hpestorapi python library manual

Release 0.9.8

Hewlett Packard Enterprise Development

Apr 11, 2020

CONTENTS:

CHAPTER ONE

PACKAGE DESCRIPTION

hpestorapi - python library that provides very simple way to use Rest API services for HPE storage and disk backup devices. Current version supports:

- HPE 3PAR StoreServ disk arrays
- HPE Primera disk arrays
- HPE XP7 and P9500 (Command View AE Configuration manager is required)
- HPE StoreOnce Gen 3 disk backup device
- HPE StoreOnce Gen 4 disk backup device

CHAPTER TWO

INSTALLATION

2.1 Requirements

hpestorapi library depends on:

- Python 3.6 or newer
- Python requests library version 2.19.1 or newer

2.2 Install from PyPI

To download and install hpestorapi you can use pip:

```
# pip install hpestorapi
```

2.3 Install from GitHub

Get a copy of source code

```
# git clone https://github.com/HewlettPackard/python-storage-clients.git
# cd python-storage-clients
```

Install package with dependencies:

```
# python setup.py install
```

Import hpestorapi library in your python script:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import hpestorapi
```

CHAPTER

THREE

HPE 3PAR STORESERV DISK ARRAY

3.1 API reference

class hpestorapi.**StoreServ**(*address*, *username*, *password*, *port=None*, *ssl=True*, *verify=True*)
HPE 3PAR array implementation class.

__init__(*address*, *username*, *password*, *port=None*, *ssl=True*, *verify=True*)
HPE 3PAR constructor.

Parameters

- **address** (*str*) – Hostname or IP address of HPE 3PAR array (management address). Web Services API should be enabled for this array (disabled by default). To enable Web Services API you should check 3PAR OS command: showwsapi.
- **username** (*str*) – User name for 3PAR Web Services API. Its recommended to create dedicated user with limited rights. For example, if you dont need to create/modify/delete objects on disk array, you should create new user with "browse" role. Of coarse, your script can work with "3paradm" user ("super" role), but its a bad idea. To create new user, you should check 3PAR OS command: createuser.
- **password** (*str*) – Password for 3PAR Web Services API.
- **port** (*int*) – (optional) Custom port number for 3PAR Web Services API.
- **ssl** (*bool*) – (optional) Use secure https (True) or plain text http (False).
- **verify** (*bool|string*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: True.

Returns None

open()
Open new Rest API session for HPE 3PAR array. You should call it prior any other requests. Do not forget to call `StoreServ.close()` if you don't plan to use session anymore, because 3PAR array has active sessions limit.

If some troubles occurs you should manually check:

- 3PAR Web services API are enabled on array (3PAR OS command: 'showwsapi')
- Array credentials (username and password)
- 3PAR array management address is correct and available
- Debug logs generated by python logging module

Returns None

close()
Close Rest API session.

Returns None

get (*url*, *query=None*)
Perform HTTP GET request to HPE 3PAR array. Method used to get information about objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'system' or 'volumes'. All available url's and requests result are described in "HPE 3PAR Web Services API Developer's Guide"
- **query** (*str*) – (optional) Query filter specification (see "WSAPI query syntax" in "HPE 3PAR Web Services API Developer's Guide").

Return type tuple(int, dict)

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'}).

post (*url*, *body*)
Perform HTTP POST request to HPE 3PAR array. Method used to create new objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'system' or 'volumes'. All available url's, request parameters and results are described in "HPE 3PAR Web Services API Developer's Guide"
- **body** (*dict*) – Request parameter, used to create new array object.

Return type tuple (int, dict)

Returns Tuple with HTTP status code and dict with request result. For example: (201, {'key':'value'}). Second value may be None if 3PAR array returns no message body.

delete (*url*)
Perform HTTP DELETE request to HPE 3PAR array.

Parameters **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'system' or 'volumes'. All available url's, request parameters and results are described in "HPE 3PAR Web Services API Developer's Guide"

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'}). Second value may be None if 3PAR array returns no message body.

put (*url*, *body*)
Perform HTTP PUT request to HPE 3PAR array.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'system' or 'volumes'. All available url's, request parameters and results are described in "HPE 3PAR Web Services API Developer's Guide"
- **body** (*dict*) – Request parameter, used to modify array object.

Return type tuple(int, dict)

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'}). Second value may be None if 3PAR array returns no message body.

property timeout

Variables ***timeout*** (*float | tuple*) – Number of seconds that Rest API client waits for response from HPE StoreServ before timeout exception generation. You can use different timeouts for connection setup and for getting first piece of data. In this case, you should use tuple(float, float) with first value - connection timeout and the second value - read timeout. Or if you want to use same values for both type of timeouts, you can use one float value. ‘None’ value can be used instead to wait forever for a device response. Default value: (1, None)

3.2 Usage

3.2.1 Session management

Simple way to open Rest API session for StoreServ 3PAR arrays (with exception handling and session auto-closing):

```
from hpestorapi import StoreServ

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        # Perform requests to array (get/post/put/delete)
        # ...
    except Exception as error:
        print(error)
    else:
        # Analyze array response
        # ...
```

3.2.2 GET request

Simple GET request usage. This code print some storage system information (name, serial number and ip address):

```
from hpestorapi import StoreServ

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        status, data = array.get('system')
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            print('Name=%s; ' % data['name'],
                  'SerialNumber=%s; ' % data['serialNumber'],
                  'Address=%s' % data['IPv4Addr']
                  )
```

GET request can also contain filter parameter (query='…'). Filter specifications are described in “HPE 3PAR Web Services API Developer’s Guide” (see section “WSAPI query syntax”). This code print Remote Copy Groups names beggining with “dfs”.

```
from hpestorapi import StoreServ

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        status, data = array.get('remotecopygroups', query='name LIKE <dfs*>')
    except Exception as error:
        print('Something went wrong:')
        raise error
```

(continues on next page)

(continued from previous page)

```
    else:
        if status == 200:
            for rc_group in data['members']:
                print('RC group name = ', rc_group['name'])
```

3.2.3 POST request

This code will create new host on the 3PAR array:

```
from hpestorapi import StoreServ

newhost = {
    'name': 'restapi-test',
    'persona': 2,
    'FCWWNs': ['50:01:55:55:55:55:55:55',
               '50:01:66:66:66:66:66:66']
}

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        status, data = array.post('hosts', newhost)
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 201:
            print('Success')
        else:
            print('Failed! Device response: ', data)
```

3.2.4 DELETE request

This code will delete host from 3PAR array:

```
from hpestorapi import StoreServ

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    hostname = 'restapi-test'
    try:
        array.open()
        status, data = array.delete(f'hosts/{hostname}')
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            print('Success')
        else:
            print('Fail! StoreServ 3PAR response: ', data)
```

3.2.5 PUT request

This code will modify host on 3PAR array (change host persona and location description):

```
from hpestorapi import StoreServ

hostname = 'restapi-test'
modify = {
    'persona': 1,
    'descriptors': {'location': 'Rack 2/42, Unit 34'}
}

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        status, data = array.put(f'hosts/{hostname}', body=modify)
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            print('Success')
        else:
            print('Fail! Device response: ', data)
```

3.2.6 Exception handling

Almost all methods, that perform interaction with StoreServ 3PAR array can generate exceptions. The best practice is to handle these exceptions.

List of method, that can raise exceptions:

- `StoreServ.open()`
- `StoreServ.get()`
- `StoreServ.post()`
- `StoreServ.delete()`
- `StoreServ.put()`

Note: StoreServ class object constructor does not perform any requests to array, therefore StoreServ object initilization can't raise exceptions.

Exception handling example:

```
import requests

from hpestorapi import StoreServ

with StoreServ('10.0.0.1', '3paruser', '3parpass') as array:
    try:
        array.open()
        status, data = array.get('system')
    except requests.exceptions.SSLError as error:
        print('SSL error occured. Please check SSL connection '
              'options.')
    except requests.exceptions.ReadTimeout:
        print('Read timeout occured. The StoreServ 3PAR array did '
              'not send any data in the alloted amount of time.')
```

(continues on next page)

(continued from previous page)

```
    except requests.exceptions.ConnectTimeout as error:
        print('Connection timeout occured. The request timed out '
              'while trying to connect to the StoreServ 3PAR '
              'array.')
    except hpestorapi.storeserv.AuthError as error:
        print('Wrong user name or password for the StoreServ 3PAR '
              'array.')
    except Exception as error:
        print(error)
    else:
        # We are successfully received response from array. You can
        # safely analyze array response (status and data variables)
        # ...
```

CHAPTER

FOUR

HPE PRIMERA DISK ARRAY

HPE Primera has minimal WSAPI changes compared to HPE 3PAR. To use Primera WSAPI you should use class `hpestorapi.Primera` with an interface similar to class:*hpestorapi.StoreServ*

CHAPTER

FIVE

HPE XP7 AND P9500 DISK ARRAY

5.1 API reference

class hpestorapi.**Xp**(*cvae*, *svp*, *serialnum*, *username*, *password*, *gen='XP7'*, *port=23451*, *ssl=True*)

XP7 / P9500 class implementation.

__init__(*cvae*, *svp*, *serialnum*, *username*, *password*, *gen='XP7'*, *port=23451*, *ssl=True*)

HPE XP constructor.

Parameters

- **cvae** (*str*) – Hostname or IP address of HPE Command View AE Configuration Manager.
- **svp** (*str*) – Hostname or IP address of Service Processor (SVP).
- **serialnum** (*str*) – Array serial number (5 or 6 digits)
- **username** (*str*) – User name for HPE XP disk array. Its recommended to create dedicated user with limited rights. It's a bad idea to use "arrayadmin" account.
- **password** (*str*) – Password for HPE XP disk array.
- **gen** (*str*) – (optional) Disk array generation. Should be P9500 or XP7. By default: XP7.
- **port** (*int*) – (optional) Command View AE configuration manager port. By default: 23451
- **ssl** (*bool*) – (optional) Use secure https (True) or plain http. By default: True.

Returns None.

property http_timeout

Number of seconds that Rest API client waits for http(s) reesponse from Configuration Manager. By default: 120 sec

open()

Open new Rest API session for HPE XP array. You should call it prior any other requests. Do not forget to call `Xp.close()` if you don't plan to use session anymore.

Return type bool

Returns Return True, if disk array provide valid session key.

close(*passive=False*)

Close Rest API session.

Parameters **passive** (*bool*) – (optional) Do not try to close session in array (just forget client key when your session is already timed out in array). This parameter may be usefull *only for method overriding* in subclasses. It completely useless for you application, because class implementation maintain session live transparent for you. False by default.

Returns None

get (*url*, ***kwargs*)

Perform HTTP GET request to HPE XP array. This method used to get information about array objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'pools', 'parity-groups', 'ldevs'. All available url's and requests result are described in "HPE XP7 Command View Advanced Edition REST API Reference Guide".
- **params** (*dict*) – (optional) Dictionary with query filter parameters. Described in 'Query parameters' section of "HPE XP7 Command View Advanced Edition REST API Reference Guide".
- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **timeout** (*float*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as *Xp.http_timeout*.
- **verify** (*bool*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: False (do not check certificate).
- **cert** (*str*) – (optional) String with path to ssl client certificate file (.pem) or tuple pair ('cert', 'key').

Return type (int, {})

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'})

post (*url*, ***kwargs*)

Perform HTTP POST request to HPE XP array. This method used to create new object.

Parameters

- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **timeout** (*float*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as *Xp.http_timeout*.
- **verify** (*bool*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: False (do not check certificate).
- **cert** (*str*) – (optional) String with path to ssl client certificate file (.pem) or tuple pair ('cert', 'key').

Return type (int, {})

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'})

delete (*url*, ***kwargs*)

Perform HTTP DELETE request to HPE XP array. This method used to remove objects.

Parameters

- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **timeout** (*float*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as *Xp.http_timeout*.

- **verify** (*bool*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: False (do not check certificate).
- **cert** (*str*) – (optional) String with path to ssl client certificate file (.pem) or tuple pair ('cert', 'key').

Return type (int, {})

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'})

put (*url*, ***kwargs*)

Perform HTTP PUT request to HPE XP array.

Parameters

- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **timeout** (*float*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as http_timeout.
- **verify** (*bool*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: False (do not check certificate).
- **cert** (*str*) – (optional) String with path to ssl client certificate file (.pem) or tuple pair ('cert', 'key').

Return type (int, {})

Returns Tuple with HTTP status code and dict with request result. For example: (200, {'key':'value'})

5.2 Usage

5.2.1 Session management

Open Rest API session for XP array:

```python
import hpestorapi

with hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                   'arrayuser', 'arraypassword') as array:
    try:
        array.open()
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        # Perform requests to array (get/post/put/delete)
        # ...
```

5.2.2 GET request

Simple GET request usage. This code print pool list from HPE XP array:

```
import hpestorapi

with hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                   'arrayuser', 'arraypassword') as array:
    try:
        array.open()
        status, body = array.get('pools')
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            for pool in body['data']:
                print('ID=%s ' % pool['poolId'],
                      'Name=%s ' % pool['poolName'],
                      'Type=%s' % pool['poolType']
                      )
```

5.2.3 POST request

This code will create new ldev on the XP array:

```
import hpestorapi

newvol = {'poolId': 1, 'byteFormatCapacity': '1G'}

with hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                   'arrayuser', 'arraypassword') as array:
    try:
        array.open()
        status, body = array.post('ldevs', json=newvol)
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            print('Success')
        else:
            print('Cannot create ldev')
```

5.2.4 DELETE request

This code will delete ldev 62:

```
import hpestorapi

ldevid=62

array = hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                      'arrayuser', 'arraypassword') as array:
    try:
        array.open()
        status, body = array.post(f'ldevs/62')
    except Exception as error:
        print('Something went wrong:')
```

(continues on next page)

(continued from previous page)

```
        raise error
    else:
        if status == 200:
            print('Success')
        else:
            print(f'Cannot delete ldev with id {ldevid}')
```

5.2.5 PUT request

This code will change label for ldev 62:

```
import hpestorapi

ldevid = 62
settings = {'label': 'RestAPI_Test'}

with hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                   'arrayuser', 'arraypassword') as array:
    try:
        array.open()
        status, body = array.put(f'ldevs/{ldevid}', json=settings)
    except Exception as error:
        print('Something went wrong:')
        raise error
    else:
        if status == 200:
            print('Success')
        else:
            print('Cannot create ldev')
```

5.2.6 Exception handling

```
import requests
import hpestorapi

with hpestorapi.Xp('cvae.domain.com', 'svp.domain.com', '123456',
                   'arrayuser', 'arraypassword') as array:
    try:
        array.open()
    except requests.exceptions.SSLError as error:
        print('Cannot connect to Configuration Manager. SSL cert '
              'cheking is enabled, but Rest API server has no '
              'valid SSL certificate.')
    except requests.exceptions.ReadTimeout:
        timeout = array.http_timeout
        print('Read timeout occured. The Rest server did not '
              'send any data in the allotted amount of time ',
              timeout)
    except requests.exceptions.ConnectTimeout as error:
        print('Connection timeout occured. The request timed out '
              'while trying to connect to the Rest server.')
    except hpestorapi.Xp.AuthError as error:
        print('Wrong username or password for the HPE XP array')
    except Exception as error:
        print(error)
    else:
        # Perform requests to array (get/post/put/delete)
        # ...
```

CHAPTER

SIX

HPE STOREONCE GEN 3 DISK BACKUP

6.1 API references

class `hpestorapi.`**`StoreOnceG3`**(*address*, *user*, *password*, *cookie_dir=None*, *port=443*)

HPE StoreOnce Gen 3 disk backup device implementation class.

`__init__`(*address*, *user*, *password*, *cookie_dir=None*, *port=443*)

HPE StoreOnceG3 constructor.

Parameters

- **`address`** (*`str`*) – Hostname or IP address of HPE StoreOnce disk backup device.
- **`user`** (*`str`*) – Username for HPE StoreOnce disk backup device.
- **`password`** (*`str`*) – Password for HPE StoreOnce disk backup device.
- **`cookie_dir`** (*`str`*) – User name for HPE XP disk array. Its recommended to create dedicated user with limited rights. It's a bad idea to use "arrayadmin" account.
- **`port`** (*`int`*) – (optional) Custom port number for StoreOnce device.

Returns None.

`query`(*url*, *method*, ***kwargs*)

`get`(*url*, ***kwargs*)

Perform HTTP GET request to HPE Storeonce disk backup device. Method used to get information about objects.

Parameters

- **`url`** (*`str`*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'cluster' or 'cluster/servicesets'. All available url's and requests result are described in "HPE StoreOnce 3.18.2 REST API developer Guide"
- **`filter`** (*`RequestsCookieJar`*) – (optional) Filter cookies, that was generated by method `StoreOnceG3.filter()`
- **`timeout`** (*`int`*) – (optional)

Return type (int, string)

Returns Tuple with HTTP status code and xml string with request result. For example: (200, '<xml> … </xml>').

`post`(*url*, ***kwargs*)

Perform HTTP POST request to HPE Storeonce disk backup device. Method used to create new objects.

Parameters **`url`** (*`str`*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: 'cluster' or 'cluster/servicesets'.

All available url's and requests result are described in "HPE StoreOnce 3.18.2 REST API developer Guide"

Return type (int, string)

Returns Tuple with HTTP status code and xml string with request result. For example: (200, '<xml> … </xml>').

put (*url*, ***kwargs*)

delete (*url*, ***kwargs*)

open (*cookie_load=True*)

iterate (*url*, *items*)

property timeout

Variables ***timeout*** (*float | tuple*) – Number of seconds that Rest API client waits for response from HPE StoreOnce Gen 3 before timeout exception generation. You can use different timeouts for connection setup and for getting first piece of data. In this case, you should use tuple(float, float) with first value - connection timeout and the second value - read timeout. Or if you want to use same values for both type of timeouts, you can use one float value. 'None' value can be used instead to wait forever for a device response. Default value: (1, None).

6.2 Usage

6.2.1 Session management

Open Rest API session for StoreOnce Gen 4 disk backup device:

```
import hpestorapi

with hpestorapi.StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        # Perform requests to StoreOnce (get/post/put/delete)
        # ...
```

6.2.2 GET request

Simple GET request usage. This code print StoreOnce G3 Catalyst Stores:

```
import xml.etree.ElementTree
import hpestorapi

with hpestorapi.StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        status, data = so.get('/cluster')
        if status == 200:
```

(continues on next page)

(continued from previous page)

```
            tree = xml.etree.ElementTree.fromstring(data)
            name = tree.find('./cluster/properties/applianceName').text
            model = tree.find('./cluster/properties/productClass').text
            print(f'SO Name = "{name}"')
            print(f'SO Model = "{model}"')
```

6.2.3 POST request

POST request usage. This code activate license for StoreOnce G3:

```
import hpestorapi

body = {'key': 'demo', 'opcode': 'add'}

with hpestorapi.StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
        status, data = so.post('/d2dlicenses', params=body)
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        if status == 200:
            print('Demo license activation success.')
        else:
            print('License activation failed.'
                  'Http code: %s. Response body: %s',
                  status, data)
```

6.2.4 DELETE request

This code remove from Service Set 1 Catalyst Store 0:

```
import hpestorapi

with hpestorapi.StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        sset = 1
        store = 0
        status, data = so.delete(f'/cluster/servicesets/{sset}'
                                 f'/services/cat/stores/{store}')
        if status == 204:
            print('Catalyst store deleted')
        else:
            print('Can not delete catalyst store.'
                  'Http code: %s. Response: %s',
                  status, data)
```

6.2.5 PUT request

This code add NTP server to StoreOnce G3 configuration:

```
import hpestorapi

with hpestorapi.StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        print('Something went wrong:')
        so.open()
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        body = {'add': 'true', 'servers': '10.0.0.2'}
        status, data = so.put('/d2dservices/clusterconf/ntp',
                              params=body)
        if status == 200:
            print('Catalyst store deleted')
        else:
            print('Can not delete catalyst store.'
                  'Http code: %s. Response body: %s',
                  status, data)
```

6.2.6 Iterate multipage objects

This code print all Catalyst items from all Catalyst stores:

```
from hpestorapi import StoreOnceG3
from hpestorapi.storeonce3 import Iterator
import xml.etree.ElementTree

with StoreOnceG3('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        # Print table header row
        print('%15s %8s %30s %15s' % ('Store', 'ID', 'Name', 'Status'))

        # Iterate all ServiceSets
        for sset_xml in Iterator(so, '/cluster/servicesets',
                                 './servicesets/serviceset'):
            sset = xml.etree.ElementTree.fromstring(sset_xml)
            ssid = sset.find('./properties/ssid').text

            # Iterate all catalyst Stores
            store_url = (f'/cluster/servicesets'
                         f'/{ssid}/services/cat/stores/')
            for store_xml in Iterator(so, store_url, './stores/store'):
                store = xml.etree.ElementTree.fromstring(store_xml)
                store_id = store.find('./properties/id').text
                store_name = store.find('./properties/name').text

                # Iterate all Catalyst Items
                item_url = (f'/cluster/servicesets/{ssid}'
                            f'/services/cat/stores/{store_id}/items/')
                for item_xml in Iterator(so, item_url, './items/item'):
                    item = xml.etree.ElementTree.fromstring(item_xml)
```

(continues on next page)

(continued from previous page)

```
                item_id = item.find('./properties/id').text
                item_name = item.find('./properties/name').text
                item_status = item.find('./properties/status').text

                print('%15s %8s %30s %15s' % (store_name, item_id,
                                              item_name, item_status))
```

CHAPTER

SEVEN

HPE STOREONCE GEN 4 DISK BACKUP

7.1 API references

class hpestorapi.**StoreOnceG4**(*address*, *username*, *password*)

HPE StoreOnce Gen 4 backup device implementation class.

__init__(*address*, *username*, *password*)

HPE StoreOnce Gen 4 disk backup constructor.

Parameters

- **address** (*str*) – Hostname or IP address of HPE StoreOnce device.
- **username** (*str*) – Username for HPE StoreOnce device.
- **password** (*str*) – Password for HPE StoreOnce device.

Returns None.

open(*verify=False*)

Open new Rest API session for HPE StoreOnce Gen 4 disk backup. You should call it prior any other requests. Do not forget to call `StoreOnceG4.close()` if you don't plan to use current session anymore.

Parameters **verify** (*bool | str*) – (optional) Either a boolean, in which case it controls whether we verify the Rest server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. By default: False (do not check certificate).

Returns None

close()

Close Rest API session.

Returns None

get(*url*, ***kwargs*)

Perform HTTP GET request to HPE Storeonce G4 disk backup device. This method used to get information about objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: '/rest/alerts' or '/api/v1/management-services/licensing'. All available url's and requests result are described in HPE StoreOnce REST API
- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **params** (*dict*) – (optional) Dictionary with url encoded parameters.
- **timeout** (*float | tuple*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as `StoreOnceG4.timeout`.

Return type tuple(int, dict())

Returns Tuple with HTTP status code and dict with request result. For example: (200, {. . . }) or (204, None).

post (*url*, ***kwargs*)

Perform HTTP POST request to HPE StoreOnce G4 disk backup device. This method used to create new object.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: '/rest/alerts' or '/api/v1/management-services/licensing'. All available url's and requests result are described in HPE StoreOnce REST API
- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **params** (*dict*) – (optional) Dictionary with url encoded parameters.
- **timeout** (*float|tuple*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as `StoreOnceG4.timeout`.

Return type tuple(int, dict())

Returns Tuple with HTTP status code and dict with request result. For example: (200, {. . . }) or (204, None).

delete (*url*, ***kwargs*)

Perform HTTP DELETE request to HPE StoreOnce G4 disk backup device array. This method used to remove objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: '/rest/alerts' or '/api/v1/management-services/licensing'. All available url's and requests result are described in HPE StoreOnce REST API
- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **params** (*dict*) – (optional) Dictionary with url encoded parameters.
- **timeout** (*float|tuple*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as `StoreOnceG4.timeout`.

Return type tuple(int, dict())

Returns Tuple with HTTP status code and dict with request result. For example: (200, {. . . }) or (204, None).

put (*url*, ***kwargs*)

Perform HTTP PUT request to HPE StoreOnce G4 disk backup device array. This method used to modify objects.

Parameters

- **url** (*str*) – URL address. Base part of url address is generated automatically and you should not care about it. Example of valid url: '/rest/alerts' or '/api/v1/management-services/licensing'. All available url's and requests result are described in HPE StoreOnce REST API
- **json** (*dict*) – (optional) A JSON serializable object to send in the body of request.
- **params** (*dict*) – (optional) Dictionary with url encoded parameters.

- **timeout** (*float | tuple*) – (optional) How many second to wait for the Rest server response before giving up. By default use same value as *StoreOnceG4.timeout*.

Return type tuple(int, dict())

Returns Tuple with HTTP status code and dict with request result. For example: (200, {. . . }) or (204, None).

property timeout

Variables ***timeout*** (*float | tuple*) – Number of seconds that Rest API client waits for response from HPE StoreOnce Gen 4 before timeout exception generation. You can use different timeouts for connection setup and for getting first piece of data. In this case, you should use tuple(float, float) with first value - connection timeout and the second value - read timeout. Or if you want to use same values for both type of timeouts, you can use one float value. ‘None’ value can be used instead to wait forever for a device response. Default value: (1, None).

7.2 Usage

7.2.1 Session management

Open Rest API session for StoreOnce Gen 4 disk backup device:

```
import hpestorapi

with hpestorapi.StoreOnceG4('10.0.0.1', 'Admin', 'password') as so:
    try:
        print('Something went wrong:')
        so.open()
    except Exception as error:
        print(error)
    else:
        # Perform requests to StoreOnce (get/post/put/delete)
        # ...
```

7.2.2 GET request

Simple GET request usage. This code print StoreOnce G4 groups:

```
import hpestorapi

with hpestorapi.StoreOnceG4('10.0.0.1', 'Admin', 'password') as so:
    try:
        so.open()
        status, data = so.get('/rest/groups')
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        if status == 200:
            for group in data['members']:
                print(group['groupName'])
```

7.2.3 POST request

This code will create new Catalyst client:

```
import hpestorapi

with hpestorapi.StoreOnceG4('10.0.0.1', 'Admin', 'password') as so:
    new_client = {'name': 'sqlserver', 'description': 'New host',
                  'password': 'secretpass'
    }
    try:
        so.open()
        status, data = so.post('/api/v1/data-services/cat/clients',
                               json=new_client)
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        if status == 201:
            print('Host succefully added.')
        else:
            print('Fail! Cannot add new catalyst client. Details:', data)
```

7.2.4 DELETE request

This code remove CIFS share:

```
import hpestorapi

with hpestorapi.StoreOnceG4('10.0.0.1', 'Admin', 'password') as so:
    share = 'CifsShare01'
    try:
        so.open()
        status, _ = so.delete('/api/v1/data-services/nas/shares'
                              '/share/{id}'.format(id=share))
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        if status == 204:
            print('Share succefully removed.')
        elif status == 404:
            print('Fail! Share does not exist.')
        else:
            print('Fail! Cannot remove share.')
```

7.2.5 PUT request

This code will update current SMTP configuration:

```
import hpestorapi

with hpestorapi.StoreOnceG4('10.0.0.1', 'Admin', 'password') as so:
    body = {'emailConfigurations': {
        'smtpPort': 25,
        'enabled': 'true',
        'senderEmailAddress': 'sender@company.com',
        'password': 'secretpassword',
        'smtpServer': 'email.company.com'
```

(continues on next page)

(continued from previous page)

```
    }
    }
    try:
        so.open()
        status, data = so.put('/rest/email', json=body)
    except Exception as error:
        print('Something went wrong:')
        print(error)
    else:
        print('SMTP configuration succefully updated.')
```

CHAPTER

EIGHT

DEBUG

8.1 Enable logging

hpestorapi uses Python's Standard Library logging module. Simple python script with enabled debug logging:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

    import logging
    import hpestorapi

    if __name__ == '__main__':
        # Setup logging format, level, logfile
        logfmt = ('[%(asctime)s] '
                  '%(levelname)-8s '
                  '%(filename)-12s:%(lineno)-3d '
                  '%(message)s')

        logging.basicConfig(format=logfmt,
                            level=logging.WARNING,
                            filename='messages.log')

        # Set XP, 3PAR, StoreOnce loglevel
        logging.getLogger("hpestorapi.xp").setLevel(logging.DEBUG)
        logging.getLogger("hpestorapi.storeserv").setLevel(logging.DEBUG)
        logging.getLogger("hpestorapi.storeonce").setLevel(logging.DEBUG)

        """
        Your code starts here
        """
```

Five logging levels are used:

- DEBUG
- INFO
- WARNING
- ERROR
- FATAL

8.2 Bug reports

If you have found a bug, please create an issue in GitHub. And do not hesitate to contact me: Ivan Smirnov <ivan.smirnov@hpe.com>

INDEX