Isilon API Quick Start Guide (python)

Overview

This document is provided to enable users to quickly get up to speed with using the Isilon APIs to automate tasks. 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 listed pre-requirements are only one of many paths available when using the Isilon API

Pre-Requirements

For the purposes of simplicity this document will focus on execution of python code for the examples. Other languages can be used but will require different environments. Download and install from each category below:

Python Interpreter

For this example, a python interpreter as well as the python package manager (pip) will be required. When setting up python on Windows one of the easier options is to use Anaconda. Anaconda comes with support for Jupyter Lab (browser based python execution environment) as well as support for Visual Studio Code. Additionally, python.org provides installers for Windows and will integrate with VS Code as well.

Python Module

For these examples we will be using the python "requests" to make the REST API calls. The requests library can be installed via pip (or pip3 depending):

```
pip install requests
```

Editor (IDE)

An editor or Integrated Development Environment is helpful for editing, syntax checking, executing and debugging code within a single view. There are a number of choices available, but a couple of the preferred options are Atom and VS Code.

Isilon Cluster

Access to an Isilon Cluster to interact with is a necessity. The cluster should be reachable over the network from the system being used to test with. It is recommended that testing be done against a non-production cluster or Isilon Simulator.

Pre-Requirements Test

Once the python environment, editor and module have been installed then paste the following test code and execute it with python:

```
import requests
print("Hello World")
```

The result should print out "Hello World (without quotes).

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'
```

Module Imports

This is where the modules required are imported and we add a call to suppress the https insecure certificate warning.

```
import requests
import json
import urllib3

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

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)
```

API Call to /3/cluster/config

```
response = session.get(papi + '/3/cluster/config', verify=False)
if 200 <= response.status_code < 299:
    print(response.json())
else:
    print(f"Error: {response.content}")
```

Full Example Script

```
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 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)

response = session.get(papi + '/3/cluster/config', verify=False)
if 200 <= response.status_code < 299:
    print(response.json())
else:
    print(f"Error: {response.content}")
```