Cisco
Customer Experience

Accelerator

Advanced Feature Deep Dive:

DNA Center Encrypted Credential

Shirin Khan

Customer Success Specialist

September 2023

In today's discussion:

- DNAC Credential
- The Cryptography Open Source Project-Fernet
- APIs
- Cisco DNA Center APIs
- Demo
- Run Script

Introduction

Business Challenge: Cisco Secure Development Lifecycle compatible DNAC credential encryption for automation.

- Using Fernet to generate key or salt and save Fernet key in a folder different than python script folder
- Admin user can generate the key 0.00031693698838353157 seconds
- Give read access to the “Key” folder for the user who runs the DNAC script
- Generate DNAC Encrypted credential from Plain text, timestamp and the Key and save it in “Credential” Folder
  0.1451286189840175 2 seconds
- Save the Encrypted Credential File in a different folder with read access to only to DNAC Script run user
- User who can run the DNAC script should have read access to “Key” folder and “Credential” Folder
- User who runs DNAC script with read access can update encrypted username and password for DNAC and run the script.
  0.06 seconds to decrypt.
- CSDL compliant – Cisco Secure Development Lifecycle Compliant

❑ **Result: Encrypted credentials login only takes 0.06 seconds additional to run.**

APIs

- What is an API
- Basic Authentication vs OAuth2.0
- REST API Request Components
- REST API Request and Response Exchange
- REST API Response Components

What is an API?

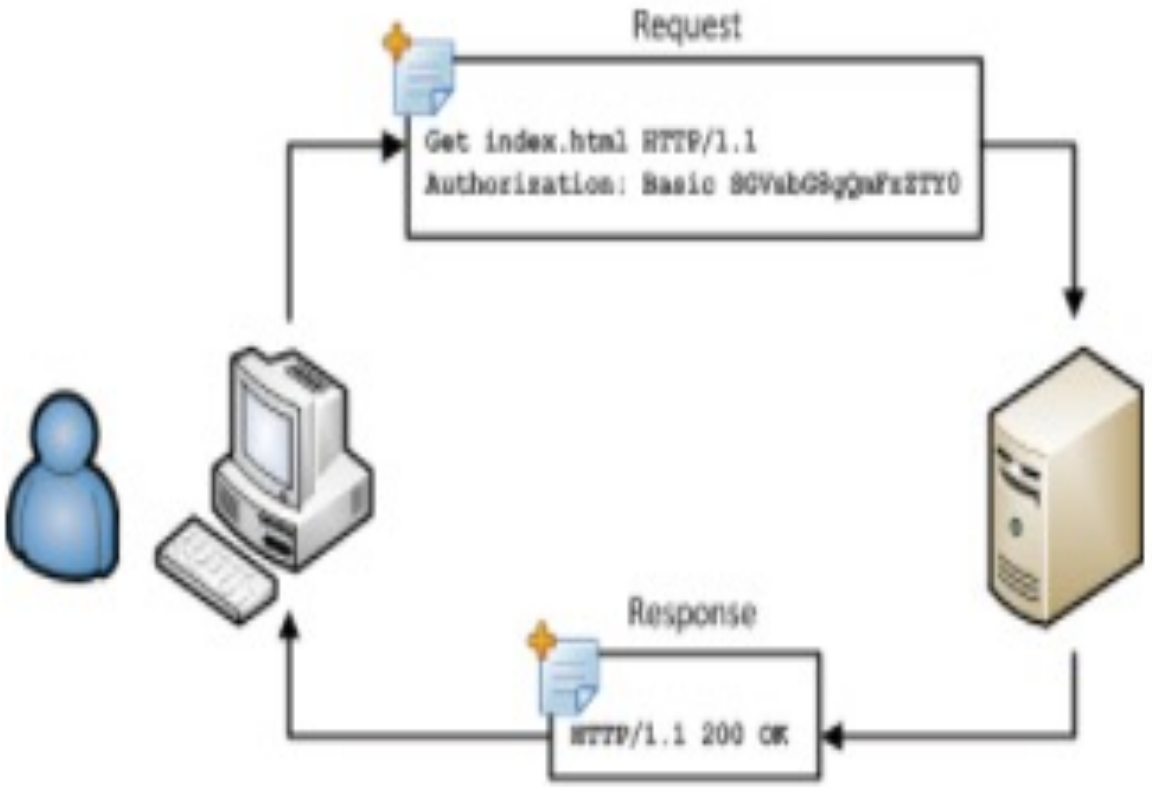
- **API** – **A**pplication **P**rogramming **I**nterface
  - Set of subroutine definitions, protocols, and tools for building application software
  - Specifies how software components should interact with each other
  - Many types of APIs exist
- **RESTful APIs (**REpresentational **"State"** Transfer**)**
- Stateless: The server does not store any state about the client session on the server-side

Use HTTP requests to **C**reate/**R**ead/**U**pdate/**D**elete (**CRUD**) operations:

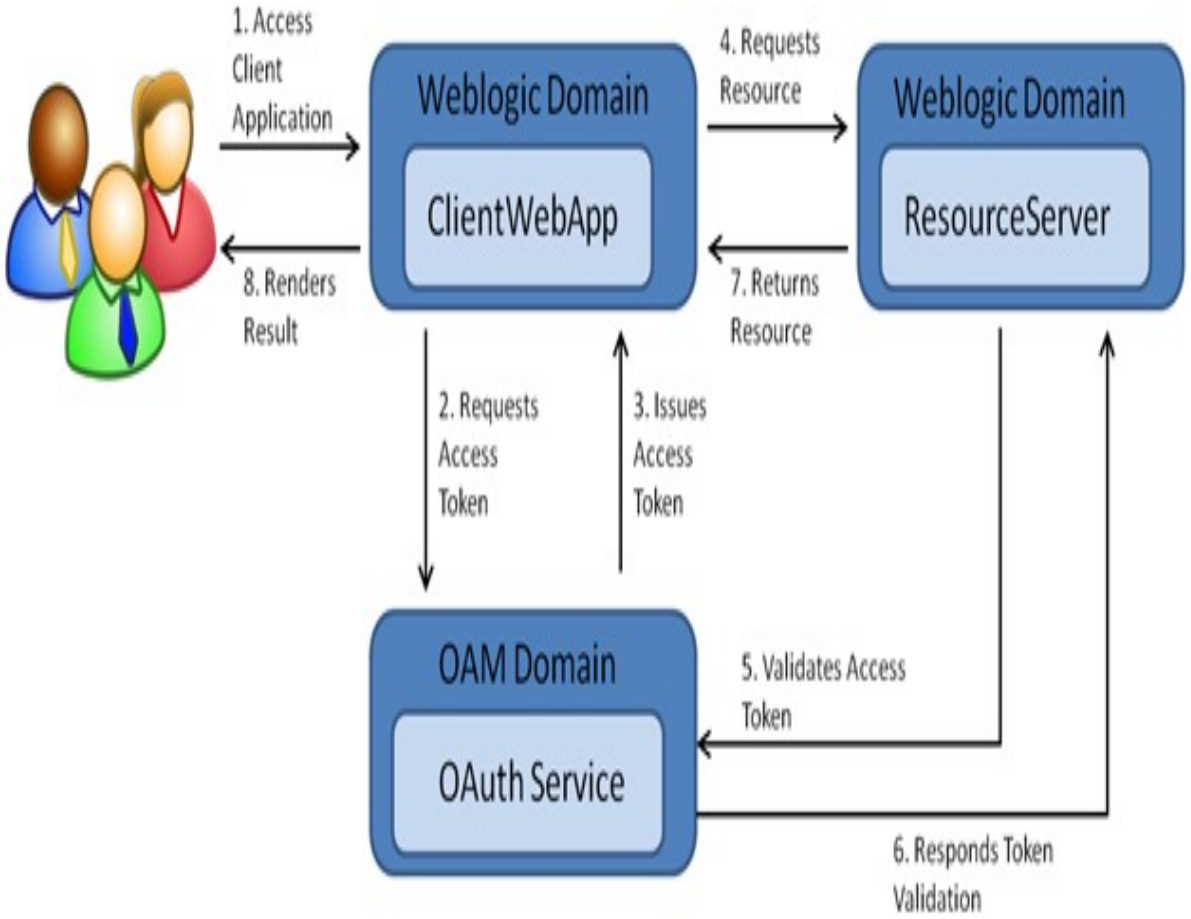
- **C**reates a new resource
- **R**etrieves/**R**ead a resource
- **U**pdates an existing resource
- **D**eletes a resource

Function–A Task & Return of the results
Task – action 1 action 2… action n

Basic Authentication vs OAuth2.0



Basic Authentication



OAUTH2.0

[Basic Authentication vs OAuth2.0]

REST API Request Components

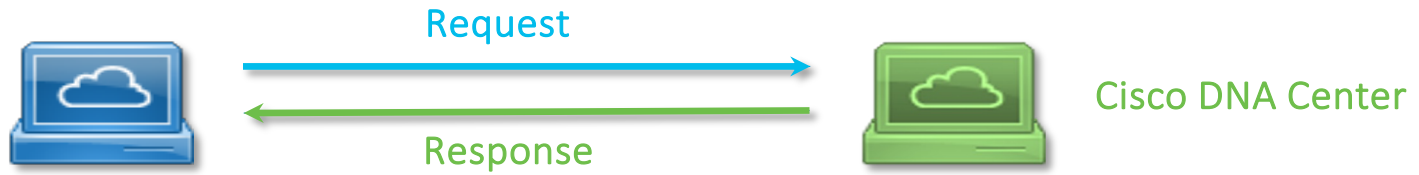
- **URL**: Application server and the API resource
- **Authentication**: HTTP basic, custom, OAuth, none
- **Headers**: HTTP headers example: Content-Type: application/json
- **Request Body**: JSON or XML - the data needed to complete request
- **Method (CRUD)**:
  - **POST** - **C**reates a new resource
  - **GET** - **R**etrieves/**R**ead a resource
  - **PUT** - **U**pdates an existing resource
  - **DELETE** - **D**eletes a resource

```
'{"name":"John", "age":30, "car":null}'
```

```
<note>
<to>Jignesh Shah</to>
<from>Shirin Khan</from>
<heading>Reminder Demo</heading>
<body>Custom Accelerator!!</body>
</note>
```

REST API request and response exchange

```
header = url = DNAC_URL + f ' /dna/intent/api/v1/network-device?limit=LIMIT&offset=OFFSET'
{'content-type': 'application/json', 'x-auth-token': dnac_jwt_token}
client_response = requests.get(url, headers=header, verify=False)
```



The response value will be assigned to variable client_response

```
"response": [
        { …
            "type": "Cisco 4800 Series Unified Access Points",
            "lastUpdateTime": 1694578455813,
            "macAddress": "a0:b4:39:7d:ca:a0",
            "softwareVersion": "17.6.1.13",
            "deviceSupportLevel": "Supported",
            "serialNumber": "FJC2422M4DL",
            "apManagerInterfaceIp": "192.168.200.206",
            "collectionStatus": "Unassociated",
            "family": "Unified AP",
            "hostname": "AP7C21.0EDA.DB2C",
             "role": "ACCESS", ….. }]
```

```
client_json = client_response.json()
client_info = client_json['response'][0]
```

Parse JSON

REST API Response Components

- HTTP status codes
  - 2xx Success – 200 OK, 201 created
  - 4xx Client Error – 400 Bad request, 401 Unauthorized, 404 Not
  - 5xx Server Error – 500 Internal server error
  - https://www.restapitutorial.com/httpstatuscodes.html
- Headers
  - Content-type – JSON or XML, cache control, date, encoding
- Response body
  - Payload with requested data formatted in JSON, XML, or other
- JSON = JavaScript Object Notation: Easy for BOTH humans and machines to read and write
- Example: [{" firstName" : " Michael" , " lastName" : " Jordan" },
  {" firstName" : " Lebron" , " lastName" : " James" }].

https://www.programiz.com/python-programming/dictionary, json.org

List = ['10.x.x.x', '192.x.x.x']

https://www.programiz.com/python-programming/list

1xx status codes: Informational requests

100 Continue
101 Switching Protocols
102 Processing

2xx status codes: Successful requests

200 OK
201 Created
202 Accepted
203 Non-Authorative information
204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status
208 Already reported

3xx status codes: Redirects

300 Multiple Choices
301Moved Permanently
302 Found
303 See Other
304 Not modified
305 Use Proxy
307 Temporary Redirect
308 Permanent Redirect

4xx status codes: Client errors

400 Bad Request
401 Unathorized
402 Payment Required
409 Conflict
410 Gone
412 Precondition Failed
416 Requested Range Not Satisfiable
417 Expectation Failed
422 Unprocessable Entity
423 Locked
424 Failed Dependency
426 Upgrade Required
429 Too Many Requests
431 Request Header Fields Too Large
451 Unavailable for Legal Reasons

5xx status codes: Server errors

500 Internal Server error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates
507 Insufficient Storage
508 Loop Detected
510 Not Extended
511 Network Authentication Required

Cisco DNA Center APIs

- Rest API - Authorization Token
- Enable and Activate Rest API Bundle on DNAC

REST API Request

Retrieve the Cisco DNA Center Token from client

```
def get_dnac_jwt_token(dnac_auth, DNAC_URL):

    url = DNAC_URL + '/api/system/v1/auth/token'

    header = {'content-type': 'application/json'}

    response = requests.post(url, auth=dnac_auth, headers=header, verify=False)

    if response.status_code == 200:

        dnac_jwt_token = response.json()['Token']
    return dnac_jwt_token
```

User defined function

URL

Application Server

API Resource

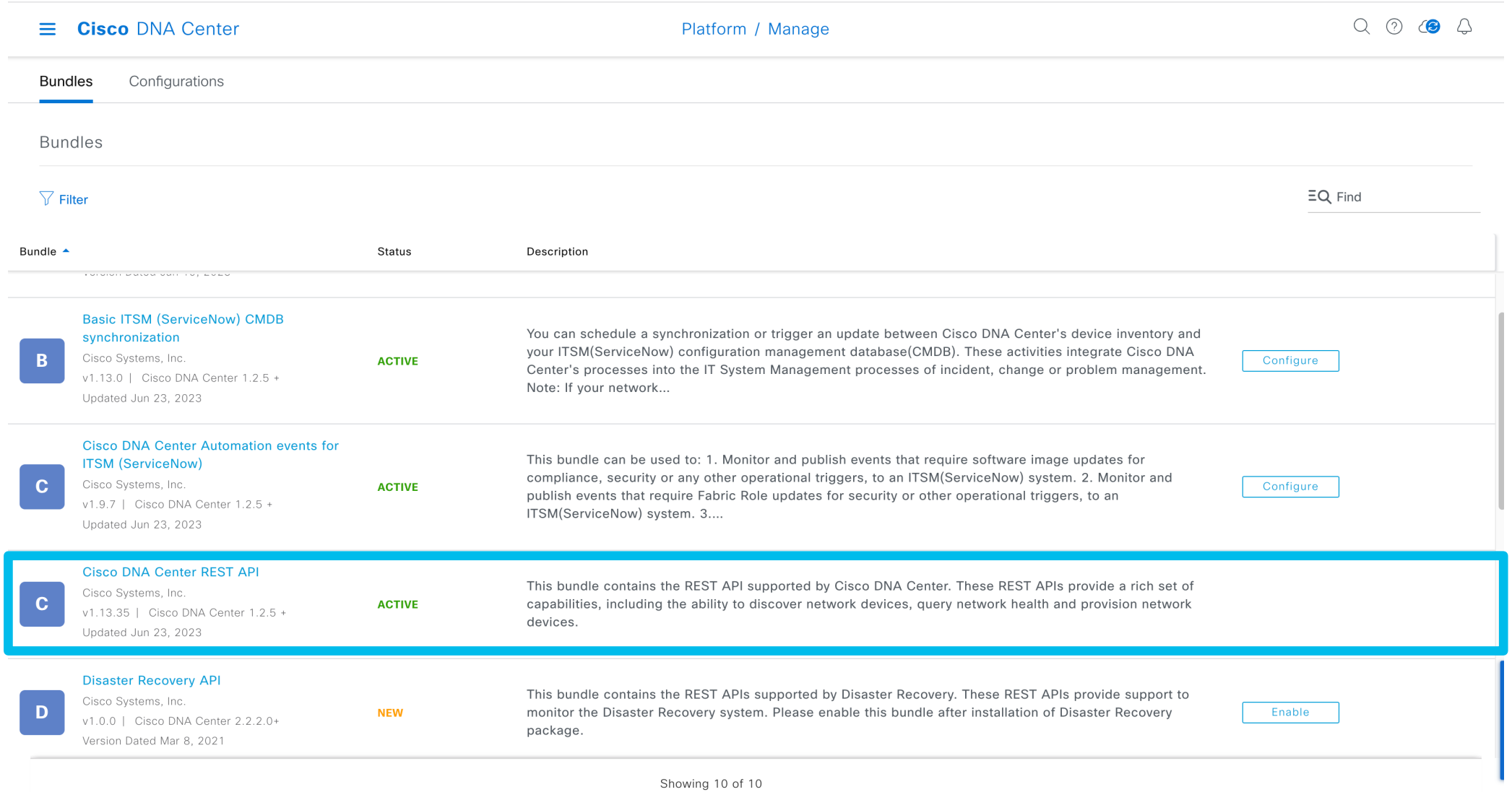
Headers

Authentication

Method

Parsing JSON

Enable and Activate Rest API Bundle



Credential Cryptography

- DNAC Credential
- Cryptography - Fernet
- Methods
- Encryption Process
- References

DNAC Credential

- Step 1: Generate Key
- Step 2: Generate Encrypted Credentials user and password with Key
  - Prime Sanitize
- Step 3: Generate decrypted credentials in plain text -
  - Prime De-sanitize

The Cryptography Open Source Project - Fernet

- Cryptography deals with the encryption of plaintext into ciphertext and decryption of ciphertext into plaintext.
- Python supports a cryptography package that helps us encrypt and decrypt data.
- The fernet module of the cryptography package has inbuilt functions for the generation of the key, encryption of plaintext into ciphertext, and decryption of ciphertext into plaintext using the encrypt and decrypt methods respectively.
- The fernet module guarantees that data encrypted using it cannot be further manipulated or read without the key.

Fernet - Encrypt/Decrypt

- **Fernet is a recipe that provides symmetric encryption and authentication to data**. It is a part of the *cryptography* library for Python, which is developed by the Python Cryptographic Authority (PYCA).
- Recipes are small python scripts that are used for solving common problems.
- **Methods:**
  - **generate_key() : G**enerates a new fernet key. The key must be kept safe as it is the most important component to decrypt the ciphertext. If the key is lost, then the user can no longer decrypt the message. Also, if an intruder or hacker gets access to the key, they can not only read the data but also forge the data.
  - **encrypt(data) :** It encrypts data passed as **a parameter to the method. The outcome of this encryption is known as a** "Fernet token" which is basically the **ciphertext.** The ***encrypted token also contains the current timestamp when it was generated in plaintext***. The encrypt method throws an exception if the data is not in bytes.
  - **decrypt(token,ttl=None) : D**ecrypts the Fernet token passed as a parameter to the method. On successful decryption the original plaintext is obtained as a result, otherwise an exception is thrown.

Fernet - encryption

The Fernet encryption and authentication process unfolds along the following steps:

1. The **timestamp is recorded**.
2. os.urandom() is used to **generate a unique and sufficiently random initialization vector**.
3. The **ciphertext is constructed**:
   a. The **plaintext is padded** out according to PKCS #7 so that each block is 128-bits.
   b. The **padded message is encrypted** with 128-bit AES in CBC mode, using an encryption key supplied by the user, as well as the initialization vector generated by os.urandom().
4. An **HMAC is computed** for the version, timestamp, initialization vector and ciphertext fields.
5. All of **the above fields, plus the HMAC are concatenated** together.
6. The **token is encoded** according to the base64url specification.

Source: https://www.comparitech.com/blog/information-security/what-is-fernet/

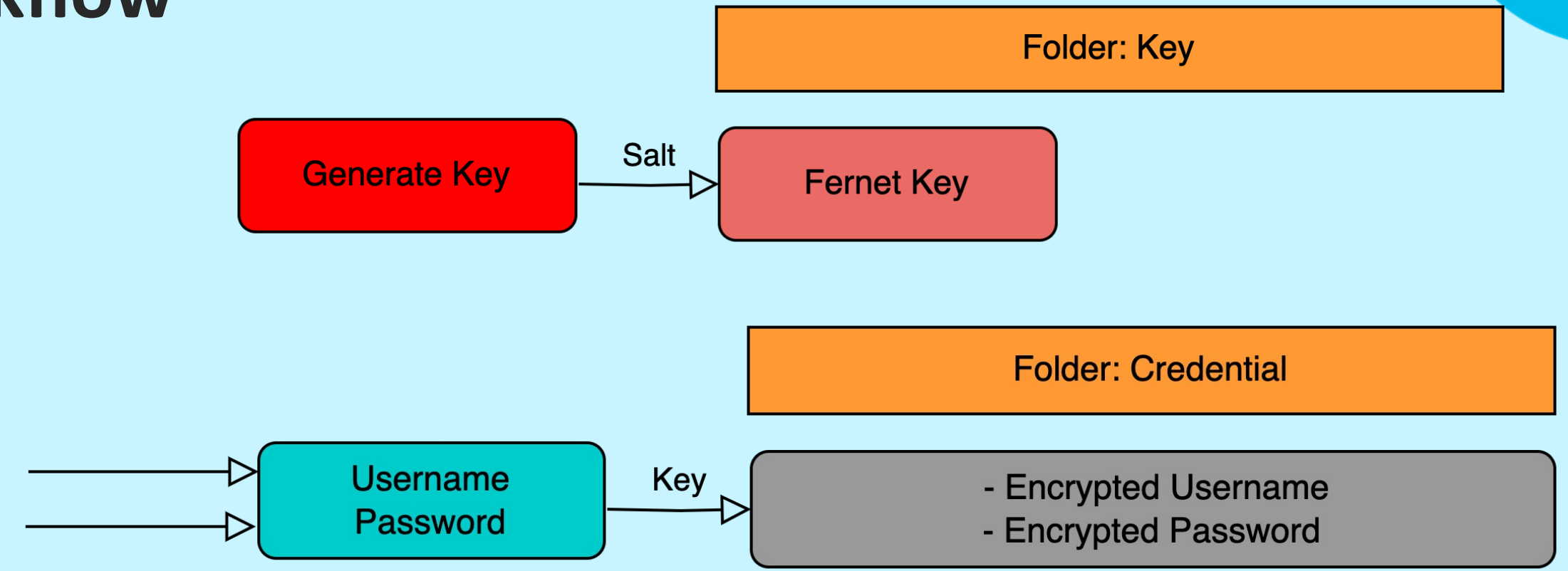
Fernet

- https://cryptography.io/en/latest/fernet/
- https://www.geeksforgeeks.org/fernet-symmetric-encryption-using-cryptography-module-in-python/
- https://www.comparitech.com/blog/information-security/what-is-fernet/
- https://pythonistaplanet.com/fernet/
- https://www.tutorialspoint.com/fernet-symmetric-encryption-using-a-cryptography-module-in-python

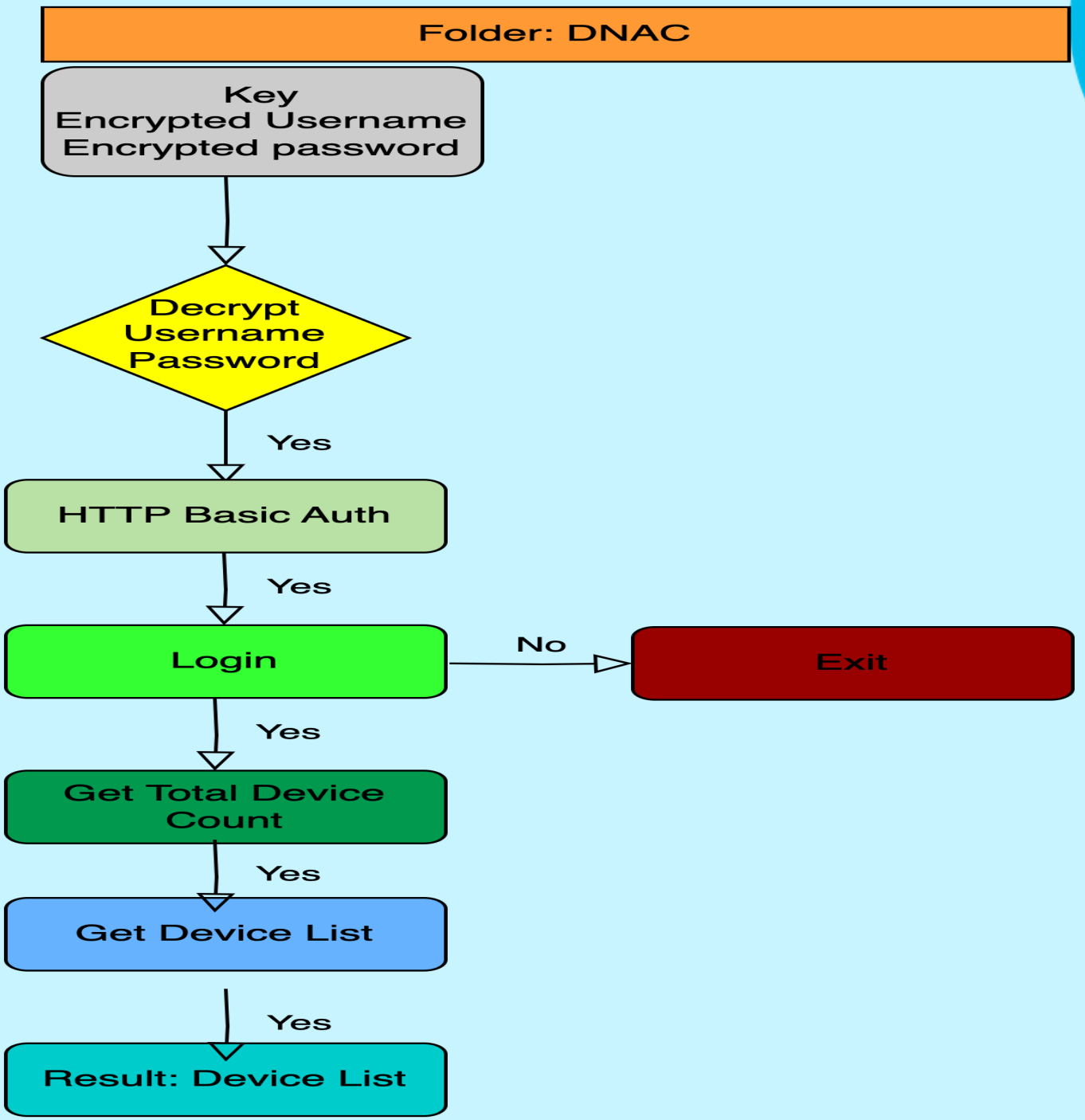
Code-Workflow-Demo

- Salt-Encryption Workflow
- Salt-Decryption Workflow
- Code in Flow chart
- Sequential Execution
- Demo

Salt - Encryption Workflow



Salt-Decryption Workflow



Code in Flow Chart

1. **dnac_encrypted.py**

Get Salt-CipherSuite: Fernet(key)

CipherSuite.decrypt(Encrypted User)

CipherSuite.decrypt(Encrypted Password)

HTTPBasicAuth(Username, Password)

get_dnac_jwt_token

get_network_device_count

get_network_device_list

2. dnac_config.py

DNAC Region,

Encrypted User,

Encrypted Password

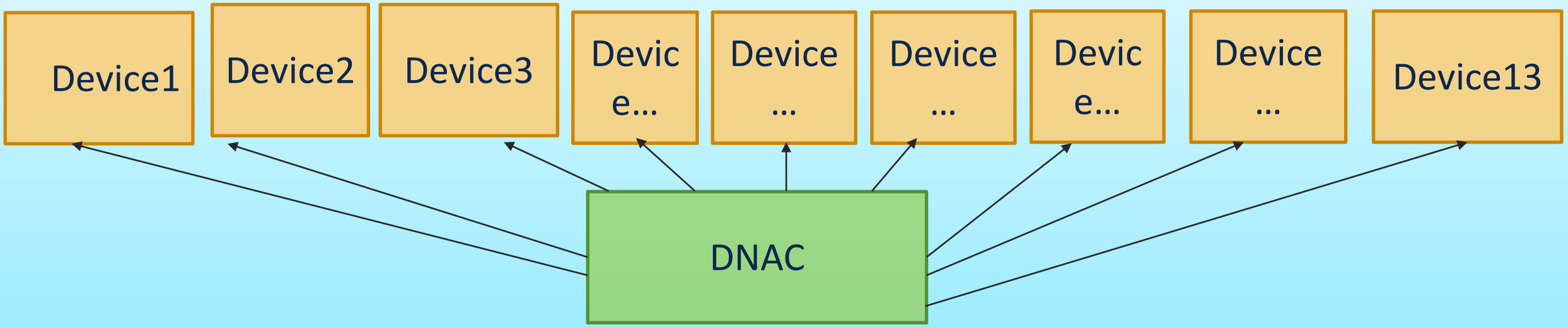
3. dnac_api.py

get_dnac_jwt_token()

get_network_device_count()

get_network_device_list()

Sequential Execution

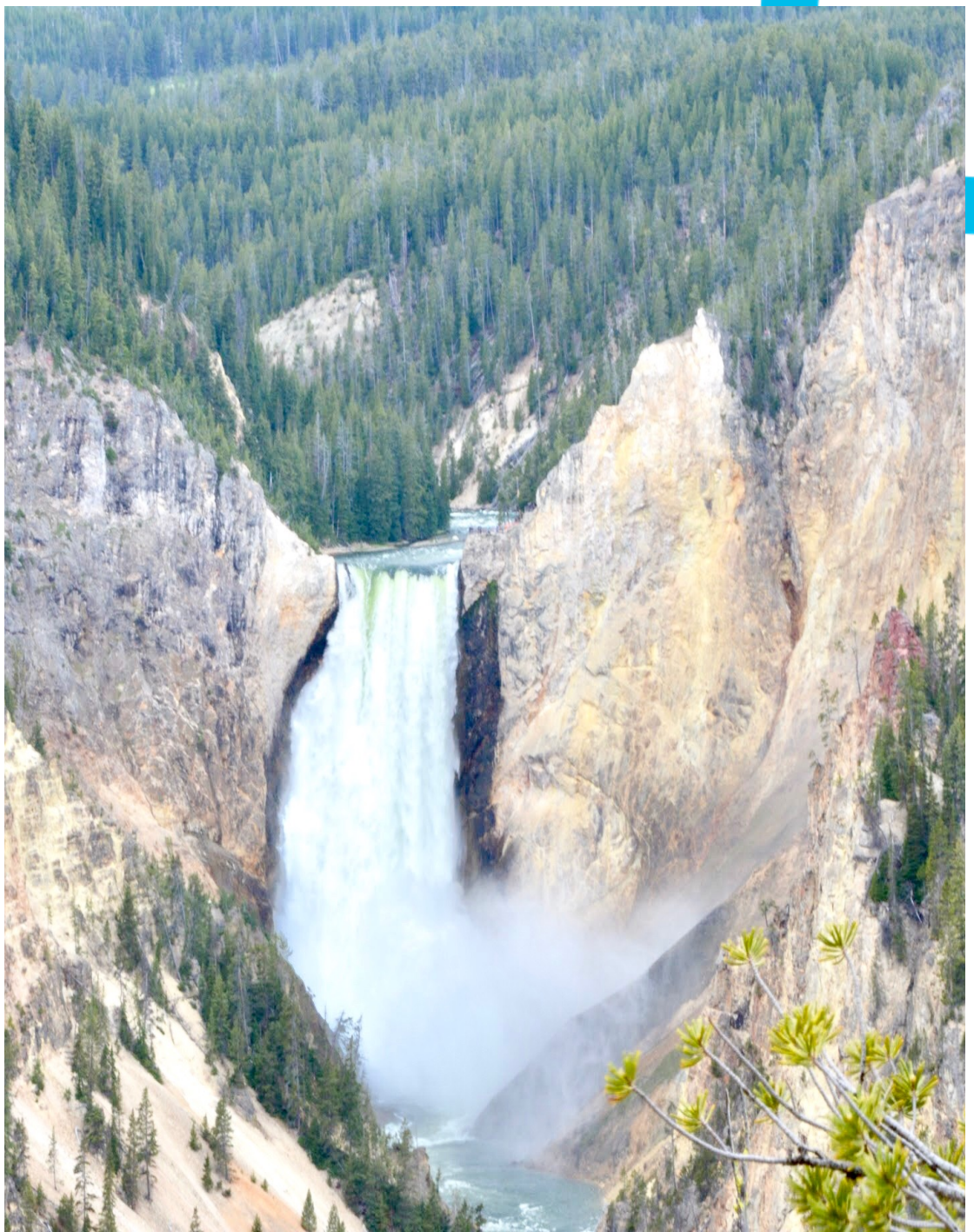


2023-09-13 18:01:09.089 INFO: Completed in 1.090731362986844 seconds: DNAC
2023-09-13 18:01:09.089 INFO: Completed in 1.156683021981734 8 seconds: Access DNAC with Encryption

Demo

CSDL Compliant DNAC Encrypted Credential

- Generate Salt or Key
- Encrypt Username & Password
- Decrypt Username & Password & Run DNAC script



Disclaimer

This document is Cisco Confidential information provided for your internal business use in connection with the Cisco Services purchased by you or your authorized reseller on your behalf. This document contains guidance based on Cisco's recommended practices.

You remain responsible for determining whether to employ this guidance, whether it fits your network design, business needs, and whether the guidance complies with laws, including any regulatory, security, or privacy requirements applicable to your business.

Cisco Secure Development Lifecycle Compliant

1. (venv) (base) SHIRKHAN-M-F1KN:wlc shirkhan$ python generate_key.py. - Salt Key

b'cBGj1GX7K16Na4Ii5MUubS8llEg9aA9aQHFMy6meqLg=‘

2. (venv) (base) SHIRKHAN-M-F1KN:wlc shirkhan$ python generate_enc_credentials.py - username and password – encryption

DNAC Username: admin

Password:

3. Copy encrypted username and password to dnac_config.py file

3. dnac_config.py add

    i. DNAC_USER_REGION1, ii.DNAC_PASS_REGION1

4. python dnac_encrypted.py

Generate Salt Key for Encryption & Encrypt Credentials: Live Run

(encryption) (base) SHIRKHAN-M-F1KN:crypto shirkhan$ python generate_key.py

Completed in 0.00031693698838353157 seconds.

(encryption) (base) SHIRKHAN-M-F1KN:crypto shirkhan$ python generate_enc_credentials.py

DNAC Username: admin

Password:

Completed in 0.14512861898401752 seconds.

Apply and Run DNAC APIs with Encrypted Credentials

```
(encryption) (base) SHIRKHAN-M-F1KN:crypto shirkhan$ python dnac_encrypted.py

2023-09-13 18:01:07.998 INFO: Starting application .....

2023-09-13 18:01:08.692 INFO: Response code: 200

2023-09-13 18:01:08.693 INFO: Response in JSON: {

   "response": 12,

   "version": "1.0"

}

2023-09-13 18:01:08.694 INFO: Total number of Devices: {total}

2023-09-13 18:01:08.694 INFO: Limit:500 offset: 1

2023-09-13 18:01:08.694 INFO: Region url https://10.122.21.78/dna/intent/api/v1/network-device

2023-09-13 18:01:08.694 INFO: Querystring {'limit': 500, 'offset': 1}

2023-09-13 18:01:09.073 INFO: Response code: 200":
```

Apply and Run DNAC APIs with Encrypted Credentials

2023-09-13 18:01:09.088 INFO: Append Device to Device list Length check: 9

2023-09-13 18:01:09.088 INFO: Append Device to Device list Length check: 10

2023-09-13 18:01:09.088 INFO: Append Device to Device list Length check: 11

2023-09-13 18:01:09.088 INFO: Append Device to Device list Length check: 12

2023-09-13 18:01:09.089 INFO: Print Network Device List {'device_total_count': 12, 'devices': ['AP7C21.0EDA.DB2C', 'dc3-wlc01.csspod2.com', 'dc3-wlc02.csspod2.com', 'DNA02-Border1.csspod2.com', 'DNA02-Border2.csspod2.com', 'DNA02-CSRv01.csspod2.com', 'DNA02-Edge1.csspod2.com', 'DNA02-Fusion1.csspod1.com', 'ISR4K-2.csspod2.com', 'POD2-9800-WLC.csspod2.com', 'POD2_Sensor1_RTP60', 'RTP30-FIAB']}

2023-09-13 18:01:09.089 INFO: DNAC Device Inventory Count: 12

2023-09-13 18:01:09.089 INFO: Device_list: {device_list}

2023-09-13 18:01:09.089 INFO: Device List: ['AP7C21.0EDA.DB2C', 'dc3-wlc01.csspod2.com', 'dc3-wlc02.csspod2.com', 'DNA02-Border1.csspod2.com', 'DNA02-Border2.csspod2.com', 'DNA02-CSRv01.csspod2.com', 'DNA02-Edge1.csspod2.com', 'DNA02-Fusion1.csspod1.com', 'ISR4K-2.csspod2.com', 'POD2-9800-WLC.csspod2.com', 'POD2_Sensor1_RTP60', 'RTP30-FIAB']

2023-09-13 18:01:09.089 INFO: Completed in 1.090731362986844 seconds: DNAC

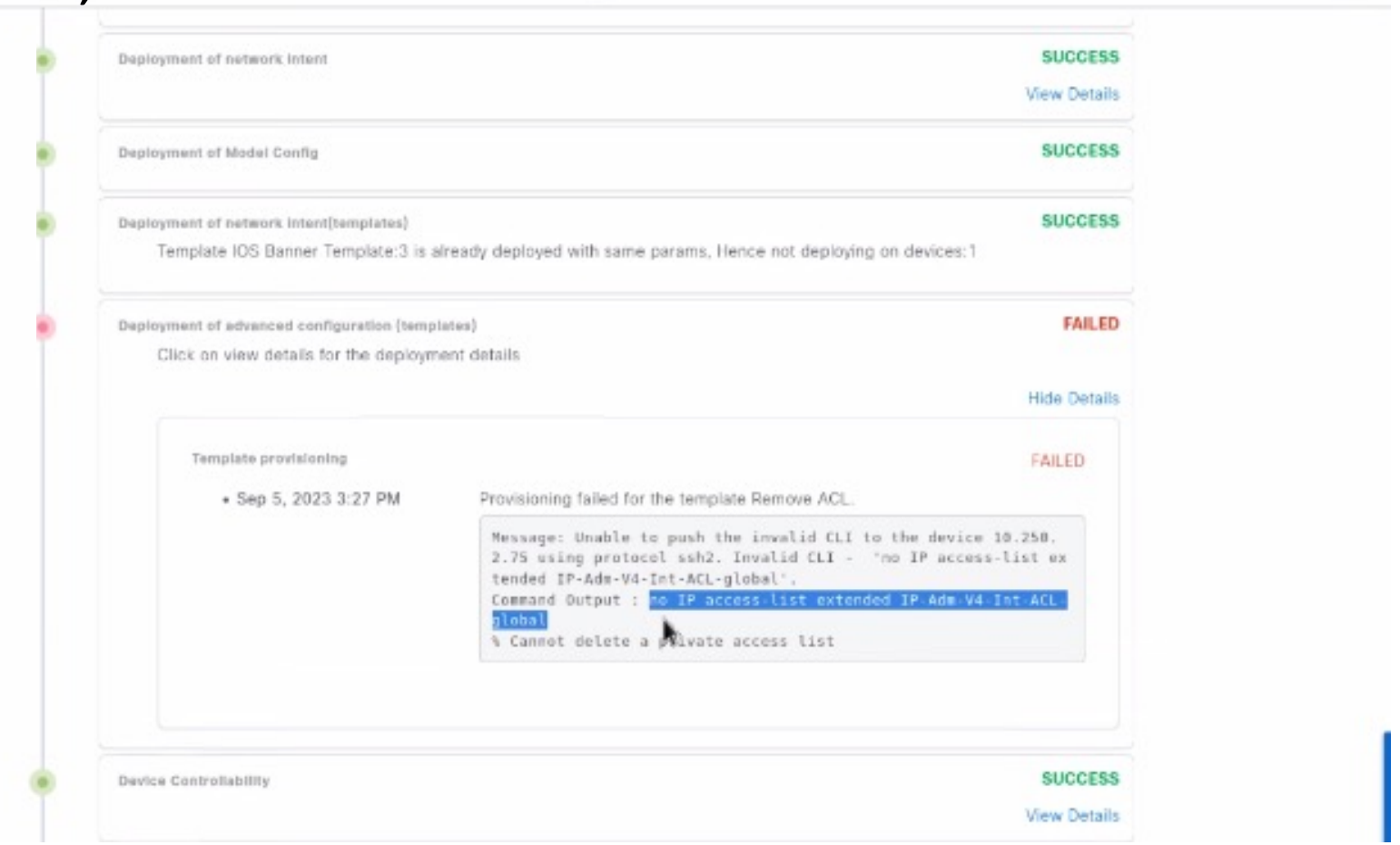
2023-09-13 18:01:09.089 INFO: Completed in 1.156683021981734 8 seconds: Access DNAC with Encryption

Feedback – Q&A

Cannot Delete Private ACLs – Expected Behavior

Does not appear in Running Configuration, old code

```
Standard IP access list ntp_servers
    10 permit 10.250.2.2 (17915 matches)
Standard IP access list snmp-service
    10 permit 10.250.5.66
    20 permit 10.250.2.132
Extended IP access list ACL_SSH_ACCESS_ALLOWED
    5 permit ip host 10.250.2.66 any (47512 matches)
    10 permit ip 10.250.5.16 0.0.0.15 any
    20 permit ip 10.250.2.64 0.0.0.63 any
    30 permit ip 10.250.2.128 0.0.0.63 any
    40 permit ip 10.250.2.0 0.0.0.63 any
    45 permit ip 10.250.4.0 0.0.1.255 any
Extended IP access list IP-Adm-V4-Int-ACL-global
    10 permit tcp any any eq 443
Extended IP access list implicit_deny
    10 deny ip any any
Extended IP access list implicit_permit
    10 permit ip any any
Extended IP access list meraki-fqdn-dns
Extended IP access list preauth_v4
    10 permit udp any any eq domain
    20 permit tcp any any eq domain
    30 permit udp any eq bootps any
    40 permit udp any any eq bootpc
    50 permit udp any eq bootpc any
    60 deny ip any any
Extended IP access list sl_def_acl
    10 deny tcp any any eq telnet
    20 deny tcp any any eq www
    30 deny tcp any any eq 22
```




```
STC_DEV_Switch(config)#no extended ?
% Unrecognized command
STC_DEV_Switch(config)#no IP access
STC_DEV_Switch(config)#no IP access-list extended meraki-fqdn-dns
STC_DEV_Switch(config)#no IP access-list extended IP-Adm-V4-Int-ACL-global
% Cannot delete a private access list
STC_DEV_Switch(config)#
```

Private ACL in Devices

- We cannot add/modify/delete private ACLs. The older releases allowed you to delete these private ACLs without really doing anything in the system.
- Post 17.7, there was an enhancement done to display the message “Cannot modify private access list”
- CSCvs03619 Display warning message for editing the private ACL (preauth_v4)
- The above defect is the reason why the template provisioning is failing as DNAC detects an issue with the template and throws a provisioning error.
- The Customer cannot delete these ACLs. These are system ACLs that are present in the IOS code/version.
- If you check for these ACLs in the running config using “sh run” or “sh run all”, you are not going to see any configuration there.
- And since these configurations do not exist in the running config, you cannot delete them. The only way to see the system ACLs is by using the “sh ip access-list” command.

IP Access List

```
DNA02-Fusion1.csspod1.com> sh ip access-list
Standard IP access list 10
    30 permit 192.168.250.1 (2 matches)
    10 permit 192.168.250.11
    20 permit 192.168.200.0, wildcard bits 0.0.0.255 (9 matches)
Extended IP access list CISCO-CWA-URL-REDIRECT-ACL
    100 deny udp any any eq domain
    101 deny tcp any any eq domain
    102 deny udp any eq bootps any
    103 deny udp any any eq bootpc
    104 deny udp any eq bootpc any
    105 permit tcp any any eq www
Extended IP access list IP-Adm-V4-Int-ACL-global
    10 permit tcp any any eq www
    20 permit tcp any any eq 443
Extended IP access list TEST_TEMPLATE_COMPLIANCE_VLAN230
    10 deny ip any host 1.1.1.1
    40 deny ip any host 1.1.1.2
Extended IP access list implicit_deny
    10 deny ip any any
Extended IP access list implicit_permit
    10 permit ip any any
Extended IP access list meraki-fqdn-dns
Extended IP access list preauth_v4
    10 permit udp any any eq domain
    20 permit tcp any any eq domain
    30 permit udp any eq bootps any
    40 permit udp any any eq bootpc
    50 permit udp any eq bootpc any
    60 deny ip any any
DNA02-Fusion1.csspod1.com>
```

Running Configuration

```
DNA02-Fusion1.csspod1.com> sh run | s access
access-session mac-move deny
 switchport access vlan 2
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
 switchport access vlan 100
 switchport mode access
ip access-list extended TEST_TEMPLATE_COMPLIANCE_VLAN230
 10 remark new_ntp_key_hash_sha1_value
 10 deny   ip any host 1.1.1.1
 40 deny   ip any host 1.1.1.2
 remark new_ntp_key_hash_sha1_value
 remark new_ntp_key_hash_sha3_value
 remark new_ntp_key_hash_sha2_value
 remark new_ntp_key_hash_sha4_value
 remark new_ntp_key_hash_sha5_value
ip access-list standard 10
 30 permit 192.168.250.1
 10 permit 192.168.250.11
 20 permit 192.168.200.0 0.0.0.255
radius-server attribute 8 include-in-access-req
radius-server attribute 25 access-request include
```

THANK YOU for attending today's Session!!

Cisco
Customer Experience