PyU4V Documentation

Release 9.1.0.0

Dell

Dec 04, 2019

Contents

CHAPTER 1

Requirements

PyU4V Version	9.1.0.0
Unisphere Version	9.1.0.5
Array Model	VMAX-3, VMAX AFA, PowerMax
Array uCode	HyperMax OS, PowerMax OS
Platforms	Linux, Windows
Python	3.6, 3.7
Requirements	Requests, Six, urllib3
Test Requirements	TestTools, Tox

Note: If you want to continue to use Unisphere 8.4.x or 9.0.x with PyU4V you will need to remain on PyU4V 3.1.x. There is no support for PyU4V 9.1 with any version of Unisphere older than 9.1.x

Note: PyU4V officially supports Python 3.6 & 3.7, Python 2.x support has been dropped as it will soon be retired.

CHAPTER 2

PyU4V Version Compatibility

PyU4V version 9.1.x is compatible with scripts written for PyU4V versions >= 3.x, there is **zero** support or compatibility for PyU4V 2.x or earlier scripts in later versions of PyU4V. If you have scripts written which specifically target Unisphere REST 8.4 or 9.0 endpoints these are still accessible via PyU4V 9.1.x however you will need to ensure you are passing the version required when performing these calls as PyU4V 9.1 will default to using 9.1 endpoints exclusively. You will also need to pay special attention to any REST JSON payloads in custom scripts as payloads are subject to change between major Unisphere REST releases.

CHAPTER 3

Installation

PyU4V can be installed from source, via `pip`, or run directly from the source directory. To clone PyU4V and install from source use `git` and `pip`:

```
$ git clone https://github.com/MichaelMcAleer/PyU4V
$ cd PyU4V/
$ pip install .
```

Installing via `pip` without cloning from source can be achieved by specifying `PyU4V` as the install package for `pip`:

```
$ pip install PyU4V
# Install a specific version
$ pip install PyU4V==9.1.0.0
```

CHAPTER 4

Configuration

Once PyU4V is installed the next step is to configure it through `PyU4V.conf`. There is a sample version of `PyU4V.conf` provided in PyU4V which has default configuration options for logging but will require environment configuration setting changes for PyU4V to work.

Copy the sample `PyU4V.conf` provided with PyU4V to either your working directory or within a directory named `.PyU4V` in your current users home directory. The `.sample` suffix has to be removed for the configuration file to become valid for loading by PyU4V:

```
$ mkdir ~/.PyU4V
$ cp PyU4V/PyU4V.conf.sample ~/.PyU4V/PyU4V.conf
```

Note: The `\~` symbol is used here to represent the users home directory regardless of operating system, however, `\~` is not a valid shortcut in Windows command prompt so direct path to users home directory should be used instead `C:\> mkdir C:\Users\{user}\.PyU4V` where `{user}` is the current user.

Note: If `PyU4V.conf` is present in both the current working directory and the current user's home directory, the version of `PyU4V.conf` in the current working directory will take precedence. See the section below on PyU4V settings precedence.

Edit PyU4V configuration settings in `PyU4V.conf` under the `[setup]` heading, these setting will need to reflect your environment configuration:

```
[setup]
username=pyu4v-user
password=secret-pass
server_ip=10.0.0.75
port=8443
array=00012345678
verify=/path-to-file/server_hostname.pem
```

Where...

Key	Description
username	Unisphere REST login username
password	Unisphere REST login password
server_ip	Unisphere server IP address
port	Unisphere server port number
array	12 digit array serial number
verify	True - Load SSL cert from CA certificate bundle /path/to/file - Load SSL cert from file location False - Disable SSL verification

4.1 Alternative PyU4V.conf Load Using U4VConn().file_path

It is also possible to override PyU4V.conf in both the working directory and home directory by specifying U4VConn.file_path='/path/to/PyU4V.conf' before initialising PyU4V.

```
import PyU4V

PyU4V.U4VConn.file_path = '~/path/to/PyU4V.conf'
# Instantiate U4VConn() using the PyU4V config file specified in file_path
conn = U4VConn()
```

If you specify a file_path whilst having a copy of PyU4V.conf in both your working directory and home directory in ~/.PyU4V, the instance of PyU4V.conf as specified in file_path will take precedence. See the section below on PyU4V settings precedence.

4.2 Passing Environment Configuration to U4VConn() on Initialisation

Instead of specifying PyU4V configuration options within PyU4V.conf it is possible to pass these values directly to U4VConn() on initialisation. The key/values expected are the same as those specified in PyU4V.conf.

```
>>> import PyU4V
>>> conn = U4VConn(
        username='pyu4v-user', password='secret-pass',
        server_ip='10.0.0.75', port='8443', verify=True,
        array_id='00012345678')
>>> conn.common.get_unisphere_version()
{'version': 'V9.1.0.5'}
```

If you pass configuration into U4VConn() directly in the code, these settings will override any that are defined in PyU4V.conf at any location.

4.3 PyU4V Configuration Loading Precedence

There are a number of ways to initialise PyU4V with your environment settings through PyU4V.conf or passing the values directly. These various methods of setting PyU4V environment configuration have a load precedence, these are

listed in order with number 1 being the first load precedent:

1. Configuration key/values passed directly to `U4VConn()`
2. `PyU4V.conf` as specified in `U4VConn.file_path`
3. `PyU4V.conf` in current working directory
4. `PyU4V.conf` in current users home directory
5. If none of the above or missing mandatory options raise `MissingConfigurationException`

4.4 PyU4V Logger Configuration

Logger options in PyU4V have been streamlined since the previous 3.1.x version, all options are now consolidated to save on duplicate options being presented. All logger configuration options in PyU4V.conf can be found under the comment `; log configuration` in the sections `[loggers*]`, `[handlers*]`, and `[formatters*]`. There are a number of configuration options which you can change to suit your needs, the most relevant of those for the installation and configuration process are outlined in the table below.

Section	Config Option	Description
`[logger_PyU4V]`	`level=INFO`	Sets the PyU4V log level, this defaults to INFO but can be changed to any logger LOG level
`[handler_consoleHandler]`	`args=(sys.stdout,)`	Control how log messages are output to console
`[handler_fileHandler]`	`args=('PyU4V.log', 'a', 10485760, 10)`	Control how log messages are written to log files and where the log file is located
`[formatter_simpleFormatter]`	`format=%(asctime)s - %(name)s - %(levelname)s - %(message)s`	Set the format for the log prefix output in PyU4V.log

Note: PyU4V log functionality is run on top of Python's great inbuilt logger. If you require in depth descriptions of the PyU4V logger configuration options, the logger sections, or input arguements for the handlers, please see the official Python Logger documentation here.

CHAPTER 5

Quick Start Guide

5.1 Initialise PyU4V

First, make sure that PyU4V is installed as directed in the *Installation Guide*:

```
$ pip show PyU4V
```

To initialise the connection with Unisphere use `U4VConn`, it will load the configuration provided in `PyU4V.conf` configured during installation:

```
import PyU4V
conn = PyU4V.U4VConn()
```

With a connection to the Unisphere server you can start to run some test queries to validate the successful connection.

```
conn.common.get_unisphere_version()
('V9.1.0.5', '91')
conn.common.get_array_list()
["000197900123", "000197900124", "000197900125", "000197900126"]
```

If you want to query another array, you can set the `array_id` to a value of your choice which overrides what is set in `PyU4V.conf`. Alternatively you can initialise a new PyU4V connection and pass in the `array_id` but load all other configuration settings from `PyU4V.conf`.

```
# Option 1 - Set new array ID in current PyU4V connection
conn.set_array_id('000197900126')
# Option 2 - Create new PyU4V connection using PyU4V.conf settings
new_conn = PyU4V.U4VConn(array_id='000197900126')
```

5.2 PyU4V Unisphere REST Coverage

The functions in PyU4V have been divided into logical categories which reflect the various categories provided by the Unisphere REST API:

- Common - REST methods and assistive utilities
- Migration - all migration related calls
- Performance - all performance and threshold calls
- Provisioning - all provisioning and masking related calls
- Replication - all local and remote replication calls
- System - all system level calls
- Workload Planner - all workload planner calls
- Utils - assistive functions to aid with PyU4V usage

There are plans to further increase the coverage of Unisphere REST calls in version 9.1. All changes are reflected in the PyU4V change log (link).

5.3 Perform a Custom REST Call in PyU4V

At its core PyU4V is your typical REST client. It creates a request header, defines any request parameters in JSON, sends the request with an associated method which dictates the action being performed (`GET`, `POST`, etc.), and expects a response payload in most instances (`DELETE` calls tend to return a status instead of a response payload).

There are functions created in `PyU4V.common` which provide the ability to perform `GET`, `POST` and other request calls directly with the Unisphere REST API to any supported endpoint.

- `PyU4V.common.get_resource()` - `GET`
- `PyU4V.common.create_resource()` - `POST`
- `PyU4V.common.modify_resource()` - `PUT`
- `PyU4V.common.delete_resource()` - `DELETE`

If there is any functionality that is provided by the Unisphere REST API that is not yet implemented in PyU4V, it is possible to create a custom function which use the above functions to make use of that functionality. For information on the Unisphere REST API please its related documentation.

To find out more information on the any PyU4V calls refer to the supporting function documentation in the *API Glossary* , there are also programmers guide examples provided with this documentation which demonstrate a range of functions using PyU4V.

CHAPTER 6

PyU4V Usage Recommendations

In this section we aim to highlight various topics that we hope will improve your overall experience when using PyU4V. If you have any recommendations that you would like to make please let us know by opening an issue and we will review it for addition here.

6.1 Volume Naming Conventions

It is strongly recommended that you create a volume with a unique volume_name or volume_identifier. When you search for a volume device_id based on it's volume_name, it is preferable to receive a single device id rather than a list of device ids, of which any could be the device that you just created.

6.2 Performance Monitoring

When using PyU4V for performance metrics collection there are a number of best practices that you should follow:

- After enabling Unisphere for performance metrics collection allow Unisphere 30 minutes to gather enough data before making any calls.
- The most granular time available with PyU4V performance metrics collection is 5 minutes, so querying for data more frequently than 5 minutes is a wasteful use of resources.
- If you want to ensure that your performance metric collection is querying the most recent performance data available set a recency window value on your calls.
- If the performance timestamp is not recent as of 5-10 minutes ago there is a strong likelihood that your instance of Unisphere has gone into catchup mode and is processing a backlog of performance data. It will resume normal operations once this backlog processing is complete and be indicated by performance timestamps with a window of 5-10 minutes from the current time.
- When querying a single instance of Unisphere for performance metrics across a number of arrays be careful on the load placed on Unisphere and try to determine if it is possible to query that amount of data in the given time frame you have set.

- If querying for data at regular intervals, examine your calls to see if you can create the interval time. If information is not likely to change over the course of 24 hours then querying once a day would be sufficient.

Lastly, and most importantly, with great power comes great responsibility, PyU4V provides you with the ability to query every performance metric for every performance category. Instead of gathering everything possible, be resourceful with your calls and only query what is needed. This will provide improvements in PyU4V performance, network load, and Unisphere REST performance. If you are only interested in querying for KPIs, you can specify that only KPI metrics are returned, but better still only query for a subset of metrics that you are interested in.

CHAPTER 7

Contribute to PyU4V

Please do! Create a fork of the project into your own repository. Make all your necessary changes and create a pull request with a description on what was added or removed and details explaining the changes in lines of code.

The following tests must all run cleanly:

```
$ cd PyU4V
$ tox -e py36
$ tox -e py37
$ tox -e pep8
$ tox -e pylint
```

Note: If you do not have all the versions of Python installed, just run tox on the versions you have.

Once the above tests all run clean and CI tests are run to ensure there is no impact on existing functionality, PyU4V core reviewers will review the code to ensure it conforms to all conventions outlined in the section below. If all looks good we will merge it to the PyU4V master branch.

7.1 Conventions

For neatness and readability we will enforce the following conventions going forward on all code in PyU4V.

1. Single quotes `'` unless double quotes `"` necessary.
2. Use `.format()` for string concatenation.

3. Use the following format for doc strings, the return description uses `:returns:` instead of the docstring default `:return:`. Pep8 will guide you with all the docstring conventions.

```
def my_test_func(input_1, input_2):
    """The is my summary of the method with full stop.

```

(continues on next page)

(continued from previous page)

```
    This is a brief description of what the method does.  Keep
    it as simple as possible.

    :param input_1: brief description of input parameter 1, if it goes over
                    one line it must be indented with the start of the
                    previous lines -- str
    :param input_2: brief description of input parameter 2, you must also
                    provide the input parameter type after the description
                    after double dash -- int
    :returns: what gets returned from method, omit if none, type must also
              be specified, in this case it is a boolean -- bool
    :raises: Exceptions raised, omit if none
    """
    return True if input_1 or input_2 else raise Exception
```

5. Class names are mixed case with no underscores _.

```
class ClassFunctions(object):
    """Collection of functions ClassFunctions."""
```

6. Public Methods are separated by underscores _. Make the name as meaningful as possible.

```
def public_function_does_exactly_what_it_says_it_does(self):
    """Function does exactly what it says on the tin."""
```

7. Private Methods are prefixed and separated by underscores _. Make the name as meaningful as possible.

```
def _private_function_does_exactly_what_it_says_it_does(self):
    """Function does exactly what it says on the tin."""
```

8. If functions seems to big or too complicated then consider breaking them into smaller functions.

9. If a line of code must extend over more than one line, use parenthesis `()` around the code instead of `\` at the end of the line.

```
my_multi_line_string = ('This is an example of a string '
                        'that extends over more than one line.')

my_multi_line_function = (
    this_is_a_very_long_function_call_that_cannot_meet_79_char_limit())
```

10. Each new function must be unit tested.

11. Each bug fix must be unit tested.

12. Unix and OS X format only. If in doubt run

```
$ sudo apt-get install dos2unix
$ dos2unix myfile.txt
```

or in PyCharm:

CHAPTER 8

Issues & Support

We greatly value your feedback! Please file bugs and issues on the Github issues page for this project.

We aim to track and document everything related to this repo via the issues page. The code and documentation are released with no warranties or SLAs and are intended to be supported through a community driven process.

When opening an issue please include the following information to help us debug:

- PyU4V version
- Unisphere version
- Description of the problem
- Impacted functions
- PyU4V logs showing issue

CHAPTER 9

Programmers Guide

In this section a number of examples demonstrating various PyU4V functionality are provided. If you have an example which you believe would make a good addition here let us know be opening a support issue and we will review it for addition!

Although the scope of these programmers examples is limited, it is worth pointing out that if you want to see a working example of any function you can do so by looking at that function's associated continuous integration test.

Note: In the first two examples print statements will be included to show how responses from REST requests for newly created assets can be used for further operations, in all later examples print statements will not be included for succinctness.

9.1 First connection to PyU4V

In this basic example we initialise a connection with Unisphere, retrieving version and array information, and finishing by closing the REST session.

```
"""examples/unisphere_connect.py."""

import PyU4V

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn(
    u4v_version='90', server_ip='10.0.0.75', port=8443,
    verify='~/.PyU4V/Unisphere91.pem', username='pyu4v-user',
    password='secret-pass')

# Get the Unisphere version
version = conn.common.get_uni_version()

# Retrieve a list of arrays managed by your instance of Unisphere
```

(continues on next page)

(continued from previous page)

```
array_list = conn.common.get_array_list()

# Output results to screen
print('Congratulations on your first connection to Unisphere, your '
      'version is: {ver}'.format(ver=version[0]))
print('This instance of Unisphere instance manages the following arrays: '
      '{arr_list}'.format(arr_list=array_list))

# GET those arrays which are local to this instance of Unisphere
local_array_list = list()
for array_id in array_list:
    array_details = conn.common.get_array(array_id)
    if array_details['local']:
        local_array_list.append(array_id)

# Output results to screen
print('The following arrays are local to this Unisphere instance: '
      '{arr_list}'.format(arr_list=local_array_list))

# Close the session
conn.close_session()
```

9.2 A-Synchronous Provisioning & Creating Storage for a Host

This example demonstrates checking an array SRP and service level to determine if there is enough headroom to provision storage of a set size, if so, proceed to creating a storage group with volume. Create a host, port group, and masking view to tie all the elements together, close the session when done.

```
"""examples/async_provision.py"""

import PyU4V

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn(
    u4v_version='90', server_ip='10.0.0.75', port=8443,
    verify='~/.PyU4V/Unisphere91.pem', username='pyu4v-user',
    password='secret-pass')

# Before provisioning storage we are going to check that there is enough
# headroom left on the array for our provisioning operations
REQUESTED_CAPACITY = 10

# Get the available headroom for the SRP and service level required
headroom_info = conn.wlp.get_headroom(array_id=conn.array_id,
                                      srp='SRP_1', slo='Diamond')

# Extract the capacity value from the headroom_info REST response
headroom_capacity = headroom_info[
    'OLTP'][0]['headroom'][0]['headroomCapacity']

# If the requested capacity of 10GB is less than or equal to the available
# capacity proceed with the provisioning operations
if REQUESTED_CAPACITY <= int(headroom_capacity):
    # Create a non-empty storage group using asynchronous request - we can
```

(continues on next page)

(continued from previous page)

```
    # wait until the job completes or proceed with operations and check
    # back at a later time
    storage_group_async_job = (
        conn.provisioning.create_non_empty_storage_group(
            srp_id='SRP_1', storage_group_id='example-sg',
            service_level='Diamond', num_vols=1,
            vol_size=REQUESTED_CAPACITY, cap_unit='GB', _async=True))

    # We will wait this time on the results of the storage group create
    # request
    conn.common.wait_for_job(operation='Create SG with volume',
                             job=storage_group_async_job)
    print('Storage Group created successfully...')

    # Get information on our new storage group
    storage_group_info = conn.provisioning.get_storage_group(
        storage_group_name='Example-SG')
    print('Storage Group details: {details}'.format(
            details=storage_group_info))

    # Create a Host using supplied initiator IDs, these can be also be
    # retrieved via the call conn.provisioning.get_available_initiator()
    initiator_list = ['iqn:2019-test1', 'iqn:2019-test1']
    host_info = conn.provisioning.create_host(
        host_name='Example-Host', initiator_list=initiator_list)
    print('Host created successfully...')
    print('New Host details: {details}'.format(details=host_info))

    # Create a Port Group using supplied ports, these could be also be
    # retrieved via the call conn.provisioning.get_port_list()
    port_group_info = conn.provisioning.create_port_group(
        port_group_id='Example-PG', director_id='SE-01', port_id='1')
    print('Port Group created successfully...')
    print('Port Group details: {details}'.format(details=port_group_info))

    # Create a Masking View and tie all the elements we have created
    # together
    masking_view_info = (
        conn.provisioning.create_masking_view_existing_components(
            port_group_name='Example-PG', masking_view_name='Example-MV',
            storage_group_name='Example-SG', host_name='Example-Host'))
    print('Masking View created...')
    print('Masking View details: {details}'.format(
        details=masking_view_info))

# Close the session
conn.close_session()
```

9.3 Local Replication with SnapVX

In this example a new storage group is created with a single 1GB volume. A snapshot name is generated using the current time so it can be easily identified, and the storage group snapshot is created. The operation is verified by querying for a list of snapshots for a given storage group and confirming the snapshot we created is present in that list.

```
"""examples/create_snapshot.py"""

import PyU4V
import time

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn()

# Create storage Group with one volume using settings specified for
# service level and capacity
storage_group = conn.provisioning.create_non_empty_storage_group(
    srp_id='SRP_1', storage_group_id='PyU4V_SG', service_level='Diamond',
    workload=None, num_vols=1, vol_size=1, cap_unit='GB')

# Define a Name for the Snapshot, in this case the name auto appends
# the host
# time for when it was taken for ease of identification
snap_name = 'PyU4V_Snap_' + time.strftime('%d%m%Y%H%M%S')

# Create the snapshot of the storage group containing the volume and
# storage group created in the previous step
snapshot = conn.replication.create_storage_group_snapshot(
    storage_group_id=storage_group['storageGroupId'], snap_name=snap_name)

# Confirm the snapshot was created successfully, get a list of storage
# group snapshots
snap_list = conn.replication.get_storage_group_snapshot_list(
    storage_group_id=storage_group['storageGroupId'])

# Assert the snapshot name is in the list of storage group snapshots
assert snapshot['name'] in snap_list

# Close the session
conn.close_session()
```

This example will create a storage group with a volume, create a snapshot of that storage group and link the snapshot to a new storage group. This is a typical workflow for provisioning a dev environment and making a copy available.

```
"""examples/link_snapshot.py"""

import PyU4V
from time import strftime

# Set up connection to Unisphere for PowerMax Server, details collected
# from configuration file in working directory where script is stored.
conn = PyU4V.U4VConn()

# Create storage Group with one volume
storage_group = conn.provisioning.create_non_empty_storage_group(
    srp_id='SRP_1', storage_group_id='PyU4V_SG', service_level='Diamond',
    workload=None, num_vols=1, vol_size=1, cap_unit='GB')

# Define a Name for the Snapshot, in this case the name auto appends the
# host time for when it was taken for ease of identification
snap_name = 'PyU4V_Snap_' + time.strftime('%d%m%Y%H%M%S')

# Create the snapshot of the storage group containing the volume and
```

(continues on next page)

(continued from previous page)

```
# storage group created in the previous step
snapshot = conn.replication.create_storage_group_snapshot(
    storage_group_id=storage_group['storageGroupId'], snap_name=snap_name)

# Link The Snapshot to a new storage group, the API will automatically
# create the link storage group with the right number of volumes if one
# with that name doesn't already exist
conn.replication.modify_storage_group_snapshot(
    src_storage_grp_id=storage_group['storageGroupId'],
    tgt_storage_grp_id='PyU4V_LNK_SG', link=True,
    snap_name=snap_name, gen_num=0)

# Close the session
conn.close_session()
```

9.4 Remote Replication with SRDF

This example will create a storage group on the PowerMax array with some volumes. Once the storage group has been created it will protect the volumes in the storage group to a remote array using SRDF/Metro, providing Active/Active business continuity via Symmetrix Remote Data Facility (SRDF).

```
"""srdf_example.py"""

import PyU4V

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn()

# Create storage Group with one volume using settings specified for
# service level and capacity
storage_group = conn.provisioning.create_non_empty_storage_group(
    srp_id='SRP_1', storage_group_id='PyU4V_SG', service_level='Diamond',
    workload=None, num_vols=1, vol_size=1, cap_unit='GB')

# Make a call to setup the remote replication, this will automatically
# create a storage group with the same name on the remote array with the
# correct volume count and size, the example here is executed
# asynchronously and a wait is added to poll the async job id until
# complete
srdf_job_id = conn.replication.create_storage_group_srdf_pairings(
    storage_group_id=storage_group['storageGroupId'],
    remote_sid=conn.remote_array, srdf_mode="Active", _async=True)

# Wait until the previous create SRDF pairing job has completed before
# proceeding
conn.common.wait_for_job_complete(job=srdf_job_id)

# The now protected storage group will have an RDFG associated with it,
# using the function conn.replication.get_storage_group_rdfg() function we
# can retrieve a list of RDFGs associated with the storage group, in this
# case there will only be one
rdfg_list = conn.replication.get_storage_group_srdf_group_list(
    storage_group_id=storage_group['storageGroupId'])
```

(continues on next page)

(continued from previous page)

```
# Extract the (only) RDFG number from the retrieved list
rdfg_number = rdfg_list[0]

# Finally the details of the protected storage group can be output to the
# user.
storage_group_srdf_info = conn.replication.get_storage_group_srdf_details(
    storage_group_id=storage_group['storageGroupId'],
    rdfg_num=rdfg_number)

# Close the session
conn.close_session()
```

9.5 Performance Metrics Gathering

This example demonstrates a range of performance functionality such as getting performance categories and metrics, timestamps from Unisphere for an array, get recent only performance information, and getting ResponseTime for all SRPs in an array.

```
"""examples/performance_data_retrieval.py"""

from PyU4V import U4VConn

# Initialise PyU4V Unisphere connection
conn = PyU4V.U4VConn(
    u4v_version='90', server_ip='10.0.0.75', port=8443,
    verify='~/.PyU4V/Unisphere91.pem', username='pyu4v-user',
    password='secret-pass')

# Get a list of performance categories
category_list = conn.performance.get_performance_categories_list()

# Get a list of supported metrics for the category 'FEDirector'
fe_dir_metrics = conn.performance.get_performance_metrics_list(
    category='FEDirector')

# Get a list of KPI only metrics for the category 'StorageGroup'
storage_group_metrics = conn.performance.get_performance_metrics_list(
    category='StorageGroup', kpi_only=True)

# Get array KPI performance metrics for the most recent timestamp only,
# set recency so timestamp has to be less than 5 minutes old
array_performance_data = conn.performance.get_array_stats(metrics='KPI',
                                                          recency=5)

# Get ResponseTime for each SRP for the last 4 hours
# Firstly get the most recent performance timestamp for your array
recent_timestamp = conn.performance.get_last_available_timestamp()
# Set the performance recency value to 10 minutes and check if the most
# recent timestamp meets that recency value
conn.performance.recency = 10
is_recent_ten = conn.performance.is_timestamp_current(recent_timestamp)
# Recency can also be passed to is_timestamp_current
is_recent_five = conn.performance.is_timestamp_current(recent_timestamp,
                                                       minutes=5)
```

(continues on next page)

(continued from previous page)

```
# Get the start and end times by providing the most recent timestamp and
# specifying a 4 hour difference
start_time, end_time = conn.performance.get_timestamp_by_hour(
    end_time=recent_timestamp, hours_difference=4)
# Get the list of SRPs
srp_keys = conn.performance.get_storage_resource_pool_keys()
srp_list = list()
for key in srp_keys:
    srp_list.append(key.get('srpId'))
# Get the performance data for each of the SRPs in the list
for srp in srp_list:
    srp_data = conn.performance.get_storage_resource_pool_stats(
        srp_id=srp, metrics='ResponseTime', start_time=start_time,
        end_time=end_time)

# Close the session
conn.close_session()
```

9.6 System

This example of system calls demonstrates performing a system health check, retrieving information from the last health check, querying for all installed disk IDs in an array and outputting information about each.

```
"""examples/system_health_check.py."""

import PyU4V

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn(
        u4v_version='90', server_ip='10.0.0.75', port=8443,
        verify='~/.PyU4V/Unisphere91.pem', username='pyu4v-user',
        password='secret-pass')

# Perform a system health check, this call can take 15-20 minutes to
# complete in Unisphere due to the nature of the checks performed
conn.system.perform_health_check(description='test-hc-dec19')

# Get details of the last system health check
health_check = conn.system.get_system_health()

# Get a list of physical disks installed in the array
disk_list = conn.system.get_disk_id_list()

# Get disk information for each disk installed
for disk in disk_list.get('disk_ids'):
    disk_info = conn.system.get_disk_details(disk_id=disk)

# Close the session
conn.close_session()
```

9.7 File Handling & Thresholds

In this example both performance threshold calls and CSV file handling with PyU4V are demonstrated. A call is made to retrieve a full list of performance threshold settings and output the results to a CSV file at a path specified by the user. That CSV file is read into a Python dictionary and the respective values within are updated. Once complete the updated threshold settings are uploaded to Unisphere to take immediate effect.

```
"""examples/thresholds.py."""

import os
import PyU4V

# Initialise PyU4V connection to Unisphere
conn = PyU4V.U4VConn()

# Set the CSV file name and path
current_directory = os.getcwd()
output_csv_name = 'thresholds-test.csv'
output_csv_path = os.path.join(current_directory, output_csv_name)

# Generate a CSV file with all of the thresholds and corresponding values
conn.performance.generate_threshold_settings_csv(
    output_csv_path=output_csv_path)

# Read the CSV values into a dictionary, cast all string booleans and
# numbers to their proper types
threshold_dict = PyU4V.utils.file_handler.read_csv_values(output_csv_path,
                                                          convert=True)

# Increase all of the first threshold values by 5 and second threshold
# values by 10, alert only on the KPIs
for i in range(0, len(threshold_dict.get('metric'))):
    threshold_dict['firstThreshold'][i] += 5
    threshold_dict['secondThreshold'][i] += 5
    if threshold_dict['kpi'][i] is True:
        threshold_dict['alertError'][i] = True

# Process the CSV file and update the thresholds with their corresponding
# values, we are only going to set the threshold value if it is a KPI
conn.performance.set_thresholds_from_csv(csv_file_path=output_csv_path,
                                         kpi_only=True)

# It is also possible to set a threshold value without editing the values
# in a CSV, the threshold metric and be edited directly
threshold_settings = conn.performance.update_threshold_settings(
    category='Array', metric='PercentCacheWP', alert=True,
    first_threshold=60, first_threshold_occurrences=2,
    first_threshold_samples=6, first_threshold_severity='INFORMATION',
    second_threshold=90, second_threshold_occurrences=1,
    second_threshold_samples=3, second_threshold_severity='CRITICAL')

# Close the session
conn.close_session()
```

CHAPTER 10

API Glossary

10.1 PyU4V API

10.1.1 PyU4V.univmax_conn

Creates the connection with the Unisphere for PowerMax instance.

univmax_conn.py.

class PyU4V.univmax_conn.**U4VConn**(*username=None*, *password=None*, *server_ip=None*, *port=None*, *verify=None*, *u4v_version='91'*, *interval=5*, *retries=200*, *array_id=None*, *application_type=None*, *remote_array=None*)

Bases: object

U4VConn.

close_session()
Close the current rest session.

set_array_id(*array_id*)
Set the array serial number.

Parameters **array_id** – the array serial number – str

set_requests_timeout(*timeout_value*)
Set the requests timeout.

Parameters **timeout_value** – the new timeout value – int

validate_unisphere()
Check that the minimum version of Unisphere is in-use.

If the version of Unisphere used does not meet minimum requirements the application will exit gracefully.

Raises SystemExit

10.1.2 PyU4V.common

common.py.

class PyU4V.common.**CommonFunctions**(*rest_client*)

Bases: object

CommonFunctions.

static check_ipv4(*ipv4*)

Check if a given string is a valid ipv6 address

Parameters **ipv4** – ipv4 address – str

Returns string is valid ipv4 address – bool

static check_ipv6(*ipv6*)

Check if a given string is a valid ipv6 address

Parameters **ipv6** – ipv6 address – str

Returns string is valid ipv6 address – bool

static check_status_code_success(*operation*, *status_code*, *message*)

Check if a status code indicates success.

Parameters

- **operation** – operation being performed – str
- **status_code** – status code – int
- **message** – server response – str

Raises VolumeBackendAPIException

static convert_to_snake_case(*camel_case_string*)

Convert a string from camel case to snake case.

Parameters **camel_case_string** – string for formatting – str

Returns snake case variant – str

static create_list_from_file(**args*, ***kwargs*)

create_resource(**args*, ***kwargs*)

Create a resource.

Parameters **kwargs** – param version: Unisphere version – int param no_version: if versionless uri – bool param category: resource category e.g. sloprovisioning – str param resource_level: resource level e.g. storagegroup – str param resource_level_id: resource level id – str param resource_type: optional name of resource – str param resource_type_id: optional name of resource – str param resource: optional name of resource – str param resource_id: optional name of resource – str param object_type: optional name of resource – str param object_type_id: optional name of resource – str param payload: query parameters – dict

Returns resource object – dict

delete_resource(**args*, ***kwargs*)

Delete a resource.

Parameters **kwargs** – param version: Unisphere version – int param no_version: if versionless uri – bool param category: resource category e.g. sloprovisioning – str param resource_level: resource level e.g. storagegroup – str param resource_level_id: resource level id – str param resource_type: optional name of resource – str param resource_type_id: optional name of

resource – str param resource: optional name of resource – str param resource_id: optional name of resource – str param object_type: optional name of resource – str param object_type_id: optional name of resource – str param payload: query parameters

get_array(*array_id*)
Get array details.

Parameters **array_id** – array id – str

Returns array details – dict

get_array_list(*filters=None*)
Return a list of arrays.

Parameters **filters** – optional filters – dict

Returns arrays ids – list

get_headroom(***kwargs*)

get_iterator_page_list(*iterator_id*, *start*, *end*)
Get a page of results from an iterator instance.

Parameters

- **iterator_id** – iterator id – str
- **start** – the start number – int
- **end** – the end number – int

Returns iterator page results – dict

get_iterator_results(*rest_response*)
Get all results from all pages of an iterator if count > 1000.

Parameters **rest_response** – response JSON from REST API – dict

Returns all results – dict

get_job_by_id(*job_id*)
Get details of a specific job.

Parameters **job_id** – job id – str

Returns job details – dict

get_request(*target_uri*, *resource_type*, *params=None*)
Send a GET request to the array.

Parameters

- **target_uri** – target uri – str
- **resource_type** – the resource type, e.g. maskingview – str
- **params** – optional filter params – dict

Returns resource_object – dict

Raises ResourceNotFoundException

get_resource(**args*, ***kwargs*)
Get resource details from the array.

Parameters **kwargs** – param version: Unisphere version – int param no_version: if versionless uri – bool param category: resource category e.g. sloprovisioning – str param resource_level: resource level e.g. storagegroup – str param resource_level_id: resource level id – str param

resource_type: optional name of resource – str param resource_type_id: optional name of resource – str param resource: optional name of resource – str param resource_id: optional name of resource – str param object_type: optional name of resource – str param object_type_id: optional name of resource – str param params: query parameters – dict

Returns resource object – dict

get_uni_version()
Get the unisphere version from the server.

Returns version and major_version e.g. "V9.1.0.2", "91" – str, str

get_v3_or_newer_array_list(*filters=None*)
Return a list of V3 or newer arrays in the environment.

Parameters **filters** – optional filters – dict

Returns arrays ids – list

get_wlp_information(***kwargs*)

modify_resource(**args*, ***kwargs*)
Modify a resource.

Parameters **kwargs** – param version: Unisphere version – int param no_version: if versionless uri – bool param category: resource category e.g. sloprovisioning – str param resource_level: resource level e.g. storagegroup – str param resource_level_id: resource level id – str param resource_type: optional name of resource – str param resource_type_id: optional name of resource – str param resource: optional name of resource – str param resource_id: optional name of resource – str param object_type: optional name of resource – str param object_type_id: optional name of resource – str param payload: query parameters

Returns resource object – dict

static read_csv_values(**args*, ***kwargs*)

wait_for_job(*operation*, *status_code*, *job*)
Check if call is async, wait for it to complete.

Parameters

- **operation** – operation being performed – str
- **status_code** – status code – int
- **job** – job id – str

Returns task details – list

Raises VolumeBackendAPIException

wait_for_job_complete(*job*)
Given the job wait for it to complete.

Parameters **job** – job details – dict

Returns response code, result, status, task details – int, str, str, list

Raises VolumeBackendAPIException

10.1.3 PyU4V.migration

migration.py.

class PyU4V.migration.**MigrationFunctions**(*array_id*, *rest_client*)
Bases: object

MigrationFunctions.

create_migration_environment(*target_array_id*)
Create a new migration environment between two arrays.

Creates a new migration environment between two arrays for use with non disruptive migrations

Parameters **target_array_id** – target array id – str

Returns migration environment info – dict

create_storage_group_migration(*storage_group_name*, *target_array_id*, *srp_id=None*, *port_group_id=None*, *no_compression=False*, *pre_copy=False*, *validate=False*)
Create a migration session for a storage group.

Parameters

- **storage_group_name** – storage group id – str
- **target_array_id** – target array id – str
- **srp_id** – storage resource pool id – str
- **port_group_id** – port group id – str
- **no_compression** – dont use compression – bool
- **pre_copy** – use pre copy – bool
- **validate** – validate – bool

Returns new storage group – dict

delete_migration_environment(*target_array_id*)
Delete migration environment.

Given a target array will delete migration environment, used once all migrations are complete

Parameters **target_array_id** – target array id – str

delete_storage_group_migration(*storage_group_name*)
Given a name, delete the storage group migration session.

Parameters **storage_group_name** – storage group id – str

get_array_migration_capabilities()
Check what migration facilities are available.

Returns array capabilities – dict

get_environment(*target_array_id*)
Given a name, return migration environment details.

Parameters **target_array_id** – target array id – str

Returns environment details – dict

get_environment_list()
Get list of all migration environments.

Returns environments – list

get_migration_info()
Return migration information for an array.

Returns migration info – dict

get_storage_group(*storage_group_name*)
Given a name, return storage group migrations details.

Parameters **storage_group_name** – storage group id – str

Returns storage group details – dict

get_storage_group_list(*include_migrations=False*)
Get list of all storage groups or migrating storage groups.

Parameters **include_migrations** – return only SGs with migration sessions – bool

Returns storage groups or migrating storage groups – list

get_storage_groups()
Get all storage groups and migrating storage groups.

Returns storage groups and migrating storage groups – dict

modify_storage_group_migration(*storage_group_name*, *action*, *options=None*, *_async=False*)
Modify the state of a storage group's migration session.

Valid migrations options are 'Cutover', 'Sync', 'Commit', 'Recover', and 'ReadyTgt'.

Parameters

- **storage_group_name** – storage group id – str
- **action** – migration action – str
- **options** – migration options, example: {cutover': {'force': True}} – dict
- **_async** – if call should be async – bool

Returns modified storage group info – dict

10.1.4 PyU4V.performance

performance.py.

class PyU4V.performance.**PerformanceFunctions**(*array_id*, *rest_client*)
Bases: object

PerformanceFunctions.

extract_timestamp_keys(*array_id=None*, *category=None*, *director_id=None*, *key_tgt_id=None*)
Retrieve the timestamp keys for a given performance asset.

Note: If a director key timestamp is required, set this as the key_tgt_id, the input parameter director_id is only required for port key extraction.

Parameters

- **array_id** – array id – str
- **category** – performance category – str
- **director_id** – director id – str
- **key_tgt_id** – object id for the timestamp required – str

Returns timestamp in milliseconds since epoch – str

static format_metrics(*metrics*)
Format metrics input for inclusion in REST request.

Take metric parameters and format them correctly to be used in REST request body. Valid input types are string and list.

Parameters **metrics** – metric(s) – str or list

Returns metrics – list

Raises InvalidInputException

format_time_input(*array_id=None*, *category=None*, *director_id=None*, *key_tgt_id=None*, *start_time=None*, *end_time=None*)
Format time range for use in the request object.

Use cases are: 1. If start is set but not end, set end to most recent timestamp 2. If end is set but not start, set start time to first available 3. If neither are set, use most recent timestamp 4. If both are set, skip if conditions and check input is valid

Note: If a director key timestamp is required, set this as the key_tgt_id, the input parameter director_id is only required for port key extraction. A category is only required when timestamp extraction is required.

Parameters

- **array_id** – array id – str
- **category** – performance category – str
- **director_id** – director id (for port key extraction only) – str
- **key_tgt_id** – object id for the timestamp required – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns start time, end time (tuple) – str, str

Raises InvalidInputException

generate_threshold_settings_csv(*output_csv_path*)
Generate a csv file with threshold settings.

Creates a CSV file with current alert configuration for the given unisphere instance category, metric, first_threshold, second_threshold, alert_user, kpi.

Parameters **output_csv_path** – filename for CSV to be generated – str

get_all_fe_director_metrics(***kwargs*)

get_array_keys()
List Arrays registered for performance data collection.

Returns Arrays with first and last available dates – list

get_array_metrics(***kwargs*)

get_array_stats(*metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List performance data for specified array for giving time range.

Parameters

- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str

- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_backend_director_keys(*array_id=None*)
List BE directors for the given array.

Parameters **array_id** – array id – str

Returns BE directors with first and last available dates – list

get_backend_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given BE director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_backend_emulation_keys(*array_id=None*)
List BE emulations for the given array.

Parameters **array_id** – array id – str

Returns BE emulation info with first and last available dates – list

get_backend_emulation_stats(*emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given BE emulation.

Parameters

- **emulation_id** – emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_backend_port_keys(*director_id*, *array_id=None*)
List BE ports for the given array.

Parameters

- **director_id** – array id – str
- **array_id** – director id – str

Returns BE port info with first and last available dates – list

get_backend_port_stats(*director_id*, *port_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given BE port.

Parameters

- **director_id** – director id – str
- **port_id** – port id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_board_keys(*array_id=None*)
List boards for the given array.

Parameters **array_id** – array id – str

Returns board info with first and last available dates – list

get_board_stats(*board_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given board.

Parameters

- **board_id** – board id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_cache_partition_keys(*array_id=None*)
List cache partitions for the given array.

Parameters **array_id** – array id – str

Returns cache partition info with first and last available dates – list

get_cache_partition_perf_stats(*cache_partition_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given cache partition.

Parameters

- **cache_partition_id** – cache partition id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_core_keys(*array_id=None*)
List cores for the given array.

Parameters **array_id** – array id – str

Returns core info with first and last available dates – list

get_core_stats(*core_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given core.

Parameters

- **core_id** – core id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_database_keys(*array_id=None*)
List databases for the given array.

Parameters **array_id** – array id – str

Returns database info with first and last available dates – list

get_database_stats(*database_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given database.

Parameters

- **database_id** – database id – str
- **metrics** – performance metrics to retrieve – str or list

- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_days_to_full(*array_id=None*, *array_to_full=False*, *srp_to_full=False*, *thin_pool_to_full=False*)

Get days to full information.

Requires at least 10 Days of Performance data, available categories are 'Array', 'SRP', and 'ThinPool'.

Parameters

- **array_id** – array id – str
- **array_to_full** – get array days to full info – bool
- **srp_to_full** – get storage resource pool days to full info – bool
- **thin_pool_to_full** – get thin pool days to full info – bool

Returns days to full information – list

get_device_group_keys(*array_id=None*)

List device groups for the given array.

Parameters **array_id** – array id – str

Returns device group info with first and last available dates – list

get_device_group_stats(*device_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given device group.

Parameters

- **device_group_id** – device group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_director_info(***kwargs*)

get_disk_group_keys(*array_id=None*)

List disk groups for the given array.

Parameters **array_id** – array id – str

Returns disk info with first and last available dates – list

get_disk_group_stats(*disk_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given disk group.

Parameters

- **disk_group_id** – disk group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_disk_keys(*array_id=None*)
List disks for the given array.

Parameters **array_id** – array id – str

Returns disk info with first and last available dates – list

get_disk_stats(*disk_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given disk.

Parameters

- **disk_id** – disk id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_disk_technology_pool_keys(*array_id=None*)
List disk technology pools for the given array.

Parameters **array_id** – array id – str

Returns disk technology pool info with first and last available dates – list

get_disk_technology_pool_stats(*disk_tech_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given disk technology.

Parameters

- **disk_tech_id** – disk technology id – str
- **metrics** – performance metrics to retrieve – str or list

- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_eds_director_keys(*array_id=None*)
List EDS directors for the given array.

Parameters **array_id** – array id – str

Returns EDS director info with first and last available dates – list

get_eds_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given EDS director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_eds_emulation_keys(*array_id=None*)
List EDS emulations for the given array.

Parameters **array_id** – array id – str

Returns EDS emulation info with first and last available dates – list

get_eds_emulation_stats(*emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given EDS emulation.

Parameters

- **emulation_id** – emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_external_director_keys(*array_id=None*)
List external directors for the given array.

Parameters **array_id** – array id – str

Returns external directors with first and last available dates – list

get_external_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given external director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_external_disk_group_keys(*array_id=None*)
List external disk groups for the given array.

Parameters **array_id** – array id – str

Returns external disk groups with first and last available dates – list

get_external_disk_group_stats(*disk_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given external disk group.

Parameters

- **disk_group_id** – disk group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_external_disk_keys(*array_id=None*)
List external disks for the given array.

Parameters **array_id** – array id – str

Returns external disks with first and last available dates – list

get_external_disk_stats(*disk_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given external disk.

Parameters

- **disk_id** – disk id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_fe_director_list(***kwargs*)

get_fe_director_metrics(***kwargs*)

get_fe_port_list(***kwargs*)

get_fe_port_metrics(***kwargs*)

get_fe_port_util_last4hrs(***kwargs*)

get_ficon_emulation_keys(*array_id=None*)
List FICON emulations for the given array.

Parameters **array_id** – array id – str

Returns FICON emulation info with first and last available dates – list

get_ficon_emulation_stats(*ficon_emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given FICON emulation.

Parameters

- **ficon_emulation_id** – FICON emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_ficon_emulation_thread_keys(*array_id=None*)
List FICON emulation threads for the given array.

Parameters **array_id** – array id – str

Returns FICON emulation thread info with first and last available dates – list

get_ficon_emulation_thread_stats(*ficon_emulation_thread_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given FICON emulation thread.

Parameters

- **ficon_emulation_thread_id** – FICON emulation thread id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_ficon_port_thread_keys(*array_id=None*)

List FICON port threads for the given array.

Parameters **array_id** – array id – str

Returns FICON port info with first and last available dates – list

get_ficon_port_thread_stats(*ficon_port_thread_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given FICON port thread.

Parameters

- **ficon_port_thread_id** – FICON port thread id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_frontend_director_keys(*array_id=None*)

List FE directors for the given array.

Parameters **array_id** – array id – str

Returns FE directors with first and last available dates – list

get_frontend_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given FE director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str

- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_frontend_emulation_keys(*array_id=None*)
List FE emulations for the given array.

Parameters **array_id** – array id – str

Returns BE emulation info with first and last available dates – list

get_frontend_emulation_stats(*emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given FE emulation.

Parameters

- **emulation_id** – emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_frontend_port_keys(*director_id*, *array_id=None*)
List FE ports for the given array.

Parameters

- **director_id** – array id – str
- **array_id** – director id – str

Returns FE port info with first and last available dates – list

get_frontend_port_stats(*director_id*, *port_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given FE port.

Parameters

- **director_id** – director id – str
- **port_id** – port id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_host_keys(*array_id=None*, *start_time=None*, *end_time=None*)
List active hosts for the given array by time range.

Only active hosts from within the specified time range are returned. If no time range is provided, start and end times from array level are used.

Parameters

- **array_id** – array id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns host info with first and last available dates – list

get_host_metrics(***kwargs*)

get_host_stats(*host_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given host.

Performance details will only be returned if the host was active during the specified time range. If no time range is provided, start and end times from array level are used.

Parameters

- **host_id** – host id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_im_director_keys(*array_id=None*)
List IM directors for the given array.

Parameters **array_id** – array id – str

Returns IM directors with first and last available dates – list

get_im_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given IM director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_im_emulation_keys(*array_id=None*)
List IM emulations for the given array.

Parameters **array_id** – array id – str

Returns IM emulation info with first and last available dates – list

get_im_emulation_stats(*emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given IM emulation.

Parameters

- **emulation_id** – emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_initiator_by_port_keys(*array_id=None*, *start_time=None*, *end_time=None*)
List active initiators by port for the given array by time range

Only active initiators from within the specified time range are returned. If no time range is provided, start and end times from array level are used.

Parameters

- **array_id** – array id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns host info with first and last available dates – list

get_initiator_by_port_stats(*initiator_by_port_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given initiator.

Performance details will only be returned if the initiator was active during the specified time range. If no time range is provided, start and end times from array level are used.

Parameters

- **initiator_by_port_id** – initiator by port id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str

- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_initiator_perf_keys(*array_id=None*, *start_time=None*, *end_time=None*)
List active initiators for the given array by time range

Only active initiators from within the specified time range are returned. If no time range is provided, start and end times from array level are used.

Parameters

- **array_id** – array id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns host info with first and last available dates – list

get_initiator_stats(*initiator_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given initiator.

Performance details will only be returned if the initiator was active during the specified time range. If no time range is provided, start and end times from array level are used.

Parameters

- **initiator_id** – initiator id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_ip_interface_keys(*array_id=None*)
List IP interfaces for the given array.

Parameters **array_id** – array id – str

Returns IP interface info with first and last available dates – list

get_ip_interface_stats(*ip_interface_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given IP interface.

Parameters

- **ip_interface_id** – IP interface id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str

- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_iscsi_target_keys(*array_id=None*)
List iSCSI targets for the given array.

Parameters **array_id** – array_id: array id – str

Returns iSCSI interfaces info with first and last available dates – list

get_iscsi_target_stats(*iscsi_target_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given iSCSI target.

Parameters

- **iscsi_target_id** – iSCSI target id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_last_available_timestamp(*array_id=None*)
Get the last recorded performance timestamp.

Parameters **array_id** – array_id: array id – str

Returns timestamp – int

Raises ResourceNotFoundException

get_perf_category_threshold_settings(***kwargs*)

get_perf_threshold_categories(***kwargs*)

static get_performance_categories_list()
Get the list of supported performance categories.

Returns categories – list

get_performance_key_list(*category*, *array_id=None*, *director_id=None*, *storage_group_id=None*, *storage_container_id=None*, *storage_resource_id=None*, *start_time=None*, *end_time=None*)
Get performance key list for a given performance category.

Parameters

- **category** – performance category – str
- **array_id** – array id – str
- **director_id** – director id – str
- **storage_group_id** – storage group id – str
- **storage_container_id** – storage container id – str

- **storage_resource_id** – storage resource id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns category performance keys – list

Raises InvalidInputException

static get_performance_metrics_list(*category*, *kpi_only=False*)

For a given category, return the list of valid metrics.

Parameters

- **category** – performance category – str
- **kpi_only** – if only KPI metrics should be returned – bool

Returns metrics – list

get_performance_stats(*category*, *metrics*, *data_format='Average'*, *array_id=None*, *request_body=None*, *start_time=None*, *end_time=None*, *recency=None*)

Retrieve the performance statistics for a given category and object.

Parameters

- **category** – category id – str
- **array_id** – array id – str
- **metrics** – performance metrics, options are individual metrics, a list of metrics, 'KPI' for KPI metrics only, and 'ALL' for all metrics – str/list
- **data_format** – response data format 'Average' or 'Maximum' – str
- **request_body** – request params and object IDs – dict
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

Raises VolumeBackendAPIException, InvalidInputException

get_port_group_keys(*array_id=None*)

List port group for the given array.

Parameters **array_id** – array_id: array id – str

Returns port group info with first and last available

get_port_group_metrics(***kwargs*)

get_port_group_stats(*port_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given port group.

Parameters

- **port_group_id** – port group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str

- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_rdf_director_keys(*array_id=None*)
List RDF directors for the given array.

Parameters **array_id** – array id – str

Returns RDF directors with first and last available dates – list

get_rdf_director_stats(*director_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given RDF director.

Parameters

- **director_id** – director id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_rdf_emulation_keys(*array_id=None*)
List RDF emulations for the given array.

Parameters **array_id** – array id – str

Returns RDF emulation info with first and last available dates – list

get_rdf_emulation_stats(*emulation_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given RDF emulation.

Parameters

- **emulation_id** – emulation id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_rdf_port_keys(*director_id*, *array_id=None*)
List RDF ports for the given array.

Parameters

- **director_id** – array id – str
- **array_id** – director id – str

Returns RDF port info with first and last available dates – list

get_rdf_port_stats(*director_id*, *port_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given RDF port.

Parameters

- **director_id** – director id – str
- **port_id** – port id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_rdfa_keys(*array_id=None*)
List RDFA groups for the given array.

Parameters **array_id** – array_id: array id – str

Returns RDFA info with first and last available dates – list

get_rdfa_stats(*rdfa_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given RDFA group.

Parameters

- **rdfa_group_id** – RDFA group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_rdfs_keys(*array_id=None*)
List RDFS groups for the given array.

Parameters **array_id** – array_id: array id – str

Returns RDFS info with first and last available dates – list

get_rdfs_stats(*rdfs_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)
List time range performance data for given RDFS group.

Parameters

- **rdfs_group_id** – RDFS group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_container_keys(*array_id=None*)

List storage containers for the given array.

Parameters **array_id** – array id – str

Returns storage container info with first and last available dates – list

get_storage_container_stats(*storage_container_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage container.

Parameters

- **storage_container_id** – storage container id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_group_by_pool_keys(*storage_group_id*, *array_id=None*, *start_time=None*, *end_time=None*)

List storage groups by thin pool for the given array by time range.

Only active pools from within the specified time range are returned. If no time range is provided, start and end times from array level are used.

Parameters

- **storage_group_id** – storage group id – str
- **array_id** – array id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns pool info with first and last available dates – list

get_storage_group_by_pool_stats(*storage_group_id*, *pool_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage group by pool.

Performance details will only be returned if the storage was active during the specified time range. If no time range is provided, start and end times from array level are used.

Parameters

- **storage_group_id** – storage group id – str
- **pool_id** – pool id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_group_keys(*array_id=None*)

List storage groups for the given array.

Parameters **array_id** – array id – str

Returns storage container info with first and last available dates – list

get_storage_group_metrics(***kwargs*)

get_storage_group_stats(*storage_group_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage group.

Parameters

- **storage_group_id** – storage group id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_resource_by_pool_keys(*storage_container_id*, *storage_resource_id*, *array_id=None*, *start_time=None*, *end_time=None*)

List storage resource by pool for the given array by time range.

Only active pools from within the specified time range are returned. If no time range is provided, start and end times from array level are used.

Parameters

- **storage_container_id** – storage container id – str

- **storage_resource_id** – storage resource id – str
- **array_id** – array id – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

Returns pool info with first and last available dates – list

get_storage_resource_by_pool_stats(*storage_container_id*, *storage_resource_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage resource.

Performance details will only be returned if the pool was active during the specified time range. If no time range is provided, start and end times from array level are used.

Parameters

- **storage_container_id** – storage container id – str
- **storage_resource_id** – storage resource id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_resource_keys(*array_id=None*)

List storage resources for the given array.

Parameters **array_id** – array id – str

Returns storage resource info with first and last available dates – list

get_storage_resource_pool_keys(*array_id=None*)

List storage resource pools for the given array.

Parameters **array_id** – array id – str

Returns storage resource pool info with first and last available dates – list

get_storage_resource_pool_stats(*srp_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage resource pools.

Parameters

- **srp_id** – storage resource pool id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str

- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_storage_resource_stats(*storage_container_id*, *storage_resource_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given storage resource.

Parameters

- **storage_container_id** – storage container id – str
- **storage_resource_id** – storage resource id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_thin_pool_keys(*array_id=None*)

List thin pools for the given array.

Parameters **array_id** – array id – str

Returns thin pools with first and last available dates – list

get_thin_pool_stats(*thin_pool_id*, *metrics*, *array_id=None*, *data_format='Average'*, *start_time=None*, *end_time=None*, *recency=None*)

List time range performance data for given thin pool.

Parameters

- **thin_pool_id** – thin pool id – str
- **metrics** – performance metrics to retrieve – str or list
- **array_id** – array id – str
- **data_format** – response data format 'Average' or 'Maximum' – str
- **start_time** – timestamp in milliseconds since epoch – str
- **end_time** – timestamp in milliseconds since epoch – str
- **recency** – check recency of timestamp in minutes – int

Returns performance metrics – dict

get_threshold_categories()

Get a list of performance threshold categories.

Returns performance threshold categories – list

get_threshold_category_settings(*category*)

Get performance threshold category settings.

Parameters **category** – category id – str

Returns category settings – dict

static get_timestamp_by_hour(*start_time=None*, *end_time=None*, *hours_difference=None*)
Get timestamp difference in hours from supplied time.

If start time is provided but not end time, the time difference will be after the start time.

If end time is provided but not start time, the time difference will be before the end time.

If neither start or end time are provided, or both are incorrectly provided, the time difference is from the current time.

Parameters

- **start_time** – timestamp in milliseconds since epoch – int
- **end_time** – timestamp in milliseconds since epoch – int
- **hours_difference** – difference in hours – int

Returns timestamp in milliseconds since epoch – str

is_array_performance_registered(*array_id=None*)
Check if an array is registered for diagnostic performance data.

This will return False is an array is registered for real-time data but not for diagnostic performance data.

Parameters **array_id** – array id – str

Returns bool

is_timestamp_current(*timestamp*, *minutes=None*)
Check if the timestamp is less than a user specified set of minutes.

If no minutes value is provided, self.recency is used. Seven minutes is recommended to provide a small amount of time for the STP daemon to record the next set of metrics in five minute intervals.

Parameters

- **timestamp** – timestamp in milliseconds since epoch – int
- **minutes** – timestamp recency in minutes – int

Returns if timestamp is less than recency value – bool

set_array_id(*array_id*)
Set the array id.

Parameters **array_id** – array id – str

set_perf_threshold_and_alert(***kwargs*)

set_perfthresholds_csv(***kwargs*)

set_recency(*minutes*)
Set the recency value in minutes.

Parameters **minutes** – recency minutes – int

set_thresholds_from_csv(*csv_file_path*, *kpi_only=True*)
Set performance thresholds using a CSV file.

Reads CSV file and sets performance threshold metrics on the values contained within. The following headers are required: category, metric, firstthreshold, secondthreshold, notify, kpi

It is advisable to generate the CSV file from the function performance.generate_threshold_settings_csv() and edit those values within that you would like to change.

Parameters

- **csv_file_path** – path to CSV file – str
- **kpi_only** – set only KPI thresholds – bool

set_timestamp(*timestamp*)
Set the performance timestamp.

Parameters **timestamp** – the performance timestamp – str

update_threshold_settings(*category*, *metric*, *first_threshold*, *second_threshold*, *alert=True*, *first_threshold_occurrences=3*, *first_threshold_samples=5*, *first_threshold_severity='WARNING'*, *second_threshold_occurrences=3*, *second_threshold_samples=5*, *second_threshold_severity='CRITICAL'*)
Edit an existing global threshold across all arrays.

Parameters

- **category** – category id – str
- **metric** – performance metric – str
- **first_threshold** – first threshold value – int
- **second_threshold** – second threshold value – int
- **alert** – alert on/off – bool
- **first_threshold_occurrences** – error occurrences – int
- **first_threshold_samples** – error samples – int
- **first_threshold_severity** – error severity, valid values are 'INFORMATION', 'WARNING', and 'CRITICAL' – str
- **second_threshold_occurrences** – error occurrences – int
- **second_threshold_samples** – error samples – int
- **second_threshold_severity** – error severity, valid values are 'INFORMATION', 'WARNING', and 'CRITICAL' – str

Returns operation success details – dict

static validate_category(*category*)
Check that a supplied category is valid.

Raises InvalidInputException

10.1.5 PyU4V.provisioning

provisioning.py.

class PyU4V.provisioning.**ProvisioningFunctions**(*array_id*, *rest_client*)
Bases: object

ProvisioningFunctions.

add_child_sg_to_parent_sg(***kwargs*)

add_child_storage_group_to_parent_group(*child_storage_group*, *parent_storage_group*)
Add a storage group to a parent storage group.

This method adds an existing storage group to another storage group, i.e. cascaded storage groups.

Parameters

- **child_storage_group** – child storage group id – str
- **parent_storage_group** – parent storage group id – str

Returns storage group details – dict

add_existing_vol_to_sg(***kwargs*)

add_existing_volume_to_storage_group(*storage_group_id*, *vol_ids*, *_async=False*)
Expand an existing storage group by adding existing volumes.

Parameters

- **storage_group_id** – storage group id – str
- **vol_ids** – volume device id(s) – str or list
- **_async** – if call should be async – bool

Returns storage group details – dict

add_new_vol_to_storagegroup(***kwargs*)

add_new_volume_to_storage_group(*storage_group_id*, *num_vols*, *vol_size*, *cap_unit*, *_async=False*, *vol_name=None*, *create_new_volumes=None*)
Expand an existing storage group by adding new volumes.

Parameters

- **storage_group_id** – storage group id – str
- **num_vols** – number of volumes to be created – int
- **vol_size** – the volume size – str
- **cap_unit** – capacity unit (MB, GB, TB, CYL) – str
- **_async** – if call should be async – bool
- **vol_name** – name to give to the volume, optional – str
- **create_new_volumes** – new volumes only, no ro-use – bool

Returns storage group details – dict

create_empty_sg(***kwargs*)

create_empty_storage_group(*srp_id*, *storage_group_id*, *service_level*, *workload*, *disable_compression=False*, *_async=False*)
Create an empty storage group.

Set the disable_compression flag for disabling compression on an All Flash array (where compression is on by default).

Parameters

- **srp_id** – SRP id – str
- **storage_group_id** – storage group id – str
- **service_level** – service level id – str
- **workload** – workload id – str
- **disable_compression** – disable compression – bool
- **_async** – if call should be async – bool

Returns storage group details – dict

create_host(*host_name*, *initiator_list=None*, *host_flags=None*, *init_file=None*, *_async=False*)
Create a host with the given initiators.

Accepts either initiator_list, e.g. [10000000ba873cbf, 10000000ba873cba], or file. The initiators must not be associated with another host. An empty host can also be created by not passing any initiator ids.

Parameters

- **host_name** – name of the new host – str
- **initiator_list** – list of initiators – list
- **host_flags** – optional host flags to apply – dict
- **init_file** – path to file containing initiator names – str
- **_async** – if call should be _async – bool

Returns new host details – dict

create_host_group(*host_group_id*, *host_list*, *host_flags=None*, *_async=False*)
Create a host group containing the given hosts.

Parameters

- **host_group_id** – name of the new host group – str
- **host_list** – hosts – list
- **host_flags** – optional host flags to apply – dict
- **_async** – if call should be async – bool

Returns new host group details – dict

create_hostgroup(***kwargs*)

create_masking_view_existing_components(*port_group_name*, *masking_view_name*, *storage_group_name*, *host_name=None*, *host_group_name=None*, *_async=False*)
Create a new masking view using existing groups.

Must enter either a host name or a host group name, but not both.

Parameters

- **port_group_name** – name of the port group – str
- **masking_view_name** – name of the new masking view – str
- **storage_group_name** – name of the storage group – str
- **host_name** – name of the host (initiator group) – str
- **host_group_name** – name of host group – str
- **_async** – if command should be run asynchronously – bool

Returns masking view details – dict

Raises InvalidInputException

create_multiport_port_group(*port_group_id*, *ports*)
Create a new port group.

Parameters

- **port_group_id** – name of the new port group – str
- **ports** – port dicts Example: [{'directorId': director_id, 'portId': port_id}] – list

Returns new port group details – dict

create_multiport_portgroup(***kwargs*)

create_non_empty_storage_group(*srp_id*, *storage_group_id*, *service_level*, *workload*, *num_vols*, *vol_size*, *cap_unit*, *disable_compression=False*, *_async=False*)

Create a new storage group with the specified volumes.

Generates a dictionary for json formatting and calls the create_sg function to create a new storage group with the specified volumes. Set the disable_compression flag for disabling compression on an All Flash array (where compression is on by default).

Parameters

- **srp_id** – SRP id – str
- **storage_group_id** – storage group id – str
- **service_level** – service level id – str
- **workload** – workload id – str
- **num_vols** – number of volumes to be created – int
- **vol_size** – the volume size – str
- **cap_unit** – capacity unit (MB, GB, TB, CYL) – str
- **disable_compression** – disable compression – bool
- **_async** – if call should be async – bool

Returns storage group details – dict

create_non_empty_storagegroup(***kwargs*)

create_port_group(*port_group_id*, *director_id*, *port_id*)

Create a new port group.

Parameters

- **port_group_id** – name of the new port group - str
- **director_id** – director id – str
- **port_id** – port id – str

Returns new port group details – dict

create_port_group_from_file(*file_name*, *port_group_id*)

Given a file with director:port pairs, create a portgroup.

Each director:port pair must be on a new line. Example director:port - FA-1D:4.

Parameters

- **file_name** – path to the file – str
- **port_group_id** – name for the port group – str

Returns new port group details – dict

create_portgroup(***kwargs*)

create_portgroup_from_file(***kwargs*)

create_storage_group(*srp_id*, *sg_id*, *slo*, *workload=None*, *do_disable_compression=False*, *num_vols=0*, *vol_size='0'*, *cap_unit='GB'*, *allocate_full=False*, *_async=False*, *vol_name=None*)

Create the volume in the specified storage group.

Parameters

- **srp_id** – SRP id – str
- **sg_id** – storage group id – str
- **slo** – service level id – str
- **workload** – workload id – str
- **do_disable_compression** – disable compression – bool
- **num_vols** – number of volumes to be created – int
- **vol_size** – the volume size – str
- **cap_unit** – capacity unit (MB, GB, TB, CYL) – str
- **allocate_full** – allocate full capacity – bool
- **_async** – if call should be async – bool
- **vol_name** – name to give to the volume, optional – str

Returns storage group details – dict

create_volume_from_sg_return_dev_id(***kwargs*)

create_volume_from_storage_group_return_id(*volume_name*, *storage_group_id*, *vol_size*, *cap_unit='GB'*)

Create a new volume in the given storage group.

Parameters

- **volume_name** – volume name – str
- **storage_group_id** – storage group id – str
- **vol_size** – volume size – str
- **cap_unit** – capacity unit (MB, GB, TB, CYL) – str

Returns device id – str

deallocate_volume(*device_id*)

Deallocate all tracks on a volume.

Necessary before deletion. Please note that it is not possible to know exactly when a de-allocation is complete. This method will return when the array has accepted the request for de-allocation; the de-allocation itself happens as a background task on the array.

Parameters **device_id** – device id – str

Returns volume details – dict

delete_host(*host_id*)

Delete a given host.

Cannot delete if associated with a masking view.

Parameters **host_id** – name of the host – str

delete_host_group(*host_group_id*)
Delete a given host group.

Cannot delete if associated with a masking view.

Parameters **host_group_id** – name of the hostgroup – str

delete_hostgroup(***kwargs*)

delete_masking_view(*maskingview_name*)
Delete a masking view.

Parameters **maskingview_name** – masking view name – str

delete_port_group(*port_group_id*)
Delete a port group.

Parameters **port_group_id** – name of the port group – str

delete_portgroup(***kwargs*)

delete_storage_group(*storage_group_id*)
Delete a given storage group.

A storage group cannot be deleted if it is associated with a masking view.

Parameters **storage_group_id** – storage group id – str

delete_storagegroup(***kwargs*)

delete_volume(***kwargs*)
Delete a volume.

Parameters **device_id** – device id – str

extend_volume(*device_id*, *new_size*, *_async=False*, *rdf_group_num=None*)
Extend a volume.

Parameters

- **device_id** – device id – str
- **new_size** – the new size for the device – int
- **_async** – if call should be async – bool
- **rdf_group_num** – RDF group number to extend R2 device in same operation – int

Returns volume details – dict

find_host_lun_id_for_vol(***kwargs*)

find_host_lun_id_for_volume(*masking_view_id*, *device_id*)
Find the host_lun_id for a volume in a masking view.

Parameters

- **masking_view_id** – masking view id – str
- **device_id** – the device id – str

Returns host lun id – str

find_low_volume_utilization(*low_utilization_percentage*, *csvname*)
Find volumes under a certain utilization threshold.

Function to find volumes under a specified percentage, (e.g. find volumes with utilization less than 10%) - may be long running as will check all sg on array and all storage group. Only identifies volumes in storage group, note if volume is in more than one sg it may show up more than once.

Parameters

- **low_utilization_percentage** – low utilization percent – int
- **csvname** – filename for CSV output file – str

find_volume_device_id(*volume_name*)
Given a volume identifier, find the corresponding device_id.

Parameters **volume_name** – the volume name – str

Returns device id – str

find_volume_identifier(*device_id*)
Get the volume identifier of a volume.

Parameters **device_id** – device id – str

Returns volume identifier – str

static format_director_port(*director*, *port*)
Format separate director port into single string.

Parameters

- **director** – director e.g. FA-2D – str
- **port** – port e.g. 4 – str

Returns formatted director:port string –str

get_active_masking_view_connections()
Get list of active connections from any masking view.

Returns masking view name, connection details – str, list

get_any_director_port(*director*, *filters=None*)
Get a non-GuestOS port from a director.

Parameters

- **director** – director to search for ports with – str
- **filters** – filters to apply when search for port – str

Returns port – int

get_available_initiator(*director_type=None*)
Get an available initiator.

Parameters **director_type** – director type filter – str

Returns single available initiator – str

get_available_initiator_wwn_as_list()
Get an available initiator wwn string in a list.

Returns single available initiator wwn – list

get_child_sg_from_parent(***kwargs*)

get_child_storage_groups_from_parent(*parent_name*)
Get child storage group list from parent storage group.

Parameters **parent_name** – parent sg name – str

Returns child sg details – list

get_common_masking_views(***kwargs*)

get_compressibility_report(*srp_id*)
Get a specified SRP Compressibility Report.

Parameters **srp_id** – srp id – str

Returns compressibility reports – list

get_director(*director*)
Query for details of a director for a symmetrix.

Parameters **director** – the director ID e.g. FA-1D – str

Returns director details – dict

get_director_list()
Query for details of Symmetrix directors for a symmetrix.

Returns directors – list

get_director_port(*director*, *port_no*)
Get details of the symmetrix director port.

Parameters

- **director** – the director ID e.g. FA-1D – str
- **port_no** – the port number e.g. 1 – str

Returns director port details – dict

get_director_port_list(*director*, *filters=None*)
Get list of the ports on a particular director.

Can be filtered by optional parameters, please see documentation.

Parameters

- **director** – the director ID e.g. FA-1D – str
- **filters** – optional filters - dict

Returns port key dicts – list

get_element_from_masking_view(*maskingview_name*, *portgroup=False*, *host=False*, *storagegroup=False*)
Return the name of the specified element from a masking view.

Parameters

- **maskingview_name** – masking view name – str
- **portgroup** – port group name – str
- **host** – the host name – str
- **storagegroup** – storage group name – str

Returns specified element name – str

Raises ResourceNotFoundException

get_fa_directors()
Get all FA directors on the array.

Returns fa director strings – list

get_host(*host_id*)
Get details on a host on the array.

Parameters **host_id** – the name of the host – str

Returns host details – dict

get_host_from_masking_view(*masking_view_id*)
Given a masking view, get the associated host or host group.

Parameters **masking_view_id** – name of the masking view – str

Returns host id – str

get_host_from_maskingview(***kwargs*)

get_host_group(*host_group_id*)
Get details on a host group on the array.

Parameters **host_group_id** – name of the host group – str

Returns host group details – dict

get_host_group_list(*filters=None*)
Get list of host group(s) on the array.

See unisphere documentation for applicable filters.

Parameters **filters** – optional list of filters – dict

Returns host group list – list

get_host_list(*filters=None*)
Get list of the hosts on the array.

See documentation for applicable filters.

Parameters **filters** – optional list of filters – dict

Returns hosts – list

get_hostgroup(***kwargs*)

get_hostgroup_list(***kwargs*)

get_in_use_initiator(*director_type=None*)
Get an initiator that is in use.

Parameters **director_type** – director type filter – str

Returns single in-use initiator – str

get_in_use_initiator_list_from_array()
Get the list of initiators which are in-use from the array.

Gets the list of initiators from the array which are in hosts/ initiator groups.

Returns in-use initiators – list

get_initiator(*initiator_id*)
Get details of an initiator.

Parameters **initiator_id** – initiator id – str

Returns initiator details – dict

get_initiator_group_from_initiator(*initiator*)
Given an initiator, get its corresponding initiator group, if any.

Parameters **initiator** – the initiator id – str

Returns found initiator group name – str or None

get_initiator_ids_from_host(*host_id*)
Get initiator details from a host.

Parameters **host_id** – name of the host – str

Returns initiator IDs – list

get_initiator_list(*params=None*)
Retrieve initiator list from the array.

Parameters **params** – optional params – dict

Returns initiators – list

get_iscsi_ip_address_and_iqn(*port_id*)
Get the ip addresses from the director port.

Parameters **port_id** – director port identifier – str

Returns ip addresses, iqn – list, str

get_masking_view(*masking_view_name*)
Get details of a masking view.

Parameters **masking_view_name** – the masking view name – str

Returns masking view details – dict

get_masking_view_connections(*masking_view_id*, *filters=None*)
Get all connection information for a given masking view.

Parameters

- **masking_view_id** – masking view id – str
- **filters** – optional filter parameters – dict

Returns masking view connection dicts – list

get_masking_view_from_storage_group(*storage_group*)
Get the associated masking views from a given storage group.

Parameters **storage_group** – name of the storage group – str

Returns Masking views – list

get_masking_view_list(*filters=None*)
Get a masking view or list of masking views.

See unisphere documentation for possible filters.

Parameters **filters** – filters – dict

Returns masking views – list

get_masking_views_by_host(***kwargs*)

get_masking_views_by_initiator_group(*initiator_group_name*)
Given a host (initiator group), retrieve the masking view name.

Retrieve the list of masking views associated with the given initiator group.

Parameters **initiator_group_name** – name of the initiator group – str

Returns masking view names – list

get_masking_views_from_host (*host_id*)
Retrieve masking view information for a specified host.

Parameters **host_id** – name of the host – str

Returns masking views – list

get_masking_views_from_storage_group (*storagegroup*)
Return any masking views associated with a storage group.

Parameters **storagegroup** – storage group name – str

Returns masking view list – list

get_maskingview_connections (***kwargs*)

get_mv_from_sg (***kwargs*)

get_mvs_from_host (***kwargs*)

get_num_vols_in_sg (***kwargs*)

get_num_vols_in_storage_group (*storage_group_name*)
Get the number of volumes in a storage group.

Parameters **storage_group_name** – storage group name – str

Returns number of volumes – int

get_port_group (*port_group_id*)
Get port group details.

Parameters **port_group_id** – name of the portgroup – str

Returns port group details – dict

get_port_group_common_masking_views (*port_group_name*, *initiator_group_name*)
Get common masking views for a given port group and initiator group.

Parameters

- **port_group_name** – port group name – str
- **initiator_group_name** – initiator group name – str

Returns masking views - list

get_port_group_from_masking_view (*masking_view_id*)
Given a masking view, get the associated port group.

Parameters **masking_view_id** – masking view name – str

Returns name of the port group – str

get_port_group_list (*filters=None*)
Get port group details.

Parameters **filters** – optional filters – dict

Returns port groups – list

get_port_identifier (*director*, *port_no*)
Get the identifier (wwn) of the physical port.

Parameters

- **director** – the id of the director – str
- **port_no** – the number of the port – str

Returns wwn (FC) or iqn (iscsi) – str or None

get_port_list (*filters=None*)
Query for a list of Symmetrix port keys.

Note a mixture of Front end, back end and RDF port specific values are not allowed. See UniSphere documentation for possible values.

Parameters **filters** – optional filters e.g. {'vnx_attached': 'true'} – dict

Returns port key dicts – list

get_portgroup (***kwargs*)

get_portgroup_from_maskingview (***kwargs*)

get_portgroup_list (***kwargs*)

get_ports_from_pg (***kwargs*)

get_ports_from_port_group (*port_group*)
Get a list of port identifiers from a port group.

Parameters **port_group** – name of the portgroup – list

Returns port ids – list

get_service_level (*service_level_id*)
Get details on a specific service level.

Parameters **service_level_id** – service level agreement – str

Returns service level details – dict

get_service_level_list (*filters=None*)
Retrieve the list of service levels from the array.

Parameters **filters** – optional filters – dict

Returns service level names – list

get_size_of_device_on_array (*device_id*)
Get the size of the volume from the array.

Parameters **device_id** – device id – str

Returns size – float

get_slo (***kwargs*)

get_slo_list (***kwargs*)

get_srp (*srp*)
Get details on a specific SRP.

Parameters **srp** – storage resource pool id – str

Returns srp details – dict

get_srp_list (*filters=None*)
Get a list of available SRPs on a given array.

Parameters **filters** – filter parameters – dict

Returns SRPs – list

get_storage_group(*storage_group_name*)
Given a name, return storage group details.

Parameters **storage_group_name** – name of the storage group – str

Returns storage group details – dict

get_storage_group_demand_report(*srp_id=None*)
Get the storage group demand report.

Get the storage group demand report from Unisphere.

Parameters **srp_id** – id of the Storage Resource Pool – str

Returns demand report – dict

get_storage_group_from_masking_view(*masking_view_id*)
Given a masking view, get the associated storage group.

Parameters **masking_view_id** – masking view name – str

Returns name of the storage group – str

get_storage_group_from_volume(*volume_id*)
Retrieve storage group information for a specified volume.

Parameters **volume_id** – device id – str

Returns storage groups – list

get_storage_group_list(*filters=None*)
Return a list of storage groups.

Parameters **filters** – filter parameters – dict

Returns storage groups – list

get_storagegroup_from_maskingview(***kwargs*)

get_storagegroup_from_vol(***kwargs*)

get_target_wwns_from_pg(***kwargs*)

get_target_wwns_from_port_group(*port_group_id*)
Get the director ports' WWNs.

Parameters **port_group_id** – the name of the port group – str

Returns target_wwns – target wwns for the port group – list

get_vol_effective_wwn_details_84(***kwargs*)

get_vols_from_storagegroup(***kwargs*)

get_volume(*device_id*)
Get a volume from array.

Parameters **device_id** – device id – str

Returns volume details – dict

get_volume_effective_wwn_details(*vol_list*, *output_file_name=None*)
Get the effective wwn for a list of vols.

Get volume details for a list of volume device ids.

Parameters

- **vol_list** – device id(s) – list

- **output_file_name** – name of the output file – str

Returns volume details list (nested) – list

get_volume_list(*filters=None*)
Get list of volumes from array.

Parameters **filters** – filters parameters – dict

Returns device ids – list

get_volumes_from_storage_group(*storage_group_id*)
Retrieve volume information associated with a given storage group.

Parameters **storage_group_id** – storage group id – name

Returns device ids – list

get_workload_settings()
Get valid workload options from array.

Returns workload settings – list

is_child_sg_in_parent_sg(***kwargs*)

is_child_storage_group_in_parent_storage_group(*child_name*, *parent_name*)
Check if a child storage group is a member of a parent group.

Parameters

- **child_name** – child sg name – str
- **parent_name** – parent sg name – str

Returns bool

is_compression_capable()
Check if array is compression capable.

Returns bool

is_initiator_in_host(*initiator*)
Check to see if a given initiator is already assigned to a host.

Parameters **initiator** – the initiator ID – str

Returns if initiator is assigned – bool

is_volume_in_storage_group(*device_id*, *storage_group_id*)
See if a volume is a member of the given storage group.

Parameters

- **device_id** – device id – str
- **storage_group_id** – storage group id – name

Returns bool

is_volume_in_storagegroup(***kwargs*)

modify_host(*host_id*, *host_flag_dict=None*, *remove_init_list=None*, *add_init_list=None*, *new_name=None*)
Modify an existing host.

Only one parameter can be modified at a time.

Parameters

- **host_id** – host name – str
- **host_flag_dict** – host flags – dict
- **remove_init_list** – initiators to be removed – list
- **add_init_list** – initiators to be added – list
- **new_name** – new host name – str

Returns modified host details – dict

modify_host_group(*host_group_id*, *host_flag_dict=None*, *remove_host_list=None*, *add_host_list=None*, *new_name=None*)

Modify an existing host group.

Only one parameter can be modified at a time.

Parameters

- **host_group_id** – name of the host group – str
- **host_flag_dict** – host flags – dict
- **remove_host_list** – hosts to be removed – list
- **add_host_list** – hosts to be added – list
- **new_name** – new name of the host group – str

Returns modified host group details – dict

modify_hostgroup(***kwargs*)

modify_initiator(*initiator_id*, *remove_masking_entry=None*, *replace_init=None*, *rename_alias=None*, *set_fcid=None*, *initiator_flags=None*)

Modify an initiator.

Only one parameter can be edited at a time.

Parameters

- **initiator_id** – initiator id – str
- **remove_masking_entry** – 'true' or 'false' – str
- **replace_init** – new initiator id – str
- **rename_alias** – ('new node name', 'new port name') – tuple
- **set_fcid** – fcid – str
- **initiator_flags** – initiator flags to set – dict

Returns modified initiator details – dict

modify_port_group(*port_group_id*, *remove_port=None*, *add_port=None*, *rename_port_group=None*)

Modify an existing port group.

Only one parameter can be modified at a time.

Parameters

- **port_group_id** – name of the port group – str
- **remove_port** – port details (director_id, port_id) – tuple
- **add_port** – port details (director_id, port_id) – tuple

- **rename_port_group** – new port group name – str

Returns modified port group details – dict

modify_portgroup(***kwargs*)

modify_service_level(*service_level_id*, *new_name*)
Modify an SLO.

Currently, the only modification permitted is renaming.

Parameters

- **service_level_id** – current name of the service level – str
- **new_name** – new name for the – str

Returns modified service level details – dict

modify_slo(***kwargs*)

modify_storage_group(*storage_group_id*, *payload*)
Modify a storage group.

Parameters

- **storage_group_id** – storage group id – str
- **payload** – request payload – dict

Returns modified storage group details – dict

move_volumes_between_storage_groups(*device_ids*, *source_storagegroup_name*, *target_storagegroup_name*, *force=False*, *_async=False*)
Move volumes to a different storage group.

Requires force set to True if volume is in a masking view.

Parameters

- **device_ids** – volume device id(s) – str or list
- **source_storagegroup_name** – originating storage group name – str
- **target_storagegroup_name** – destination storage group name – str
- **force** – force flag – bool
- **_async** – if call should be async – bool

Returns storage group details – dict

remove_child_sg_from_parent_sg(***kwargs*)

remove_child_storage_group_from_parent_group(*child_storage_group*, *parent_storage_group*)
Remove a storage group from its parent storage group.

This method removes a child storage group from its parent group.

Parameters

- **child_storage_group** – child storage group id – str
- **parent_storage_group** – parent storage group id – str

Returns storage group details – dict

remove_vol_from_storagegroup(***kwargs*)

remove_volume_from_storage_group(*storage_group_id*, *vol_id*, *_async=False*)
Remove a volume from a given storage group.

Parameters

- **storage_group_id** – storage group id – str
- **vol_id** – device id – str
- **_async** – if call should be async – bool

Returns storage group details – dict

rename_masking_view(*masking_view_id*, *new_name*)
Rename an existing masking view.

Currently, the only supported modification is "rename".

Parameters

- **masking_view_id** – current name of the masking view – str
- **new_name** – new name of the masking view – str

Returns modified masking view details – dict

rename_volume(*device_id*, *new_name*)
Rename a volume.

Parameters

- **device_id** – device id – str
- **new_name** – new name for the volume – str

set_host_io_limit_iops_or_mbps(*storage_group*, *iops*, *dynamic_distribution*, *mbps=None*)
Set the Host IO Limits on an existing storage group.

Parameters

- **storage_group** – storage group id – str
- **iops** – IO per second, min Value 100, must be specified as multiple of 100 – int
- **dynamic_distribution** – 'Always', 'Never', 'OnFailure' – str
- **mbps** – MB per second, min Value 100, must be specified as multiple of 100 – int

Returns storage group details – dict

update_storage_group_qos(*storage_group_id*, *qos_specs*)
Update the storage group instance with QoS details.

If maxIOPS or maxMBPS is in qos_specs, then DistributionType can be modified in addition to maxIOPs or/and maxMBPS. If maxIOPS or maxMBPS is NOT in qos_specs, we check to see if either is set in Storage Group. If so, then DistributionType can be modified. Example qos specs: {'maxIOPS': '4000', 'maxMBPS': '4000', 'DistributionType': 'Dynamic'}

Parameters

- **storage_group_id** – storage group id – str
- **qos_specs** – qos specifications – dict

Returns storage group details – dict

update_storagegroup_qos(***kwargs*)

10.1.6 PyU4V.replication

replication.py.

class PyU4V.replication.**ReplicationFunctions**(*array_id*, *rest_client*)
Bases: object

ReplicationFunctions.

are_vols_rdf_paired(***kwargs*)

are_volumes_rdf_paired(*remote_array*, *device_id*, *target_device*, *rdf_group*)
Check if a pair of volumes are RDF paired.

Parameters

- **remote_array** – remote array serial number – str
- **device_id** – device id – str
- **target_device** – target device id – str
- **rdf_group** – rdf group number – int

Returns paired, state – bool, string

choose_snapshot_from_list_in_console(***kwargs*)

create_storage_group_snapshot(*storage_group_id*, *snap_name*, *ttl=None*, *hours=False*)
Create a snapVx snapshot of a storage group.

To establish a new generation of an existing SnapVX snapshot for a source SG, use the same name as the existing snapshot for the new snapshot.

Parameters

- **storage_group_id** – source storage group id – str
- **snap_name** – snapshot name – str
- **ttl** – Time To Live – str
- **hours** – if TTL is in hours instead of days – bool

Returns snapshot details – dict

create_storage_group_srdf_pairings(*storage_group_id*, *remote_sid*, *srdf_mode*, *establish=None*, *_async=False*, *rdfg_number=None*, *force_new_rdf_group=False*)
SRDF protect a storage group.

Valid modes are 'Active', 'AdaptiveCopyDisk', 'Synchronous', and 'Asynchronous'.

Parameters

- **storage_group_id** – storage group id – str
- **remote_sid** – remote array id – str
- **srdf_mode** – replication mode – str
- **establish** – establish srdf – bool
- **_async** – if call should be async – bool
- **rdfg_number** – rdf group number – int
- **force_new_rdf_group** – if force command should be applied – bool

Returns storage group rdf details – dict

create_storagegroup_snap(***kwargs*)

create_storagegroup_srdf_pairings(***kwargs*)

delete_storage_group_snapshot(*storage_group_id*, *snap_name*, *gen=0*)
Delete the snapshot of a storage group.

Parameters

- **storage_group_id** – storage group id – str
- **snap_name** – snapshot name – str
- **gen** – snapshot generation number – int

delete_storage_group_srdf(*storage_group_id*, *srdf_group_number=None*)
Delete srdf pairings for a given storage group.

Parameters

- **storage_group_id** – storage group id – str
- **srdf_group_number** – srdf group number – int

Returns storage group rdf details – dict

delete_storagegroup_snapshot(***kwargs*)

delete_storagegroup_srdf(***kwargs*)

establish_storage_group_srdf(*storage_group_id*, *srdf_group_number*, *establish_options=None*, *_async=False*)
Establish io on the links for the given storage group.

Optional boolean parameters to set are 'bypass', 'metroBias', 'star', 'hop2', 'force', 'symForce', 'full'.

Parameters

- **storage_group_id** – storage group id – str
- **srdf_group_number** – srdf group number – int
- **establish_options** – establish parameters – dict
- **_async** – if call should be async – bool

Returns storage group rdf details – dict

establish_storagegroup_srdf(***kwargs*)

failback_storage_group_srdf(*storage_group_id*, *srdf_group_number*, *failback_options=None*, *_async=False*)
Failback a given storage group.

Optional boolean parameters to set are 'bypass', 'recoverPoint', 'star', 'hop2', 'force', 'symForce', 'remote'.

Parameters

- **storage_group_id** – storage group id – str
- **srdf_group_number** – srdf group number – int
- **failback_options** – failback parameters – dict
- **_async** – if call should be async – bool

Returns storage group rdf details – dict

failback_storagegroup_srdf(***kwargs*)

failover_storage_group_srdf(*storage_group_id*, *srdf_group_number*, *failover_options=None*, *_async=False*)

Failover a given storage group.

Optional boolean parameters to set are 'bypass', 'star', 'restore', 'immediate', 'hop2', 'force', 'symForce', 'remote', 'establish'.

Parameters

- **storage_group_id** – storage group id – str
- **srdf_group_number** – srdf group number – int
- **failover_options** – failover parameters – dict
- **_async** – if call should be async – bool

Returns storage group rdf details – dict

failover_storagegroup_srdf(***kwargs*)

find_expired_snapvx_snapshots()

Find all expired snapvx snapshots.

Parses through all Snapshots for array and lists those that have snapshots where the expiration date has passed however snapshots have not been deleted as they have links.

Returns expired snapshot details – list

get_array_replication_capabilities()

Check what replication facilities are available.

Returns replication capability details – dict

get_rdf_group(*rdf_number*)

Get specific rdf group details.

Parameters **rdf_number** – rdf group number – int

Returns rdf group details – dict

get_rdf_group_list()

Get rdf group list from array.

Returns rdf group list – list

get_rdf_group_number(*rdf_group_label*)

Given a group label, return the associated group number.

Parameters **rdf_group_label** – rdf group label – str

Returns rdf group number – int

get_rdf_group_volume(*rdf_number*, *device_id*)

Get specific volume details, from an RDF group.

Parameters

- **rdf_number** – rdf group number – int
- **device_id** – device id – str

Returns rdf group volume details – dict

get_rdf_group_volume_list(*rdf_number*)

Get specific volume details, from an RDF group.

Parameters **rdf_number** – rdf group number – int

Returns device ids – list

get_replication_enabled_storage_groups(*has_snapshots=False*, *has_srdf=False*)
Return list of storage groups with replication.

Parameters

- **has_snapshots** – return only storage groups with snapshots
- **has_srdf** – return only storage groups with SRDF

Returns list of storage groups with associated replication

get_replication_info()
Return replication information for an array.

Returns replication details – dict

get_snapshot_generation_details(*sg_id*, *snap_name*, *gen_num*)
Get the details for a particular snapshot generation.

Parameters

- **sg_id** – storage group id – str
- **snap_name** – snapshot name – str
- **gen_num** – generation number – int

Returns snapshot generation details – dict

get_storage_group_rep(***kwargs*)

get_storage_group_rep_list(***kwargs*)

get_storage_group_replication_details(*storage_group_id*)
Given a storage group id, return storage group srdf info.

Parameters **storage_group_id** – storage group id – str

Returns storage group replication details – dict

get_storage_group_snapshot_generation_list(*storage_group_id*, *snap_name*)
Get a snapshot and its generation count information for an sg.

The most recent snapshot will have a gen number of 0. The oldest snapshot will have a gen number = genCount - 1 (i.e. if there are 4 generations of particular snapshot, the oldest will have a gen num of 3).

Parameters

- **storage_group_id** – name of the storage group – str
- **snap_name** – the name of the snapshot – str

Returns generation numbers – list

get_storage_group_snapshot_list(*storage_group_id*)
Get a list of snapshots associated with a storage group.

Parameters **storage_group_id** – storage group id – str

Returns snapshot ids – list

get_storage_group_srdf_details(*storage_group_id*, *rdfg_num*)
Get the details for an rdf group on a particular storage group.

Parameters

- **storage_group_id** – replicated storage group id – str
- **rdfg_num** – rdf group number – int

Returns storage group rdf details – dict

get_storage_group_srdf_group_list(*storage_group_id*)
Get the rdf group numbers for a storage group.

Parameters **storage_group_id** – replicated storage group id – str

Returns rdf group numbers – list

get_storagegroup_snapshot_generation_list(***kwargs*)

get_storagegroup_snapshot_list(***kwargs*)

get_storagegroup_srdf_details(***kwargs*)

get_storagegroup_srdfg_list(***kwargs*)

is_snapvx_licensed()
Check if the snapVx feature is licensed and enabled.

Returns bool

is_vol_in_rep_session(***kwargs*)

is_volume_in_replication_session(*device_id*)
Check if a volume is in a replication session.

Parameters **device_id** – device id – str

Returns snap vx target, snap vx source, rdf group – bool, bool, list

link_gen_snapshot(*sg_id*, *snap_name*, *link_sg_name*, *_async=False*, *gen_num=0*)
Link a snapshot to another storage group.

Target storage group will be created if it does not exist.

Parameters

- **sg_id** – storage group id – str
- **snap_name** – snapshot name – str
- **link_sg_name** – target storage group name – str
- **_async** – if call should be async – bool
- **gen_num** – snapshot generation number – int

Returns snapshot details – dict

modify_storage_group_snapshot(*src_storage_grp_id*, *tgt_storage_grp_id*, *snap_name*, *link=False*, *unlink=False*, *restore=False*, *new_name=None*, *gen_num=0*, *_async=False*)
Modify a storage group snapshot.

Please note that only one parameter can be modified at a time. Default action is not to create full copy

Parameters

- **src_storage_grp_id** – name of the storage group – str
- **tgt_storage_grp_id** – target sg id (Can be None) – str
- **snap_name** – snapshot name – str
- **link** – link action required – bool

- **unlink** – unlink action required – bool
- **restore** – restore action required – bool
- **new_name** – new name for the snapshot – str
- **gen_num** – generation number – int
- **_async** – if call should be async – bool

Returns modified storage group snapshot details – dict

modify_storage_group_srdf(*storage_group_id*, *action*, *srdf_group_number*, *options=None*, *_async=False*)

Modify the state of a rdf group.

This may be a long running task depending on the size of the SRDF group, can switch to async call if required. Available actions are 'Establish', 'Split', 'Suspend', 'Restore', 'Resume', 'Failover', 'Failback', 'Swap', 'SetBias', and 'SetMode'.

Parameters

- **storage_group_id** – storage group id – str
- **action** – the rdf action –str
- **srdf_group_number** – srdf group number – int
- **options** – srdf options e.g. {setMode': {'mode': 'Asynchronous'}} – dict
- **_async** – if call should be async – bool

Returns storage group rdf details – dict

modify_storagegroup_snap(***kwargs*)

modify_storagegroup_srdf(***kwargs*)

rename_snapshot(*sg_id*, *snap_name*, *new_name*, *gen_num=0*)

Rename an existing storage group snapshot.

Parameters

- **sg_id** – storage group id – str
- **snap_name** – snapshot name – str
- **new_name** – new snapshot name – str
- **gen_num** – snapshot generation number – int

Returns snapshot details – dict

restore_snapshot(*sg_id*, *snap_name*, *gen_num=0*)

Restore a storage group to its snapshot.

Parameters

- **sg_id** – storage group id – str
- **snap_name** – snapshot name – str
- **gen_num** – snapshot generation number – int

Returns snapshot details – dict

suspend_storage_group_srdf(*storage_group_id*, *srdf_group_number*, *suspend_options=None*, *_async=False*)

Suspend IO on the links for the given storage group.

Optional boolean parameters to set are “bypass”, “metroBias”, “star”, “immediate”, “hop2”, “consExempt”, “force”, “symForce”.

Parameters

- **storage_group_id** – storage group id – str
- **srdf_group_number** – srdf group number – int
- **suspend_options** – suspend parameters – dict
- **_async** – if call should be async – bool

Returns storage group rdf details – dict

suspend_storagegroup_srdf(***kwargs*)

unlink_gen_snapshot(*sg_id*, *snap_name*, *unlink_sg_name*, *_async=False*, *gen_num=0*)
Unlink a snapshot from another storage group.

Parameters

- **sg_id** – storage group id – str
- **snap_name** – snapshot name – str
- **unlink_sg_name** – target storage group name – str
- **_async** – if call should be async – bool
- **gen_num** – snapshot generation number – int

Returns snapshot details – dict

10.1.7 PyU4V.rest_requests

rest_requests.py.

class PyU4V.rest_requests.**RestRequests**(*username*, *password*, *verify*, *base_url*, *interval*, *retries*, *application_type=None*)

Bases: object

RestRequests.

close_session()
Close the current session.

establish_rest_session()
Establish a REST session.

Returns session – object

rest_request(*target_url*, *method*, *params=None*, *request_object=None*, *timeout=None*)
Send a request to the target api.

Valid methods are ‘GET’, ‘POST’, ‘PUT’, ‘DELETE’.

Parameters

- **target_url** – target url –str
- **method** – method – str
- **params** – Additional URL parameters – dict
- **request_object** – request payload – dict

- **timeout** – optional timeout override – int

Returns server response, status code – dict, int

10.1.8 PyU4V.system

system.py.

class PyU4V.system.**SystemFunctions**(*array_id*, *rest_client*)

Bases: object

SystemFunctions.

delete_health_check(*health_check_id*, *array_id=None*)

Delete a health check record.

Parameters

- **health_check_id** – health check id – str
- **array_id** – array id – str

get_disk_details(*disk_id*, *array_id=None*)

Get details for specified disk id.

Parameters

- **disk_id** – disk id – str
- **array_id** – array id – str

Returns disk details – dict

get_disk_id_list(*array_id=None*, *failed=False*)

Get a list of disks ids installed.

Parameters

- **array_id** – array id – str
- **failed** – if only failed disks should be returned – bool

Returns disk ids – list

get_health_check_details(*health_check_id*, *array_id=None*)

Gets details of individual health check.

Parameters

- **health_check_id** – health check id – str
- **array_id** – array id – str

Returns health check details – dict

get_system_health(*array_id=None*)

Query for system health information.

Parameters **array_id** – array id – str

Returns system health – dict

get_tagged_objects(*tag_name*)

Get a list of objects with specified tag.

Parameters **tag_name** – tag name – str

Returns tags – list

get_tags(*array_id=None*, *tag_name=None*, *storage_group_id=None*, *num_of_storage_groups=None*, *num_of_arrays=None*)
Query for a list of tag names.

The input parameters represent optional filters for the tag query, including any filters will apply that filter to the list of returned tags.

Parameters

- **array_id** – filter by array id – str
- **tag_name** – filter by tag name – str
- **storage_group_id** – filter by storage group id – str
- **num_of_storage_groups** – filter by tags that are in x or greater amount of storage groups – int
- **num_of_arrays** – filter by tags that in y or greater amount of arrays – int

Returns tags – list

list_system_health_check(*array_id=None*)
List previously run system health checks.

Parameters **array_id** – array id – str

Returns system health checks – list

perform_health_check(*array_id=None*, *description=None*)
Initiate a environmental health check.

Parameters

- **array_id** – array id – str
- **description** – description for health check, if not set this will default to 'PyU4V-array_id-date-time'

Returns health check property details – dict

10.1.9 PyU4V.workload_planner

workload_planner.py.

class PyU4V.workload_planner.**WLPFunctions**(*array_id*, *rest_client*)
Bases: object

get_headroom(*array_id*, *workload=None*, *srp=None*, *slo=None*)
Get the Remaining Headroom Capacity.

Get the headroom capacity for a given srp/ slo/ workload combination.

Parameters

- **array_id** – array id – str
- **workload** – the workload type – str
- **srp** – storage resource pool id – str
- **slo** – service level id – str

Returns headroom details – dict

get_wlp_information(*array_id*)
Get the latest timestamp from WLP for processing New Workloads.

Parameters **array_id** – array id – str

Returns wlp details – dict

10.1.10 PyU4V.utils

PyU4V.utils

PyU4V.utils.config_handler

config_handler.py.

PyU4V.utils.config_handler.**set_logger_and_config**(*file_path=None*)
Set logger and config file.

Parameters **file_path** – path to PyU4V configuration file – str

Returns config parser – obj

PyU4V.utils.console

console.py

PyU4V.utils.constants

constants.py.

PyU4V.utils.decorators

decorators.py

PyU4V.utils.decorators.**deprecation_notice**(*func_class*, *start_version*, *end_version*)
Notify the user of function deprecation in a future version.

Parameters

- **func_class** – the class name for deprecated function – str
- **start_version** – the start Unisphere version – float
- **end_version** – the end Unisphere version – float

Returns decorated function – decorator

PyU4V.utils.decorators.**refactoring_notice**(*func_class*, *path*, *start_version*, *end_version*)
Notify the user of function refactoring elsewhere in PyU4V.

Parameters

- **func_class** – the class name for deprecated function – str
- **path** – path to the new function – str
- **start_version** – the start Unisphere version – float

- **end_version** – the end Unisphere version – float

Returns decorated function – decorator

PyU4V.utils.decorators.**retry**(*exceptions*, *total_attempts=3*, *delay=3*, *backoff=2*)
Decorator for retrying function calls.

Parameters

- **exceptions** – expected exceptions to retry on – class or tuple
- **total_attempts** – times to run function before failure. – int
- **delay** – initial delay between retries in seconds. – int
- **backoff** – delay multiplier between each attempt – int

PyU4V.utils.exception

exception.py

exception PyU4V.utils.exception.**InvalidInputException**(*message=None*, ***kwargs*)
Bases: *PyU4V.utils.exception.PyU4VException*

InvalidInputException.

message = 'Invalid input received: %(data)s'

exception PyU4V.utils.exception.**MissingConfigurationException**(*message=None*, ***kwargs*)
Bases: *PyU4V.utils.exception.PyU4VException*

MissingConfigurationFileException

message = 'PyU4V settings not be loaded, please check file location or univmax_conn in

exception PyU4V.utils.exception.**PyU4VException**(*message=None*, ***kwargs*)
Bases: exceptions.Exception

PyU4VException.

code = 500

headers = {}

message = 'An unknown exception occurred.'

safe = False

exception PyU4V.utils.exception.**ResourceNotFoundException**(*message=None*, ***kwargs*)
Bases: *PyU4V.utils.exception.PyU4VException*

ResourceNotFoundException.

message = 'The requested resource was not found: %(data)s'

exception PyU4V.utils.exception.**UnauthorizedRequestException**(*message=None*, ***kwargs*)
Bases: *PyU4V.utils.exception.PyU4VException*

UnauthorizedRequestException.

message = 'Unauthorized request - please check credentials'

exception PyU4V.utils.exception.**VolumeBackendAPIException**(*message=None*, ***kwargs*)

Bases: *PyU4V.utils.exception.PyU4VException*

VolumeBackendAPIException.

message = 'Bad or unexpected response from the storage volume backend API: %(data)s'

PyU4V.utils.file_handler

file_handler.py

PyU4V.utils.file_handler.**create_list_from_file**(*file_name*)

Given a file, create a list from its contents.

Parameters **file_name** – the path to the file – str

Returns file contents – list

PyU4V.utils.file_handler.**read_csv_values**(*file_name*, *convert=False*)

Read any csv file with headers.

You can extract the multiple lists from the headers in the CSV file. In your own script, call this function and assign to data variable, then extract the lists to the variables. Example: data = fh.read_csv_values(mycsv.csv) sg_name_list = data['sgname'] policy_list = data['policy']

Parameters

- **file_name** – path to CSV file – str
- **convert** – convert strings to equivalent data type – bool

Returns CSV parsed data – dict

PyU4V.utils.file_handler.**write_dict_to_csv_file**(*file_path*, *dictionary*)

Write dictionary data to CSV spreadsheet.

Parameters

- **file_path** – path including name of the file to be written to – str
- **dictionary** – data to be written to file – dict

PyU4V.utils.file_handler.**write_to_csv_file**(*file_name*, *data*)

Write list data to CSV spreadsheet.

Parameters

- **file_name** – name of the file to be written to – str
- **data** – data to be written to file – list

__init__.py.

CHAPTER 11

API Index

CHAPTER 12

Welcome to PyU4V's documentation!

12.1 Overview

PyU4V is a Python module that simplifies interaction with the Unisphere for PowerMax REST API. It wraps REST calls with simple APIs that abstract the HTTP request and response handling.

Note: You can get the Unisphere for PowerMax REST documentation by navigating to a URL in your local instance of Unisphere for PowerMax. Navigate to `https://{ip}:{port}/univmax/restapi/docs` where `{ip}` is the IP address of your Unisphere server and `{port}` is the port it is listening on. A zip file will be downloaded to your computer containing complete Unisphere REST endpoint documentation.

12.2 Supported PyU4V Versions

PyU4V Version	9.1.0.0
Unisphere Version	9.1.0.5
Array Model	VMAX-3, VMAX AFA, PowerMax
Array uCode	HyperMax OS, PowerMax OS
Platforms	Linux, Windows
Python	3.6, 3.7
Requirements	Requests, Six, urllib3
Test Requirements	TestTools, Tox

Note: If you want to continue to use Unisphere 8.4.x or 9.0.x with PyU4V you will need to remain on PyU4V 3.1.x. There is no support for PyU4V 9.1 with any version of Unisphere older than 9.1.x

Note: PyU4V officially supports Python 3.6 & 3.7, Python 2.x support has been dropped as it will soon be retired.

Note: PyU4V version 9.1.x is compatible with scripts written for PyU4V versions >= 3.x, there is **zero** support or compatibility for PyU4V 2.x or earlier scripts in later versions of PyU4V. If you have scripts written which specifically target Unisphere REST 8.4 or 9.0 endpoints these are still accessible via PyU4V 9.1.x however you will need to ensure you are passing the version required when performing these calls as PyU4V 9.1 will default to using 9.1 endpoints exclusively. You will also need to pay special attention to any REST JSON payloads in custom scripts as payloads are subject to change between major Unisphere REST releases.

12.3 Getting Started

Installation Guide How to get the source code, and how to build or install the python package.

Configuration Configuring PyU4V for your environment.

Quick Start Guide Making your first calls with PyU4V.

Contribute to PyU4V Contribute to the PyU4V project.

Issues & Support How to get support with or open issues for PyU4V.

Programmers Guide A range of examples demonstrating various PyU4V module usage.

API Glossary A glossary of all available functions.

12.4 Build your own PyU4V Docs

PyU4V docs have been built using Sphinx and included with the source PyU4V package, however if you would like to build the docs from scratch use the following commands:

```
$ pip install sphinx
$ pip install sphinx-rtd-theme
$ cd PyU4V/docs
$ make clean && make html
```

All of the necessary make files and sphinx configuration files are included with PyU4V so you can build the docs after the required dependencies have been installed.

Once the above commands have been run you will find newly generated html files within the `/PyU4V/docs/build/html` folder. Open `index.html` within a browser of your choosing to view the docs offline. Generating the docs is not required, we have bundled the most up-to-date docs with PyU4V so you can still navigate to `/PyU4V/docs/build/html/index.html` within your browser to view PyU4V docs offline.

12.5 Disclaimer

PyU4V 9.1 is distributed under the Apache 2.0 License. Unless required by applicable law or agreed to in writing, software distributed under the Apache 2.0 License is distributed on an **"as is" basis, without warranties or conditions of any kind**, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Python Module Index

p

Index

A

C

D

E

F

G

H

I

L

M

P

R

S

U

V

W