Sushy Documentation

Release 4.3.5.dev1

OpenStack Foundation

May 06, 2024

CONTENTS

1 Overview 1

2 Features 3

3 Documentation 5

Python Module Index 23

Index 25

CHAPTER

ONE

OVERVIEW

Sushy is a Python library to communicate with Redfish based systems.

The goal of the library is to be extremely simple, small, have as few dependencies as possible and be very conservative when dealing with BMCs by issuing just enough requests to it (BMCs are very flaky).

Therefore, the scope of the library has been limited to what is supported by the OpenStack Ironic project. As the project grows and more features from Redfish are needed we can expand Sushy to fulfill those requirements.

- Free software: Apache license
- **Includes Redfish registry files licensed under** Creative Commons Attribution 4.0 License: https://creativecommons.org/licenses/by/4.0/
- Documentation: https://docs.openstack.org/sushy/latest/
- Usage: https://docs.openstack.org/sushy/latest/reference/usage.html
- Source: https://opendev.org/openstack/sushy
- Bugs: https://storyboard.openstack.org/#!/project/960

CHAPTER TWO

FEATURES

- Abstraction around the SystemCollection and System resources (Basic server identification and asset information)
- RAID in Redfish based Systems
- Redfish Ethernet Interface
- System mappings
- System processor
- Storage management
- Systems power management (Both soft and hard; Including NMI injection)
- Changing systems boot device, frequency (Once or permanently) and mode (UEFI or BIOS)
- Chassis management
- OEM extention
- Virtual media management
- Session Management

CHAPTER THREE

DOCUMENTATION

3.1 Installing Sushy

At the command line:

```
$ pip install sushy
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv sushy
$ pip install sushy
```

3.2 Contributing to Sushy

3.2.1 How to contribute

If you would like to contribute to the development of OpenStack, you must follow the steps in this page:

> http://docs.openstack.org/infra/manual/developers.html

If you already have a good understanding of how the system works and your OpenStack accounts are set up, you can skip to the development workflow section of this documentation to learn how changes to OpenStack should be submitted for review via the Gerrit tool:

> http://docs.openstack.org/infra/manual/developers.html#development-workflow

Pull requests submitted through GitHub will be ignored.

Bugs should be filed in StoryBoard, not GitHub:

> https://storyboard.openstack.org/#!/project/960

3.2.2 Running a Redfish emulator

Testing and/or developing Sushy without owning a real baremetal machine that supports the Redfish protocol is possible by running an emulator, the sushy-tools project ships with two emulators that can be used for this purpose. To install it run:

```
sudo pip install --user sushy-tools
```

Note: Installing the dependencies requires libvirt development files. For example, run the following command to install them on Fedora:

```
sudo dnf install -y libvirt-devel
```

Static emulator

After installing sushy-tools you will have a new CLI tool named `sushy-static`. This tool creates a HTTP server to serve any of the Redfish mockups. The files are static so operations like changing the boot device or the power state **will not** have any effect. But that should be enough for enabling people to test parts of the library.

To use `sushy-static` we need the Redfish mockup files that can be downloaded from https://www.dmtf.org/standards/redfish, for example:

```
wget https://www.dmtf.org/sites/default/files/standards/documents/DSP2043_
↪1.0.0.zip
```

After the download, extract the files somewhere in the file-system:

```
unzip DSP2043_1.0.0.zip -d <output-path>
```

Now run `sushy-static` pointing to those files. For example to serve the `DSP2043-server` mockup files, run:

```
sushy-static --mockup-files <output-path>/DSP2043-server
```

Libvirt emulator

The second emulator shipped by sushy-tools is the CLI tool named `sushy-emulator`. This tool starts a ReST API that users can use to interact with virtual machines using the Redfish protocol. So operations such as changing the boot device or the power state will actually affect the virtual machines. This allows users to test the library in a more dynamic way. To run it do

```
sushy-emulator

# Or, running with custom parameters
sushy-emulator --port 8000 --libvirt-uri "qemu:///system"
```

That's it, now you can test Sushy against the `http://locahost:8000` endpoint.

Enabling SSL

Both mockup servers supports SSL if you want Sushy with it. To set it up, first you need to generate key and certificate files with OpenSSL use following command:

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
```

Start the mockup server passing the `--ssl-certificate` and `--ssl-key` parameters to it, for example:

```
sushy-emulator --ssl-key key.pem --ssl-certificate cert.pem
```

Now to connect with SSL to the server use the `verify` parameter pointing to the certificate file when instantiating Sushy, for example:

```
import sushy

# Note the HTTP"S"
s = sushy.Sushy('https://localhost:8000', verify='cert.pem', username='foo
↪', password='bar')
```

3.3 Sushy Library Reference

3.3.1 Usage

Using Sushy

To use sushy in a project:

Specifying an authentication type

There are three authentication objects. By default we use SessionOrBasicAuth.

Authentication Modes:

- auth.SessionOrBasicAuth: Use session based authentication. If we are unable to create a session we will fallback to basic authentication.
- auth.BasicAuth: Use basic authentication only.
- auth.SessionAuth: Use session based authentication only.

```
import logging

import sushy
from sushy import auth

# Enable logging at DEBUG level
LOG = logging.getLogger('sushy')
LOG.setLevel(logging.DEBUG)
LOG.addHandler(logging.StreamHandler())
```

(continues on next page)

(continued from previous page)

```
basic_auth = auth.BasicAuth(username='foo', password='bar')
session_auth = auth.SessionAuth(username='foo', password='bar')
session_or_basic_auth = auth.SessionOrBasicAuth(username='foo',
                                                password='bar')

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                auth=basic_auth)

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                auth=session_auth)

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                auth=session_or_basic_auth)

# It is important to note that you can
# call sushy without supplying an
# authentication object. In that case we
# will use the SessionOrBasicAuth authentication
# object in an attempt to connect to all different
# types of redfish servers.
s = sushy.Sushy('http://localhost:8000/redfish/v1',
                username='foo',
                password='bar')
```

Creating and using a sushy system object

```
import logging

import sushy

# Enable logging at DEBUG level
LOG = logging.getLogger('sushy')
LOG.setLevel(logging.DEBUG)
LOG.addHandler(logging.StreamHandler())

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                username='foo', password='bar')

# Get the Redfish version
print(s.redfish_version)

# Instantiate a system object
sys_inst = s.get_system('/redfish/v1/Systems/437XR1138R2')


# Using system collections


# Instantiate a SystemCollection object
sys_col = s.get_system_collection()

# Print the ID of the systems available in the collection
print(sys_col.members_identities)
```

(continues on next page)

(continued from previous page)

```
# Get a list of systems objects available in the collection
sys_col_insts = sys_col.get_members()

# Instantiate a system object, same as getting it directly
# from the s.get_system()
sys_inst = sys_col.get_member(sys_col.members_identities[0])

# Refresh the system collection object
#
# See below for more options on how to refresh resources.
sys_col.refresh()


# Using system actions


# Power the system ON
sys_inst.reset_system(sushy.ResetType.ON)

# Get a list of allowed reset values
print(sys_inst.get_allowed_reset_system_values())

# Refresh the system object (with all its sub-resources)
sys_inst.refresh()

# Alternatively, you can only refresh the resource if it is stale by␣
↪passing
# force=False:
sys_inst.refresh(force=False)

# A resource can be marked stale by calling invalidate. Note that its
# subresources won't be marked as stale, and thus they won't be refreshed␣
↪by
# a call to refresh(force=False)
sys_inst.invalidate()

# Get the current power state
print(sys_inst.power_state)

# Set the next boot device to boot once from PXE in UEFI mode
sys_inst.set_system_boot_source(sushy.BootSource.PXE,
                                enabled=sushy.BootSourceOverrideEnabled.
↪ONCE,
                                mode=sushy.BootSourceOverrideMode.UEFI)

# Get the current boot source information
print(sys_inst.boot)

# Get a list of allowed boot source target values
print(sys_inst.get_allowed_system_boot_source_values())

# Get the memory summary
print(sys_inst.memory_summary)

# Get the processor summary
```

(continues on next page)

(continued from previous page)

```
print(sys_inst.processors.summary)
```

Creating and using a sushy manager object

```
import logging

import sushy

# Enable logging at DEBUG level
LOG = logging.getLogger('sushy')
LOG.setLevel(logging.DEBUG)
LOG.addHandler(logging.StreamHandler())

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                username='foo', password='bar')

# Instantiate a manager object
mgr_inst = s.get_manager('BMC')

# Get the manager name & description
print(mgr_inst.name)
print(mgr_inst.description)


# Using manager collections


# Instantiate a ManagerCollection object
mgr_col = s.get_manager_collection()

# Print the ID of the managers available in the collection
print(mgr_col.members_identities)

# Get a list of manager objects available in the collection
mgr_insts = mgr_col.get_members()

# Instantiate a manager object, same as getting it directly
# from the s.get_manager()
mgr_inst = mgr_col.get_member(mgr_col.members_identities[0])

# Refresh the manager collection object
mgr_col.invalidate()
mgr_col.refresh()


# Using manager actions


# Get supported graphical console types
print(mgr_inst.get_supported_graphical_console_types())

# Get supported serial console types
print(mgr_inst.get_supported_serial_console_types())
```

(continues on next page)

(continued from previous page)

```
# Get supported command shell types
print(mgr_inst.get_supported_command_shell_types())

# Get a list of allowed manager reset values
print(mgr_inst.get_allowed_reset_manager_values())

# Reset the manager
mgr_inst.reset_manager(sushy.ResetType.FORCE_RESTART)

# Refresh the manager object (with all its sub-resources)
mgr_inst.refresh(force=True)


# Using Virtual Media

# Instantiate a VirtualMediaCollection object
virtmedia_col = mgr_inst.virtual_media

# Print the ID of the VirtualMedia available in the collection
print(virtmedia_col.members_identities)

# Get a list of VirtualMedia objects available in the collection
virtmedia_insts = virtmedia_col.get_members()

# Instantiate a VirtualMedia object
virtmedia_inst = virtmedia_col.get_member(
    virtmedia_col.members_identities[0])


# Print out some of the VirtualMedia properties
print(virtmedia_inst.name,
      virtmedia_inst.media_types)

# Insert virtual media (invalidates virtmedia_inst contents)
virtmedia_inst.insert_media('https://www.dmtf.org/freeImages/Sardine.img')

# Refresh the resource to load actual contents
virtmedia_inst.refresh()

# Print out some of the VirtualMedia properties
print(virtmedia_inst.image,
      virtmedia_inst.image_path,
      virtmedia_inst.inserted,
      virtmedia_inst.write_protected)

# ... Boot the system off the virtual media...

# Eject virtual media (invalidates virtmedia_inst contents)
virtmedia_inst.eject_media()
```

Creating and using a sushy client with Sessions

```
import logging

import sushy

# Enable logging at DEBUG level
LOG = logging.getLogger('sushy')
LOG.setLevel(logging.DEBUG)
LOG.addHandler(logging.StreamHandler())

s = sushy.Sushy('http://localhost:8000/redfish/v1',
                username='foo', password='bar')

# Get the ComputerSystem object (if there is only one), otherwise
# the identity must be provided as a path to the system.
system = s.get_system()

# A session is created automatically for you.
# Print the boot field in the ComputerSystem.
print(system.boot)

# Upon session timeout, Sushy recreates the session based upon
# provided credentials. If this fails, an exception is raised.

# Explicitly request a session_key and session_uri.
# This is not stored, but may be useful.
session_key, session_uri = s.create_session(username='foo',
                                            password='bar')

# Retrieve the session
session = s.get_session(session_uri)

# Delete the session
session.delete()
```

Using OEM extensions

Before running this example, please make sure you have a Redfish BMC that includes the OEM piece for a specific vendor, as well as the Sushy OEM extension package installed in the system for the same vendor.

You can check the presence of the OEM extension within each Redfish resource by specifying the vendor ID and search for them.

In the following example, we are looking up “Acme” vendor extension to Redfish Manager resource.

```
import sushy

root = sushy.Sushy('http://localhost:8000/redfish/v1')

# Instantiate a system object
system = root.get_system('/redfish/v1/Systems/437XR1138R2')

print('Working on system resource %s' % system.identity)
```

(continues on next page)

(continued from previous page)

```
for manager in system.managers:

    print('Using System manager %s' % manager.identity)

    # Get a list of OEM extension names for the system manager
    oem_vendors = manager.oem_vendors

    print('Listing OEM extension name(s) for the System '
          'manager %s' % manager.identity )

    print(*oem_vendors, sep="\n")

    try:
        manager_oem = manager.get_oem_extension('Acme')

    except sushy.exceptions.OEMExtensionNotFoundError:
        print('ERROR: Acme OEM extension not found in '
              'Manager %s' % manager.identity)
        continue

    print('%s is an OEM extension of Manager %s'
          % (manager_oem.get_extension(), manager.identity))

    # set boot device to a virtual media device image
    manager_oem.set_virtual_boot_device(sushy.VirtualMediaType.CD,
                                        manager=manager)
```

If you do not have any real baremetal machine that supports the Redfish protocol you can look at the *Contributing to Sushy* page to learn how to run a Redfish emulator.

For the OEM extension example, presently, both of the emulators (static/dynamic) do not expose any OEM; as a result, users may need to add manually some OEM resources to emulators' templates. It may be easier to start with a static emulator.

3.3.2 Sushy Python API Reference

- modindex

sushy

sushy package

Subpackages

sushy.resources package

Subpackages

sushy.resources.certificateservice package

Submodules

sushy.resources.certificateservice.certificate module

sushy.resources.certificateservice.certificateservice module

sushy.resources.certificateservice.constants module

Module contents

sushy.resources.chassis package

Subpackages

sushy.resources.chassis.power package

Submodules

sushy.resources.chassis.power.constants module

sushy.resources.chassis.power.power module

Module contents

sushy.resources.chassis.thermal package

Submodules

sushy.resources.chassis.thermal.constants module

sushy.resources.chassis.thermal.thermal module

Module contents

Submodules

sushy.resources.chassis.chassis module

sushy.resources.chassis.constants module

Module contents

sushy.resources.compositionservice package

Submodules

sushy.resources.compositionservice.compositionservice module

sushy.resources.compositionservice.constants module

sushy.resources.compositionservice.resourceblock module

sushy.resources.compositionservice.resourcezone module

Module contents

sushy.resources.eventservice package

Submodules

sushy.resources.eventservice.constants module

sushy.resources.eventservice.eventdestination module

sushy.resources.eventservice.eventservice module

Module contents

sushy.resources.fabric package

Submodules

sushy.resources.fabric.constants module

sushy.resources.fabric.endpoint module

sushy.resources.fabric.fabric module

Module contents

sushy.resources.manager package

Submodules

sushy.resources.manager.constants module

sushy.resources.manager.manager module

sushy.resources.manager.virtual_media module

Module contents

sushy.resources.oem package

Submodules

sushy.resources.oem.base module

sushy.resources.oem.common module

sushy.resources.oem.fake module

Module contents

sushy.resources.registry package

Submodules

sushy.resources.registry.attribute_registry module

sushy.resources.registry.constants module

sushy.resources.registry.message_registry module

sushy.resources.registry.message_registry_file module

Module contents

sushy.resources.sessionservice package

Submodules

sushy.resources.sessionservice.session module

sushy.resources.sessionservice.sessionservice module

Module contents

sushy.resources.system package

Subpackages

sushy.resources.system.network package

Submodules

sushy.resources.system.network.adapter module

sushy.resources.system.network.constants module

sushy.resources.system.network.device_function module

sushy.resources.system.network.port module

Module contents

sushy.resources.system.storage package

Submodules

sushy.resources.system.storage.constants module

sushy.resources.system.storage.controller module

sushy.resources.system.storage.drive module

sushy.resources.system.storage.storage module

sushy.resources.system.storage.volume module

Module contents

Submodules

sushy.resources.system.bios module

sushy.resources.system.constants module

sushy.resources.system.ethernet_interface module

sushy.resources.system.processor module

sushy.resources.system.secure_boot module

sushy.resources.system.secure_boot_database module

sushy.resources.system.simple_storage module

sushy.resources.system.system module

Module contents

sushy.resources.taskservice package

Submodules

sushy.resources.taskservice.constants module

sushy.resources.taskservice.task module

sushy.resources.taskservice.taskservice module

Module contents

sushy.resources.updateservice package

Submodules

sushy.resources.updateservice.constants module

sushy.resources.updateservice.softwareinventory module

sushy.resources.updateservice.updateservice module

Module contents

Submodules

sushy.resources.base module

sushy.resources.common module

sushy.resources.constants module

sushy.resources.ipaddresses module

sushy.resources.settings module

Module contents

Submodules

sushy.auth module

sushy.connector module

sushy.exceptions module

exception sushy.exceptions.**AccessError**(*method*, *url*, *response*)

Bases: *sushy.exceptions.HTTPError*

exception sushy.exceptions.**ArchiveParsingError**(*message=None*, ***kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'Failed parsing archive "%(path)s": %(error)s'

exception sushy.exceptions.**BadRequestError**(*method*, *url*, *response*)

Bases: *sushy.exceptions.HTTPError*

exception sushy.exceptions.**ConnectionError**(*message=None*, ***kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'Unable to connect to %(url)s. Error: %(error)s'

exception sushy.exceptions.**ExtensionError**(*message=None*, ***kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'Sushy Extension Error: %(error)s'

exception sushy.exceptions.**HTTPError**(*method*, *url*, *response*)

Bases: *sushy.exceptions.SushyError*

Basic exception for HTTP errors

body = None

Error JSON body, if present.

code = 'Base.1.0.GeneralError'

Error code defined in the Redfish specification, if present.

detail = None

Error message defined in the Redfish specification, if present.

extended_info = None

Extended information provided in the response.

message = 'HTTP %(method)s %(url)s returned code %(code)s. %(error)s Extended information: %(ext_info)s'

property related_properties

List of properties related to the error.

status_code = None

HTTP status code.

exception sushy.exceptions.**InvalidParameterValueError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'The parameter "%(parameter)s" value "%(value)s" is invalid. Valid values are: %(valid_values)s'

exception sushy.exceptions.**MalformedAttributeError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'The attribute %(attribute)s is malformed in the resource %(resource)s: %(error)s'

exception sushy.exceptions.**MissingActionError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'The action %(action)s is missing from the resource %(resource)s'

exception sushy.exceptions.**MissingAttributeError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'The attribute %(attribute)s is missing from the resource %(resource)s'

exception sushy.exceptions.**MissingHeaderError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'Response to %(target_uri)s did not contain a %(header)s header'

exception sushy.exceptions.**MissingXAuthToken**(*method, url, response*)

Bases: *sushy.exceptions.HTTPError*

message = 'No X-Auth-Token returned from remote host when attempting to establish a session. Error: %(error)s'

exception sushy.exceptions.**OEMExtensionNotFoundError**(*message=None, **kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'No %(resource)s OEM extension found by name "%(name)s".'

exception sushy.exceptions.**ResourceNotFoundError**(*method, url, response*)

Bases: *sushy.exceptions.HTTPError*

message = 'Resource %(url)s not found'

exception sushy.exceptions.**ServerSideError**(*method, url, response*)

Bases: *sushy.exceptions.HTTPError*

exception sushy.exceptions.**SushyError**(*message=None, **kwargs*)

Bases: Exception

Basic exception for errors raised by Sushy

message = None

exception sushy.exceptions.**UnknownDefaultError**(*message=None*, ***kwargs*)

Bases: *sushy.exceptions.SushyError*

message = 'Failed at determining default for "%(entity)s": %(error)s'

sushy.exceptions.**raise_for_response**(*method*, *url*, *response*)

Raise a correct error class, if needed.

sushy.main module

sushy.taskmonitor module

sushy.utils module

Module contents

- genindex

PYTHON MODULE INDEX

S

sushy.exceptions, 19

INDEX

A

AccessError, 19
ArchiveParsingError, 19

B

BadRequestError, 19
body (*sushy.exceptions.HTTPError attribute*), 19

C

code (*sushy.exceptions.HTTPError attribute*), 19
ConnectionError, 19

D

detail (*sushy.exceptions.HTTPError attribute*), 19

E

extended_info (*sushy.exceptions.HTTPError attribute*), 19
ExtensionError, 19

H

HTTPError, 19

I

InvalidParameterValueError, 19

M

MalformedAttributeError, 20
message (*sushy.exceptions.ArchiveParsingError attribute*), 19
message (*sushy.exceptions.ConnectionError attribute*), 19
message (*sushy.exceptions.ExtensionError attribute*), 19
message (*sushy.exceptions.HTTPError attribute*), 19
message (*sushy.exceptions.InvalidParameterValueError attribute*), 20
message (*sushy.exceptions.MalformedAttributeError attribute*), 20
message (*sushy.exceptions.MissingActionError attribute*), 20
message (*sushy.exceptions.MissingAttributeError attribute*), 20
message (*sushy.exceptions.MissingHeaderError attribute*), 20
message (*sushy.exceptions.MissingXAuthToken attribute*), 20
message (*sushy.exceptions.OEMExtensionNotFoundError attribute*), 20
message (*sushy.exceptions.ResourceNotFoundError attribute*), 20
message (*sushy.exceptions.SushyError attribute*), 20
message (*sushy.exceptions.UnknownDefaultError attribute*), 21
MissingActionError, 20
MissingAttributeError, 20
MissingHeaderError, 20
MissingXAuthToken, 20
module
    sushy.exceptions, 19

O

OEMExtensionNotFoundError, 20

R

raise_for_response() (*in module sushy.exceptions*), 21
related_properties (*sushy.exceptions.HTTPError property*), 19
ResourceNotFoundError, 20

S

ServerSideError, 20
status_code (*sushy.exceptions.HTTPError attribute*), 19
sushy.exceptions
    module, 19
SushyError, 20

U

UnknownDefaultError, 21