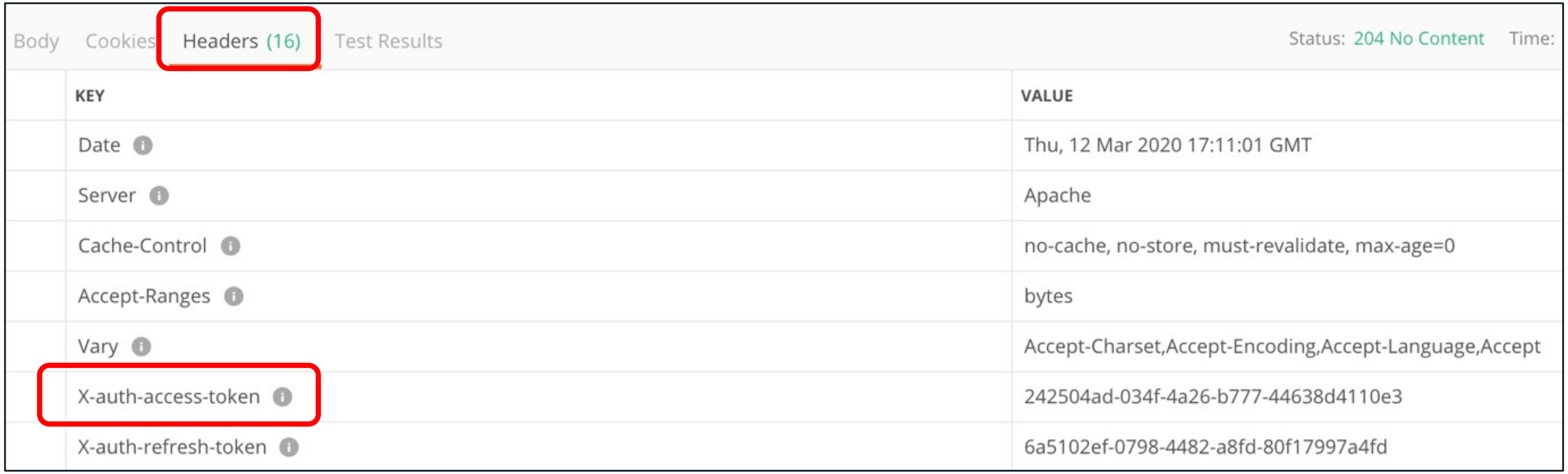
The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: admin
Password: ••••••••

- Generate Token for REST API Session
- An FMC user that is not being used is recommended.

Postman REST API Client (Cont.)

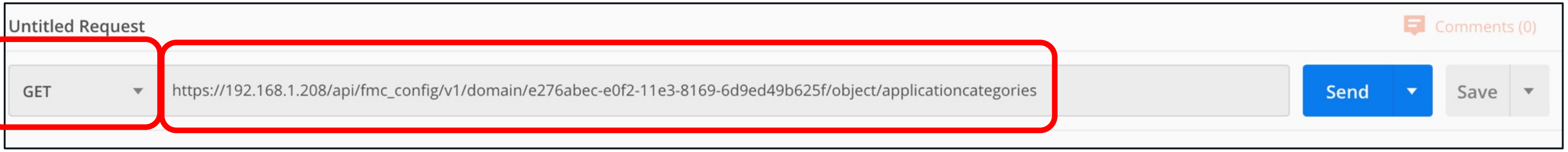


Body | Cookies | Headers (16) | Test Results — Status: 204 No Content — Time:

KEY	VALUE
Date	Thu, 12 Mar 2020 17:11:01 GMT
Server	Apache
Cache-Control	no-cache, no-store, must-revalidate, max-age=0
Accept-Ranges	bytes
Vary	Accept-Charset,Accept-Encoding,Accept-Language,Accept
X-auth-access-token	242504ad-034f-4a26-b777-44638d4110e3
X-auth-refresh-token	6a5102ef-0798-4482-a8fd-80f17997a4fd

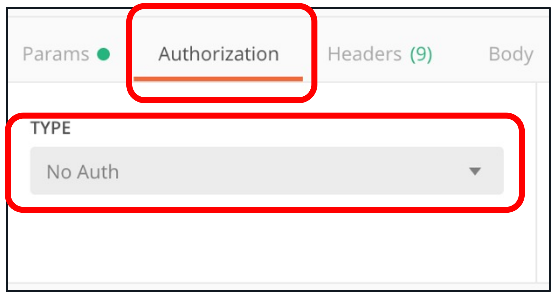
- Headers will contain the X-auth-access-token, this will be good for 30mins. For this example, this applies only to the FTD platform. Other platforms might have the API key last longer.

Postman REST API Client (Cont.)

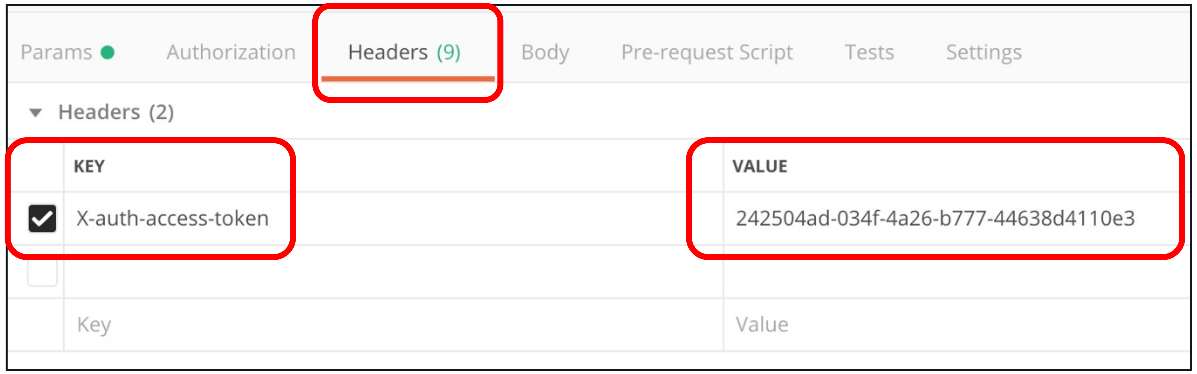

Untitled Request

GET

https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories



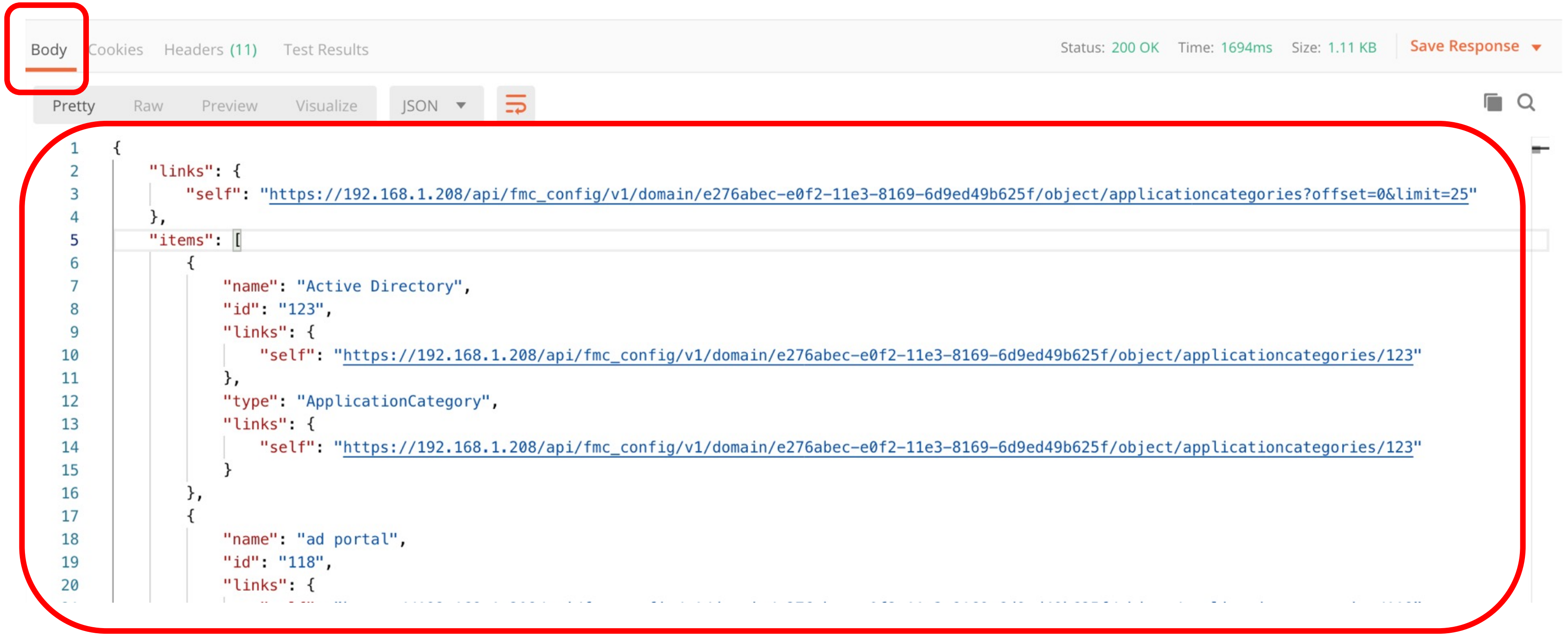
TYPE: No Auth



Headers (2)

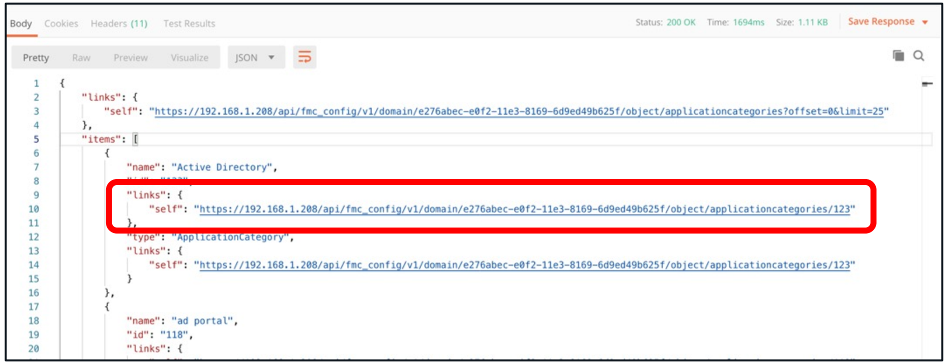
	KEY	VALUE
✓	X-auth-access-token	242504ad-034f-4a26-b777-44638d4110e3
	Key	Value

Postman REST API Client (Cont.)

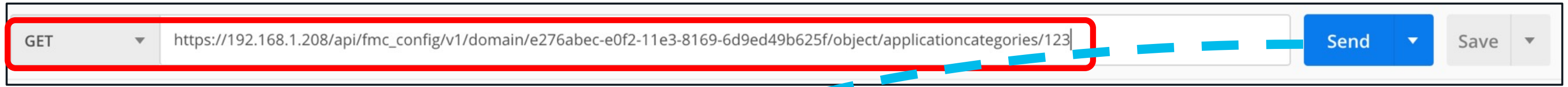


```
{
    "links": {
        "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories?offset=0&limit=25"
    },
    "items": [
        {
            "name": "Active Directory",
            "id": "123",
            "links": {
                "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
            },
            "type": "ApplicationCategory",
            "links": {
                "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
            }
        },
        {
            "name": "ad portal",
            "id": "118",
            "links": {
```

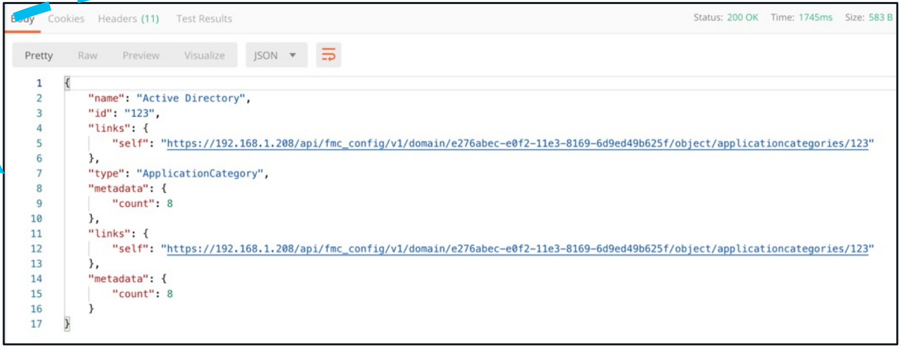
Postman REST API Client (Cont.)



```
{
    "links": {
        "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories?offset=0&limit=25"
    },
    "items": [
        {
            "name": "Active Directory",
            "id": "123",
            "links": {
                "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
            },
            "type": "ApplicationCategory",
            "links": {
                "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
            }
        },
        {
            "name": "ad portal",
            "id": "118",
            "links": {
```



```
GET https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123
```



```
{
    "name": "Active Directory",
    "id": "123",
    "links": {
        "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
    },
    "type": "ApplicationCategory",
    "metadata": {
        "count": 8
    },
    "links": {
        "self": "https://192.168.1.208/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/applicationcategories/123"
    },
    "metadata": {
        "count": 8
    }
}
```

Overview on Ansible

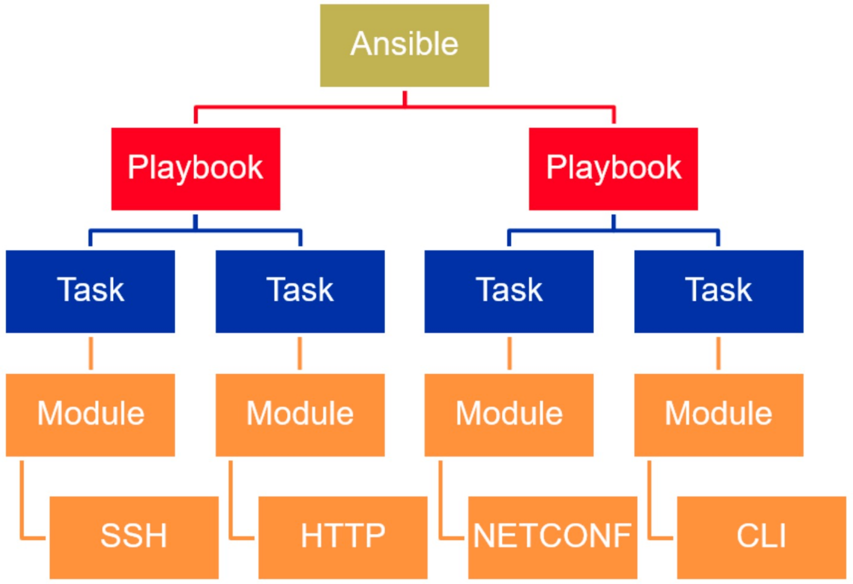
Ansible Overview

- Ansible is free automation tool created by Michael DeHaan. Acquired by RedHat.
- Able to use SSH, HTTP, NETCONF, and CLI(SSH) for transport. SSH is the default.
- Modules are used to support different features on devices.

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Ansible Architecture

- Playbooks are used to execute Tasks with supported Moules.
- For example, if an administrator wanted to configure a FTD interface configuration, there needs to be a FTD module in Ansible to configure the tasks in a playbook to deploy that configuration.



FTD to VMware install using Ansible

Challenge with Deploying Multiple FTD on VMware

- FTD devices can take an average of 30 to 45 min's to install and bootstrap manually.
- Python pexpect and Powershell scripts can be very messy, and complicated to maintain.

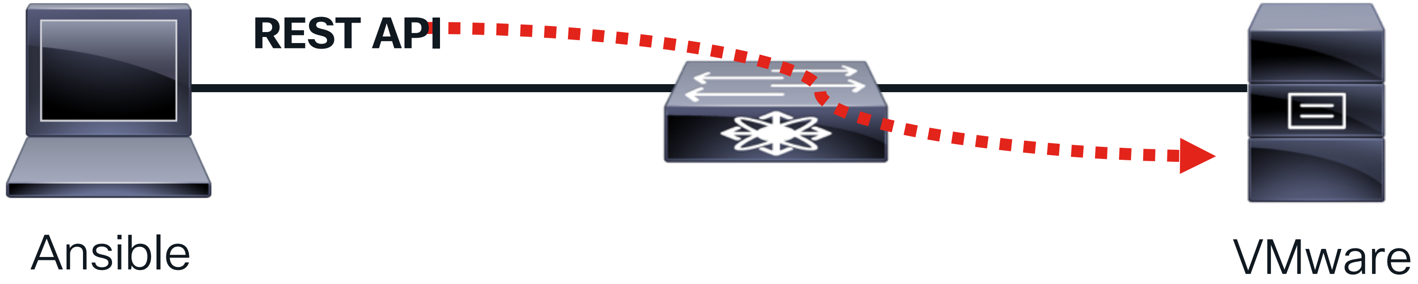
© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

VMware FTD Installation using Ansible CiscoDevNet FTD Modules

- Requirements:
  - Linux Server running Ansible 2.8 or higher
  - Python 3
  - Install vSphere Automation Python SDK from Git:
    - https://github.com/CiscoDevNet/FTDAnsible/blob/master/samples/deployment/vmware/README.md

VMware FTD Installation using Ansible CiscoDevNet FTD Modules (cont.)

- Modify /etc/ansible/ansible.conf file to enable vmware plugin:
  - [inventory] enable_plugins = vmware_vm_inventory
- Use the Vmware Deployment options from Git
  - https://github.com/CiscoDevNet/FTDAnsible/tree/master/samples/deployment/vmware
  - From here you will modify the vars.yml file to fit the needs of your VMware environment



Ansible YAML files Walk-Through

- ansible.cfg
  - Store all the ansible environment options and settings
  - The [inventory] option enable plugins that ansible uses to support default, and custom features
  - The default location where the plugs are stored are: /usr/share/ansible/plugins/inventory
  - This can be changed using the:

    inventory_plugins = /usr/share/ansible/plugins/inventory
  - Example:
  - [inventory]
  - enable_plugins = vmware_vm_inventory

Ansible YAML files Walk-Through (cont.)

- vars.yml
  - The vars.yml file is created to store all the variable that will be used during the playbook. This simplifies the configuration and avoids reptation.

```
vcenter_hostname: "{{ lookup('env','VMWARE_SERVER') }}"
vcenter_username: "{{ lookup('env','VMWARE_USERNAME') }}"
vcenter_password: "{{ lookup('env','VMWARE_PASSWORD') }}"
```

  - Another useful option is to set environment variables on your Linux Server:

```
export VMWARE_SERVER=...vCenter hostname...
export VMWARE_USERNAME=...vCenter username...
export VMWARE_PASSWORD=...vCenter password...
```

Note: the export command is not persisting after reboot.

Ansible YAML files Walk-Through (cont.)

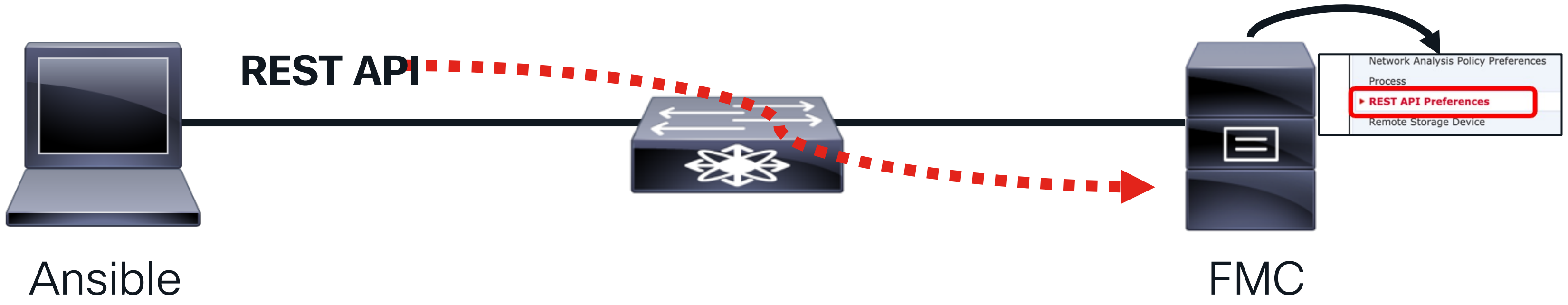
- deploy.yml
  - The deploy.yml file contains the playbook to deploy the FTD devices on VMware. As stated before, the vars.yml is critical to configure, since the deploy.yml depends on this file.
- demo_cloud.vmware.yaml
  - The demo_cloud.vmware.yaml file is used to set the inventory host cache for the deployment.

- ansible-playbook -i demo_cloud.vmware.yaml deploy_and_destroy.yaml
  - The ansible-playbook command is used to execute the playbook. Notice that we are using the deploy_and_destroy.yaml. This will create the FTD instance on Vmware and then destroy it. If you need to ONLY create it, then use the deploy.yml playbook only.

Registering FTD using Python REST API

FTD Registration using Python REST API

- Before the FTD can be registered to the FMC using Python, the REST API needs to be enabled on the FMC.
- Python 3 recommended
- GET UUID for FMC



Python Module Requirements

- **pip install** <module name>
  - json
  - sys
  - request

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Python REST API Script walk-through

```
import json
import sys
import requests

server = "https://<FMC FQDN>"

username = "admin"
if len(sys.argv) > 1:
    username = sys.argv[1]
password = "sf"
if len(sys.argv) > 2:
    password = sys.argv[2]
```

Import modules

FMC Username

FMC Password

Python REST API Script walk-through (cont.)

```
r = None
headers = {'Content-Type': 'application/json'}
api_auth_path = "/api/fmc_platform/v1/auth/generatetoken"
auth_url = server + api_auth_path

try:
    # 2 ways of making a REST call are provided:
    # One with "SSL verification turned off" and the other with "SSL verification turned on".
    # The one with "SSL verification turned off" is commented out. If you like to use that then
    # uncomment the line where verify=False and comment the line with =verify='/path/to/ssl_certificate'
    #
    # REST call with SSL verification turned off:
    r = requests.post(auth_url, headers=headers, auth=requests.auth.HTTPBasicAuth(username,password), verify=False)

    # REST call with SSL verification turned on: Download SSL certificates from your FMC first and provide its path for verification.
    # r = requests.post(auth_url, headers=headers, auth=requests.auth.HTTPBasicAuth(username,password),
verify='/path/to/ssl_certificate')

    auth_headers = r.headers
    auth_token = auth_headers.get('X-auth-access-token', default=None)
    if auth_token == None:
        print("auth_token not found. Exiting...")
        sys.exit()
except Exception as err:
    print ("Error in generating auth token --> "+str(err))
    sys.exit()
```

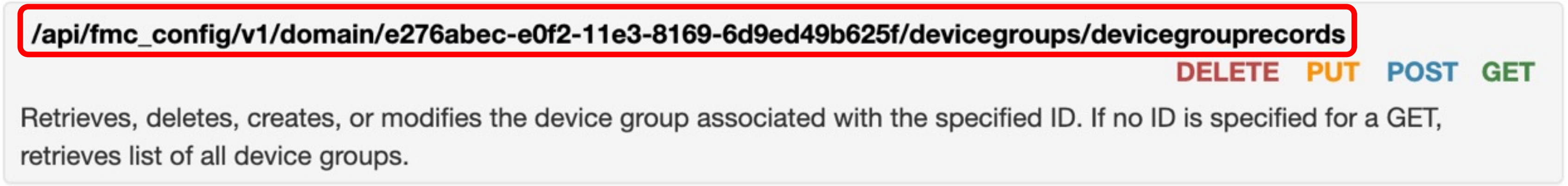
Header Encoding

API Version

Verifies that token was retrieved

Python REST API Script walk-through (cont.)

Device Groups

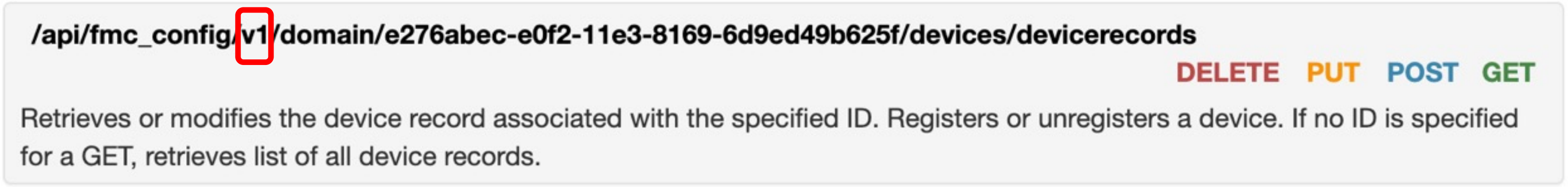


/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devicegroups/devicegrouprecords

DELETE PUT POST GET

Retrieves, deletes, creates, or modifies the device group associated with the specified ID. If no ID is specified for a GET, retrieves list of all device groups.

Devices



/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

DELETE PUT POST GET

Retrieves or modifies the device record associated with the specified ID. Registers or unregisters a device. If no ID is specified for a GET, retrieves list of all device records.

Python REST API Script walk-through (cont.)

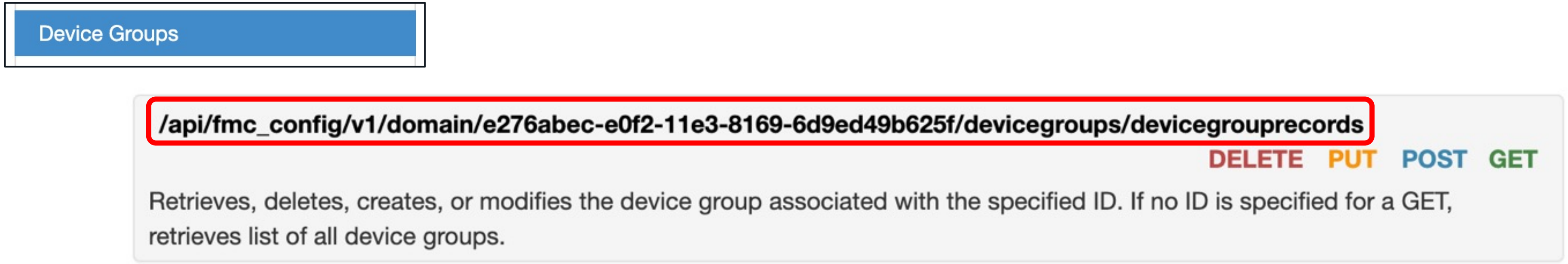
```
headers['X-auth-access-token']=auth_token

api_path = "/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords" # param
url = server + api_path if (url[-1] == '/'): url = url[:-1]

post_data = {
  "name": "xyz",
  "hostName": "abc.xyz",
  "natID": "cisco123",
  "regKey": "regkey",
  "type": "Device",
  "license_caps": [
    "BASE",
    "MALWARE",
    "URLFilter",
    "THREAT"
  ],
  "accessPolicy": {
    "id": "accessPolicyUUID",
    "type": "AccessPolicy"
  }
}
```

Unique FMC UUID

FTD Device to register

Unique ACP UUID

Python REST API Script walk-through (cont.)

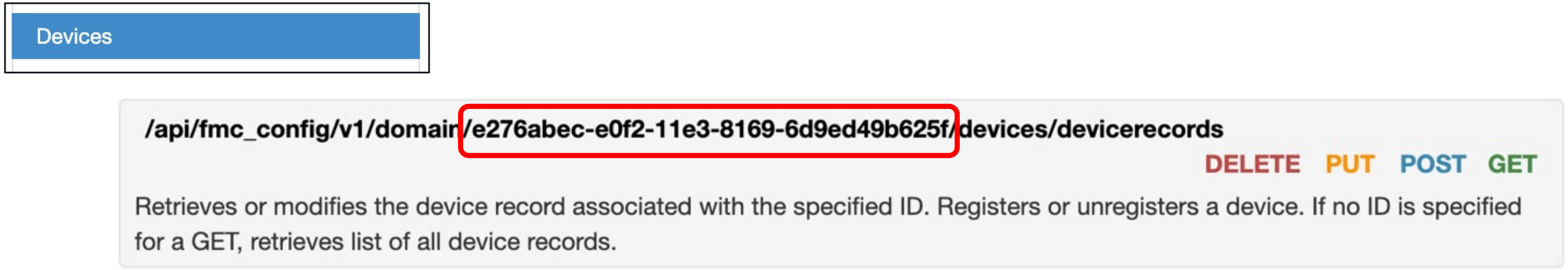


Device Groups

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devicegroups/devicegrouprecords

DELETE PUT POST GET

Retrieves, deletes, creates, or modifies the device group associated with the specified ID. If no ID is specified for a GET, retrieves list of all device groups.



Devices

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/devices/devicerecords

DELETE PUT POST GET

Retrieves or modifies the device record associated with the specified ID. Registers or unregisters a device. If no ID is specified for a GET, retrieves list of all device records.

Python REST API Script walk-through (cont.)

```
try:
    # REST call with SSL verification turned off:
    r = requests.post(url, data=json.dumps(post_data), headers=headers, verify=False)
    # REST call with SSL verification turned on:
    #r = requests.post(url, data=json.dumps(post_data), headers=headers, verify='/path/to/ssl_certificate')
    status_code = r.status_code
    resp = r.text
    print("Status code is: "+str(status_code))
    if status_code == 201 or status_code == 202:
        print ("Post was successful...")
        json_resp = json.loads(resp)
        print(json.dumps(json_resp,sort_keys=True,indent=4, separators=(',', ': ')))
    else :
        r.raise_for_status()
        print ("Error occurred in POST --> "+resp)
except requests.exceptions.HTTPError as err:
    print ("Error in connection --> "+str(err))
finally:
    if r: r.close()
```

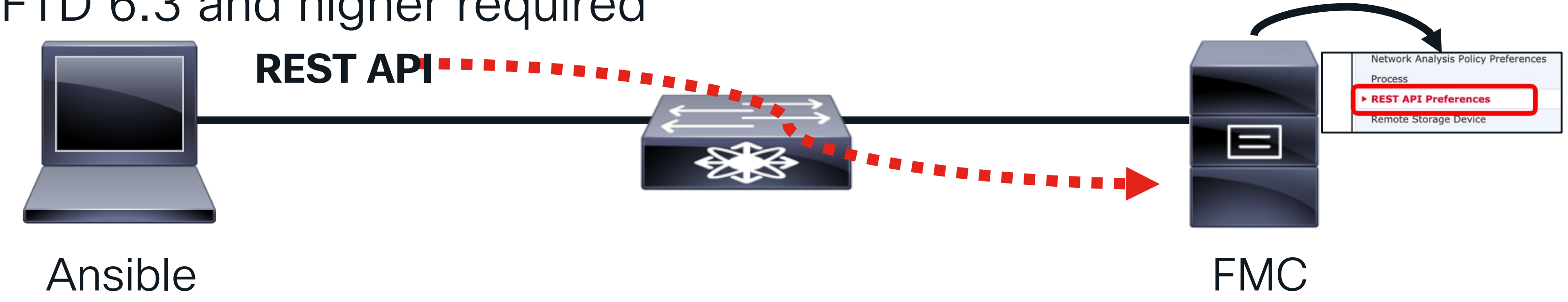
Sends the registration request to the FMC

Error Correction

Managing FTD using Ansible Modules

Cisco DEVNET FTD Modules

- Follow the instructions on: https://developer.cisco.com/docs/ftd-ansible-v6-3/#!installation-guide to install docker image to enable DEVNET FTD Modules. As of Ansible 2.7, the module is included.
- The DEVNET FTD Modules support add, delete, get, upsert, and edit options
- FTD 6.3 and higher required



Working with FTD Ansible modules

Parameter	Choices/Defaults	Comments
data dictionary		Key-value pairs that should be sent as body parameters in a REST API call
filters dictionary		Key-value dict that represents equality filters. Every key is a property name and value is its desired value. If multiple filters are present, they are combined with logical operator AND.
operation string / required		The name of the operation to execute. Commonly, the operation starts with 'add', 'edit', 'get', 'upsert' or 'delete' verbs, but can have an arbitrary name too.
path_params dictionary		Key-value pairs that should be sent as path parameters in a REST API call.
query_params dictionary		Key-value pairs that should be sent as query parameters in a REST API call.
register_as string		Specifies Ansible fact name that is used to register received response from the FTD device.

Working with FTD Ansible modules (cont.)

```
- name: Execute 'addDeployment' operation
  ftd_configuration:
    operation: "addDeployment"
    data:
        statusMessage: "{{ status_message }}"
        cliErrorMessage: "{{ cli_error_message }}"
        state: "{{ state }}"
        queuedTime: "{{ queued_time }}"
        startTime: "{{ start_time }}"
        endTime: "{{ end_time }}"
        statusMessages: "{{ status_messages }}"
        name: "{{ name }}"
        modifiedObjects: "{{ modified_objects }}"
```

```
- name: Execute 'addNetworkObject' operation
  ftd_configuration:
    operation: "addNetworkObject"
    data:
        name: "{{ name }}"
        description: "{{ description }}"
        subType: "{{ sub_type }}"
        value: "{{ value }}"
        isSystemDefined: "{{ is_system_defined
}}"
        dnsResolution: "{{ dns_resolution }}"
        type: "{{ type }}"
```

Working with FTD Ansible modules (Cont.)

```
- name: Create a network object
  ftd_configuration:
    operation: "addNetworkObject"
    data:
      name: ”Inside Subnet"
      description: ”Host Inside Network"
      subType: "HOST"
      value: ”172.16.35.0"
      dnsResolution: "IPV4_AND_IPV6"
      type: "networkobject"
      isSystemDefined: false
    register_as: "hostNetwork"
```

```
- name: Delete the network object
  ftd_configuration:
    operation: "deleteNetworkObject"
    path_params:
      objId: "{{ hostNetwork['id'] }}"
```

Conclusion

Important Facts about Ansible

- Ansible playbooks take time to buildout and test
- Administrators will need to lookup values using GET via FMC API-Explorer
- Some values stored on the FMC might not be names, but numbers. Making it much more difficult to setup on ansible playbook.
  - Example: objId: "{{ hostNetwork['id'] }}"

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

More information on FTD Automation

- FMC/FTD automation videos, and learning labs: https://developer.cisco.com/firepower/management-center/
- DEVNET Associate Certification:

https://www.cisco.com/c/en/us/training-events/training-certifications/certifications/devnet/cisco-certified-devnet-associate.html

The bridge to possible

Thank you

cisco Live!

#CiscoLive