Isilon API SMB Share Management (python)

Overview

This document is provided to enable users to quickly get up to speed with using the Isilon APIs manage SMB shares. This is meant to be a quick start guide and relies on the user's ability to provide the needed environment to work from. The guide also assumes that the user has already completed the document titled "Isilon API Quick Start Guide - python".

Getting Started with the Isilon API

The following code sections can be copy/pasted or used simply as an explanation. There is a complete (variables will need changed) script at the end of this document.

Define Variables for the Cluster

Change these variables to match the Isilon Cluster environment

```
CLUSTERIP = '172.16.10.10'
PORT=8080
USER='root'
PASS='a'
```

Session Setup and Auth

```
import requests
import json
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) # Supresses
the self signed cert warning

uri = "https://%s:%s" % (CLUSTERIP, PORT)
papi = uri + '/platform'
headers = {'Content-Type': 'application/json'}
data = json.dumps({'username': USER, 'password': PASS, 'services': ['platform']})

session = requests.Session()
response = session.post(uri + "/session/1/session", data=data, headers=headers,
verify=False)
session.headers['referer'] = uri
session.headers['X-CSRF-Token'] = session.cookies.get('isicsrf')
```

Establish Session

Once the Authorization is successful and we have a good session with CSRF headers we can use that session to make repeated API calls for up to 24h or until we close the session.

```
# uri of the cluster used in the referer header
uri = f"https://{CLUSTERIP}:{PORT}"
# url of Papi used for all further calls to Papi
papi = uri + '/platform'
# Set header as content will provided in json format
headers = {'Content-Type': 'application/json'}
# Create json dictionary for auth
data = json.dumps({'username': USER, 'password': PASS, 'services': ['platform']})
# create a session object to hold cookies
session = requests.Session()
# Establish session using auth credentials
response = session.post(uri + "/session/1/session", data=data, headers=headers,
verify=False)
if 200 <= response.status_code < 299:
    # Set headers for CSRF protection. Without these two headers all further
calls with be "auth denied"
    session.headers['referer'] = uri
    session.headers['X-CSRF-Token'] = session.cookies.get('isicsrf')
    print("Authorization Successful")
else:
    print("Authorization Failed")
    print(response.content)
```

List Shares

```
endpoint = '/6/protocols/smb/shares'
response = session.get(papi + endpoint, verify=False)
shares = response.json()['shares']
for share in shares:
    print(f"Share: {share['name']} - {share['description']} - {share['path']}")
```

Create Share

```
name = 'test'
path = '/ifs/test'
data = json.dumps({ 'name': name, 'path': path})
response = session.post(papi + endpoint, data=data, headers=headers,
verify=False)
if 200 <= response.status_code < 299:
    print("Share Created Successfully")
else:
    print(f"Error Creating Share: {response.content}")
```

Delete a Share

```
name = 'test'
response = session.delete(papi + endpoint + '/' + name, data=data, headers=head-
ers, verify=False)
if 200 <= response.status_code < 299:
    print("Share Delete Successfully")
else:
    print(f"Error Deleting Share: {response.content}")
```

Full Scripts

List Shares

```
CLUSTERIP = '172.16.10.10'
PORT=8080
USER='root'
PASS='a'


import requests
import json
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) # Supresses
the self signed cert warning

uri = "https://%s:%s" % (CLUSTERIP, PORT)
papi = uri + '/platform'
headers = {'Content-Type': 'application/json'}
data = json.dumps({'username': USER, 'password': PASS, 'services': ['platform']})

session = requests.Session()
response = session.post(uri + "/session/1/session", data=data, headers=headers,
verify=False)
session.headers['referer'] = uri
session.headers['X-CSRF-Token'] = session.cookies.get('isicsrf')

endpoint = '/6/protocols/smb/shares'

response = session.get(papi + endpoint, verify=False)
result = json.loads(response.content)

for share in result['shares']:
    print("%s: %s" % (share['name'], share['path']))
```

Create Share

```
CLUSTERIP = '172.16.10.10'
PORT=8080
USER='root'
PASS='a'

name = 'test'
path = '/ifs/test'

import requests
import json
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) # Supresses
the self signed cert warning

uri = "https://%s:%s" % (CLUSTERIP, PORT)
papi = uri + '/platform'
headers = {'Content-Type': 'application/json'}
data = json.dumps({'username': USER, 'password': PASS, 'services': ['platform']})

session = requests.Session()
response = session.post(uri + "/session/1/session", data=data, headers=headers,
verify=False)
session.headers['referer'] = uri
session.headers['X-CSRF-Token'] = session.cookies.get('isicsrf')

endpoint = '/6/protocols/smb/shares'

data = json.dumps({ 'name': name, 'path': path})
response = session.post(papi + endpoint, data=data, headers=headers,
verify=False)

if response.status_code == 201:
    print('Success!')
```

Delete Share

```
CLUSTERIP = '172.16.10.10'
PORT=8080
USER='root'
PASS='a'

name = 'test'

import requests
import json
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) # Supresses
the self signed cert warning

uri = "https://%s:%s" % (CLUSTERIP, PORT)
papi = uri + '/platform'
headers = {'Content-Type': 'application/json'}
data = json.dumps({'username': USER, 'password': PASS, 'services': ['platform']})

session = requests.Session()
response = session.post(uri + "/session/1/session", data=data, headers=headers,
verify=False)
session.headers['referer'] = uri
session.headers['X-CSRF-Token'] = session.cookies.get('isicsrf')

endpoint = '/6/protocols/smb/shares'

response = session.delete(papi + endpoint + '/' + name, data=data,
headers=headers, verify=False)

if response.status_code == 204:
    print("Success!")
```