Octavia Library Documentation

Release 3.1.1.dev3

OpenStack Octavia Team

Sep 26, 2023

CONTENTS

`openstack` `community project` `cii best practices` `passing`

A library to support Octavia provider drivers.

This python module provides a python library for Octavia provider driver developers.

See the provider driver development guide for more information:

> https://docs.openstack.org/octavia/latest/contributor/guides/providers.html

Octavia-lib is distributed under the terms of the Apache License, Version 2.0. The full terms and conditions of this license are detailed in the LICENSE file.

- Free software: Apache license

- Documentation: https://docs.openstack.org/octavia-lib/latest

- Source: https://opendev.org/openstack/octavia-lib

- Bugs: https://storyboard.openstack.org/#!/project/openstack/octavia-lib

`openstack` `community project` `cii best practices` `passing`

INSTALLATION

At the command line:

```
$ pip install octavia-lib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv octavia-lib
$ pip install octavia-lib
```

USAGE

Instructions for using the library are provided in the Provider Driver Development Guide.

CONTRIBUTOR DOCUMENTATION

3.1 Contributor Guidelines

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 on Storyboard

https://storyboard.openstack.org/#!/project/openstack/octavia-lib

CONFIGURATION

The octavia-lib library does not currently use any oslo configuration settings.

COMMAND LINE INTERFACE REFERENCE

The octavia-lib library does not provide a CLI.

See the python-octaviaclient for the Octavia CLI:

https://docs.openstack.org/python-octaviaclient/latest/

USERS GUIDE

The octavia-lib is a library supporting Octavia provider drivers.

Instructions for using the library are provided in the Provider Driver Development Guide.

ADMINISTRATORS GUIDE

The octavia-lib is a library supporting Octavia provider drivers.

There are no administrator tasks or settings for octavia-lib.

Information on administrating Octavia deployments is available in the Octavia Administration Guide.

REFERENCES

8.1 Module Reference

8.1.1 octavia_lib

octavia_lib package

Subpackages

octavia_lib.api package

Subpackages

octavia_lib.api.drivers package

Submodules

octavia_lib.api.drivers.data_models module

class octavia_lib.api.drivers.data_models.**BaseDataModel**

　　Bases: object

　　classmethod from_dict(*dict*)

　　to_dict(*calling_classes=None*, *recurse=False*, *render_unsets=False*, ***kwargs*)

　　　　Converts a data model to a dictionary.

class octavia_lib.api.drivers.data_models.**HealthMonitor**(*admin_state_up=Unset*, *delay=Unset*, *expected_codes=Unset*, *healthmonitor_id=Unset*, *http_method=Unset*, *max_retries=Unset*, *max_retries_down=Unset*, *name=Unset*, *pool_id=Unset*, *timeout=Unset*, *type=Unset*, *url_path=Unset*, *http_version=Unset*, *domain_name=Unset*, *project_id=Unset*)

> Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**L7Policy**(*action=Unset*, *admin_state_up=Unset*, *description=Unset*, *l7policy_id=Unset*, *listener_id=Unset*, *name=Unset*, *position=Unset*, *redirect_pool_id=Unset*, *redirect_url=Unset*, *rules=Unset*, *redirect_prefix=Unset*, *redirect_http_code=Unset*, *project_id=Unset*)

> Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**L7Rule**(*admin_state_up=Unset*, *compare_type=Unset*, *invert=Unset*, *key=Unset*, *l7policy_id=Unset*, *l7rule_id=Unset*, *type=Unset*, *value=Unset*, *project_id=Unset*)

> Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**Listener**(*admin_state_up=Unset,*
connection_limit=Unset,
default_pool=Unset,
default_pool_id=Unset,
default_tls_container_ref=Unset,
default_tls_container_data=Unset,
description=Unset,
insert_headers=Unset,
l7policies=Unset,
listener_id=Unset,
loadbalancer_id=Unset,
name=Unset, protocol=Unset,
protocol_port=Unset,
sni_container_refs=Unset,
sni_container_data=Unset,
timeout_client_data=Unset,
timeout_member_connect=Unset,
timeout_member_data=Unset,
timeout_tcp_inspect=Unset,
client_ca_tls_container_ref=Unset,
client_ca_tls_container_data=Unset,
client_authentication=Unset,
client_crl_container_ref=Unset,
client_crl_container_data=Unset,
project_id=Unset,
allowed_cidrs=Unset,
tls_versions=Unset,
tls_ciphers=Unset,
alpn_protocols=Unset)

Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**LoadBalancer**(*admin_state_up=Unset,*
description=Unset,
flavor=Unset, listeners=Unset,
loadbalancer_id=Unset,
name=Unset, pools=Unset,
project_id=Unset,
vip_address=Unset,
vip_network_id=Unset,
vip_port_id=Unset,
vip_subnet_id=Unset,
vip_qos_policy_id=Unset,
additional_vips=Unset,
availability_zone=Unset)

Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**Member**(*address=Unset,*
admin_state_up=Unset,
member_id=Unset,
monitor_address=Unset,
monitor_port=Unset, name=Unset,
pool_id=Unset, protocol_port=Unset,
subnet_id=Unset, weight=Unset,
backup=Unset, project_id=Unset)

Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**Pool**(*admin_state_up=Unset,*
description=Unset,
healthmonitor=Unset,
lb_algorithm=Unset,
loadbalancer_id=Unset,
members=Unset, name=Unset,
pool_id=Unset, listener_id=Unset,
protocol=Unset,
session_persistence=Unset,
tls_container_ref=Unset,
tls_container_data=Unset,
ca_tls_container_ref=Unset,
ca_tls_container_data=Unset,
crl_container_ref=Unset,
crl_container_data=Unset,
tls_enabled=Unset, project_id=Unset,
tls_versions=Unset, tls_ciphers=Unset,
alpn_protocols=Unset)

Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

class octavia_lib.api.drivers.data_models.**UnsetType**

Bases: object

class octavia_lib.api.drivers.data_models.**VIP**(*vip_address=Unset,*
vip_network_id=Unset,
vip_port_id=Unset, vip_subnet_id=Unset,
vip_qos_policy_id=Unset)

Bases: *octavia_lib.api.drivers.data_models.BaseDataModel*

octavia_lib.api.drivers.driver_lib module

class octavia_lib.api.drivers.driver_lib.**DriverLibrary**(*status_socket='/var/run/octavia/status.sock',*
stats_socket='/var/run/octavia/stats.sock',
get_socket='/var/run/octavia/get.sock',
**kwargs)

Bases: object

get_healthmonitor(*healthmonitor_id*)

Get a health monitor object.

> **Parameters** `healthmonitor_id` (*UUID string*) -- The health monitor ID to
> lookup.

> **Raises**
>> • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.
>>
>> • *DriverError* -- An unexpected error occurred.
>
> **Returns** A HealthMonitor object or None if not found.

get_l7policy(*l7policy_id*)

> Get a L7 policy object.
>
>> **Parameters l7policy_id** (*UUID string*) -- The L7 policy ID to lookup.
>>
>> **Raises**
>>> • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.
>>>
>>> • *DriverError* -- An unexpected error occurred.
>>
>> **Returns** A L7Policy object or None if not found.

get_l7rule(*l7rule_id*)

> Get a L7 rule object.
>
>> **Parameters l7rule_id** (*UUID string*) -- The L7 rule ID to lookup.
>>
>> **Raises**
>>> • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.
>>>
>>> • *DriverError* -- An unexpected error occurred.
>>
>> **Returns** A L7Rule object or None if not found.

get_listener(*listener_id*)

> Get a listener object.
>
>> **Parameters listener_id** (*UUID string*) -- The listener ID to lookup.
>>
>> **Raises**
>>> • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.
>>>
>>> • *DriverError* -- An unexpected error occurred.
>>
>> **Returns** A Listener object or None if not found.

get_loadbalancer(*loadbalancer_id*)

> Get a load balancer object.
>
>> **Parameters loadbalancer_id** (*UUID string*) -- The load balancer ID to lookup.
>>
>> **Raises**
>>> • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.
>>>
>>> • *DriverError* -- An unexpected error occurred.

Returns A LoadBalancer object or None if not found.

get_member(*member_id*)

>   Get a member object.

>   **Parameters member_id** (`UUID string`) -- The member ID to lookup.

>   **Raises**

>   • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.

>   • *DriverError* -- An unexpected error occurred.

>   **Returns** A Member object or None if not found.

get_pool(*pool_id*)

>   Get a pool object.

>   **Parameters pool_id** (`UUID string`) -- The pool ID to lookup.

>   **Raises**

>   • *DriverAgentTimeout* -- The driver agent did not respond inside the time-out.

>   • *DriverError* -- An unexpected error occurred.

>   **Returns** A Pool object or None if not found.

update_listener_statistics(*statistics*)

>   Update listener statistics.

>   **Parameters statistics** (`dict`) -- Statistics for listeners: id (string): ID for listener. active_connections (int): Number of currently active connections. bytes_in (int): Total bytes received. bytes_out (int): Total bytes sent. request_errors (int): Total requests not fulfilled. total_connections (int): The total connections handled.

>   **Raises** UpdateStatisticsError

>   **Returns** None

update_loadbalancer_status(*status*)

>   Update load balancer status.

>   **Parameters status** (`dict`) -- dictionary defining the provisioning status and operating status for load balancer objects, including pools, members, listeners, L7 policies, and L7 rules. iod (string): ID for the object. provisioning_status (string): Provisioning status for the object. operating_status (string): Operating status for the object.

>   **Raises** UpdateStatusError

>   **Returns** None

octavia_lib.api.drivers.exceptions module

exception octavia_lib.api.drivers.exceptions.**DriverAgentNotFound**(**args*, ***kwargs*)

Bases: Exception

Exception raised when the driver agent cannot be reached.

Each exception will include a message field that describes the error. :param fault_string: String describing the fault. :type fault_string: string

fault_string = 'The driver-agent process was not found or not ready.'

exception octavia_lib.api.drivers.exceptions.**DriverAgentTimeout**(**args*, ***kwargs*)

Bases: Exception

Exception raised when the driver agent does not respond.

Raised when communication with the driver agent times out. Each exception will include a message field that describes the error. :param fault_string: String describing the fault. :type fault_string: string

fault_string = 'The driver-agent timeout.'

exception octavia_lib.api.drivers.exceptions.**DriverError**(**args*, ***kwargs*)

Bases: Exception

Catch all exception that drivers can raise.

This exception includes two strings: The user fault string and the optional operator fault string. The user fault string, "user_fault_string", will be provided to the API requester. The operator fault string, "operator_fault_string", will be logged in the Octavia API log file for the operator to use when debugging.

> **Parameters**
>
> - **user_fault_string** (*string*) -- String provided to the API requester.
>
> - **operator_fault_string** (*string*) -- Optional string logged by the Octavia API for the operator to use when debugging.

operator_fault_string = 'An unknown driver error occurred.'

user_fault_string = 'An unknown driver error occurred.'

exception octavia_lib.api.drivers.exceptions.**NotFound**(**args*, ***kwargs*)

Bases: Exception

Exception raised when the driver cannot find a resource.

This exception includes two strings: The user fault string and the optional operator fault string. The user fault string, "user_fault_string", will be provided to the API requester. The operator fault string, "operator_fault_string", will be logged in the Octavia API log file for the operator to use when debugging.

> **Parameters**
>
> - **user_fault_string** (*string*) -- String provided to the API requester.
>
> - **operator_fault_string** (*string*) -- Optional string logged by the Octavia API for the operator to use when debugging.

```
operator_fault_string = 'The provider driver could not find a resource.'

user_fault_string = 'The provider driver could not find a resource.'
```

exception octavia_lib.api.drivers.exceptions.**NotImplementedError**(*args*, ***kwargs*)

Bases: `Exception`

Exception raised when a driver does not implement an API function.

> **Parameters**
>
> - **user_fault_string** (`string`) -- String provided to the API requester.
>
> - **operator_fault_string** (`string`) -- Optional string logged by the Octavia API for the operator to use when debugging.

```
operator_fault_string = 'This feature is not implemented by this
provider.'

user_fault_string = 'This feature is not implemented by the provider.'
```

exception octavia_lib.api.drivers.exceptions.**UnsupportedOptionError**(*args*, ***kwargs*)

Bases: `Exception`

Exception raised when a driver does not support an option.

Provider drivers will validate that they can complete the request -- that all options are supported by the driver. If the request fails validation, drivers will raise an UnsupportedOptionError exception. For example, if a driver does not support a flavor passed as an option to load balancer create(), the driver will raise an UnsupportedOptionError and include a message parameter providing an explanation of the failure.

> **Parameters**
>
> - **user_fault_string** (`string`) -- String provided to the API requester.
>
> - **operator_fault_string** (`string`) -- Optional string logged by the Octavia API for the operator to use when debugging.

```
operator_fault_string = 'A specified option is not supported by this
provider.'

user_fault_string = 'A specified option is not supported by this
provider.'
```

exception octavia_lib.api.drivers.exceptions.**UpdateStatisticsError**(*args*, ***kwargs*)

Bases: `Exception`

Exception raised when a statistics update fails.

Each exception will include a message field that describes the error and references to the failed record if available. :param fault_string: String describing the fault. :type fault_string: string :param status_object: The object the fault occurred on. :type status_object: string :param status_object_id: The ID of the object that failed stats update. :type status_object_id: string :param status_record: The stats update record that caused the fault. :type status_record: string

---

```
fault_string = 'The statistics update had an unknown error.'

stats_object = None

stats_object_id = None

stats_record = None
```

exception octavia_lib.api.drivers.exceptions.**UpdateStatusError**(*args*, ***kwargs*)

Bases: `Exception`

Exception raised when a status update fails.

Each exception will include a message field that describes the error and references to the failed record if available. :param fault_string: String describing the fault. :type fault_string: string :param status_object: The object the fault occurred on. :type status_object: string :param status_object_id: The ID of the object that failed status update. :type status_object_id: string :param status_record: The status update record that caused the fault. :type status_record: string

```
fault_string = 'The status update had an unknown error.'

status_object = None

status_object_id = None

status_record = None
```

octavia_lib.api.drivers.provider_base module

class octavia_lib.api.drivers.provider_base.**ProviderDriver**

Bases: `object`

create_vip_port(*loadbalancer_id*, *project_id*, *vip_dictionary*, *additional_vip_dicts*)

Creates a port for a load balancer VIP.

If the driver supports creating VIP ports, the driver will create a VIP port with the primary VIP and all additional VIPs added to the port, and return the vip_dictionary populated with the vip_port_id and a list of vip_dictionaries populated with data from the additional VIPs (which are guaranteed to be in the same Network). This might look like: {'port_id': port_id, 'subnet_id': subnet_id_1, 'ip_address': ip1}, [{'subnet_id': subnet_id_2, 'ip_address': ip2}, {...}, {...}]] If the driver does not support port creation, the driver will raise a NotImplementedError.

Parameters

- **loadbalancer_id** (*string*) -- ID of loadbalancer.
- **project_id** (*string*) -- The project ID to create the VIP under.

Param vip_dictionary: The VIP dictionary.

Param additional_vip_dicts: A list of additional VIP dictionaries, with subnets guaranteed to be in the same network as the primary vip_dictionary.

Returns VIP dictionary with vip_port_id + a list of additional VIP dictionaries (vip_dict, additional_vip_dicts).

Raises

- *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- The driver does not support creating VIP ports.

get_supported_availability_zone_metadata()

> Returns a dict of supported availability zone metadata keys.
>
> The returned dictionary will include key/value pairs, 'name' and 'description.'
>
> > **Returns** The availability zone metadata dictionary
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- The driver does not support AZs.

get_supported_flavor_metadata()

> Returns a dict of flavor metadata keys supported by this driver.
>
> The returned dictionary will include key/value pairs, 'name' and 'description.'
>
> > **Returns** The flavor metadata dictionary
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- The driver does not support flavors.

health_monitor_create(*healthmonitor*)

> Creates a new health monitor.
>
> > **Parameters healthmonitor** (*object*) -- The health monitor object.
> >
> > **Returns** Nothing if the create request was accepted.
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- if driver does not support request.
> >
> > - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

health_monitor_delete(*healthmonitor*)

> Deletes a healthmonitor_id.
>
> > **Parameters healthmonitor** (*object*) -- The monitor to delete.
> >
> > **Returns** Nothing if the create request was accepted.
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- if driver does not support request.

health_monitor_update(*old_healthmonitor*, *new_healthmonitor*)

> Updates a health monitor.
>
> > **Parameters**
> >
> > - **old_healthmonitor** (*object*) -- The baseline health monitor object.

- **new_healthmonitor** (*object*) -- The updated health monitor object.

> **Returns** Nothing if the create request was accepted.

> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.
>
> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

l7policy_create(*l7policy*)

> Creates a new L7 policy.

> **Parameters l7policy** (*object*) -- The L7 policy object.

> **Returns** Nothing if the create request was accepted.

> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.
>
> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

l7policy_delete(*l7policy*)

> Deletes an L7 policy.

> **Parameters l7policy** (*object*) -- The L7 policy to delete.

> **Returns** Nothing if the delete request was accepted.

> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.

l7policy_update(*old_l7policy*, *new_l7policy*)

> Updates an L7 policy.

> **Parameters**
>
> - **old_l7policy** (*object*) -- The baseline L7 policy object.
>
> - **new_l7policy** (*object*) -- The updated L7 policy object.

> **Returns** Nothing if the update request was accepted.

> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.
>
> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

l7rule_create(*l7rule*)

    Creates a new L7 rule.

        Parameters `l7rule` (*object*) -- The L7 rule object.

        Returns Nothing if the create request was accepted.

        Raises

- *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- if driver does not support request.

- *UnsupportedOptionError* -- if driver does not support one of the configuration options.

l7rule_delete(*l7rule*)

    Deletes an L7 rule.

        Parameters `l7rule` (*object*) -- The L7 rule to delete.

        Returns Nothing if the delete request was accepted.

        Raises

- *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- if driver does not support request.

l7rule_update(*old_l7rule*, *new_l7rule*)

    Updates an L7 rule.

        Parameters

- `old_l7rule` (*object*) -- The baseline L7 rule object.

- `new_l7rule` (*object*) -- The updated L7 rule object.

        Returns Nothing if the update request was accepted.

        Raises

- *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- if driver does not support request.

- *UnsupportedOptionError* -- if driver does not support one of the configuration options.

listener_create(*listener*)

    Creates a new listener.

        Parameters `listener` (*object*) -- The listener object.

        Returns Nothing if the create request was accepted.

        Raises

- *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- if driver does not support request.

- *UnsupportedOptionError* -- if driver does not support one of the configuration options.

listener_delete(*listener*)

> Deletes a listener.
>
>> **Parameters** **listener** (`object`) -- The listener to delete.
>>
>> **Returns** Nothing if the delete request was accepted.
>>
>> **Raises**
>>
>>> - *DriverError* -- An unexpected error occurred in the driver.
>>>
>>> - *NotImplementedError* -- if driver does not support request.

listener_update(*old_listener*, *new_listener*)

> Updates a listener.
>
>> **Parameters**
>>
>>> - **old_listener** (`object`) -- The baseline listener object.
>>>
>>> - **new_listener** (`object`) -- The updated listener object.
>>
>> **Returns** Nothing if the update request was accepted.
>>
>> **Raises**
>>
>>> - *DriverError* -- An unexpected error occurred in the driver.
>>>
>>> - *NotImplementedError* -- if driver does not support request.
>>>
>>> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

loadbalancer_create(*loadbalancer*)

> Creates a new load balancer.
>
>> **Parameters** **loadbalancer** (`object`) -- The load balancer object.
>>
>> **Returns** Nothing if the create request was accepted.
>>
>> **Raises**
>>
>>> - *DriverError* -- An unexpected error occurred in the driver.
>>>
>>> - *NotImplementedError* -- The driver does not support create.
>>>
>>> - *UnsupportedOptionError* -- The driver does not support one of the configuration options.

loadbalancer_delete(*loadbalancer*, *cascade=False*)

> Deletes a load balancer.
>
>> **Parameters**
>>
>>> - **loadbalancer** (`object`) -- The load balancer to delete.
>>>
>>> - **cascade** (`bool`) -- If True, deletes all child objects (listeners, pools, etc.) in addition to the load balancer.
>>
>> **Returns** Nothing if the delete request was accepted.
>>
>> **Raises**
>>
>>> - *DriverError* -- An unexpected error occurred in the driver.

- *NotImplementedError* -- if driver does not support request.

loadbalancer_failover(*loadbalancer_id*)

> Performs a fail over of a load balancer.
>
> > **Parameters loadbalancer_id** (`string`) -- ID of the load balancer to failover.
> >
> > **Returns** Nothing if the failover request was accepted.
> >
> > **Raises** *DriverError* -- An unexpected error occurred in the driver.
> >
> > **Raises** NotImplementedError if driver does not support request.

loadbalancer_update(*old_loadbalancer*, *new_loadbalancer*)

> Updates a load balancer.
>
> > **Parameters**
> >
> > - **old_loadbalancer** (`object`) -- The baseline load balancer object.
> >
> > - **new_loadbalancer** (`object`) -- The updated load balancer object.
> >
> > **Returns** Nothing if the update request was accepted.
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- The driver does not support request.
> >
> > - *UnsupportedOptionError* -- The driver does not support one of the configuration options.

member_batch_update(*pool_id*, *members*)

> Creates, updates, or deletes a set of pool members.
>
> > **Parameters**
> >
> > - **pool_id** (`string`) -- The id of the pool to update.
> >
> > - **members** (`list`) -- List of member objects.
> >
> > **Returns** Nothing if the create request was accepted.
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- if driver does not support request.
> >
> > - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

member_create(*member*)

> Creates a new member for a pool.
>
> > **Parameters member** (`object`) -- The member object.
> >
> > **Returns** Nothing if the create request was accepted.
> >
> > **Raises**
> >
> > - *DriverError* -- An unexpected error occurred in the driver.
> >
> > - *NotImplementedError* -- if driver does not support request.

- *UnsupportedOptionError* -- if driver does not support one of the configuration options.

member_delete(*member*)

Deletes a pool member.

> **Parameters member** (*object*) -- The member to delete.
>
> **Returns** Nothing if the create request was accepted.
>
> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.

member_update(*old_member*, *new_member*)

Updates a pool member.

> **Parameters**
>
> - **old_member** (*object*) -- The baseline member object.
>
> - **new_member** (*object*) -- The updated member object.
>
> **Returns** Nothing if the create request was accepted.
>
> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.
>
> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

name = None

pool_create(*pool*)

Creates a new pool.

> **Parameters pool** (*object*) -- The pool object.
>
> **Returns** Nothing if the create request was accepted.
>
> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.
>
> - *NotImplementedError* -- if driver does not support request.
>
> - *UnsupportedOptionError* -- if driver does not support one of the configuration options.

pool_delete(*pool*)

Deletes a pool and its members.

> **Parameters pool** (*object*) -- The pool to delete.
>
> **Returns** Nothing if the create request was accepted.
>
> **Raises**
>
> - *DriverError* -- An unexpected error occurred in the driver.

      • *NotImplementedError* -- if driver does not support request.

pool_update(*old_pool*, *new_pool*)

    Updates a pool.

> **Parameters**
>
>   • **pool** (*object*) -- The baseline pool object.
>
>   • **pool** -- The updated pool object.
>
> **Returns** Nothing if the create request was accepted.
>
> **Raises**
>
>   • *DriverError* -- An unexpected error occurred in the driver.
>
>   • *NotImplementedError* -- if driver does not support request.
>
>   • *UnsupportedOptionError* -- if driver does not support one of the configuration options.

validate_availability_zone(*availability_zone_metadata*)

    Validates if driver can support the availability zone.

> **Parameters** **availability_zone_metadata** (*dict*) -- Dictionary with az metadata.
>
> **Returns** Nothing if the availability zone is valid and supported.
>
> **Raises**
>
>   • *DriverError* -- An unexpected error occurred in the driver.
>
>   • *NotImplementedError* -- The driver does not support availability zones.
>
>   • *UnsupportedOptionError* -- if driver does not support one of the configuration options.

validate_flavor(*flavor_metadata*)

    Validates if driver can support the flavor.

> **Parameters** **flavor_metadata** (*dict*) -- Dictionary with flavor metadata.
>
> **Returns** Nothing if the flavor is valid and supported.
>
> **Raises**
>
>   • *DriverError* -- An unexpected error occurred in the driver.
>
>   • *NotImplementedError* -- The driver does not support flavors.
>
>   • *UnsupportedOptionError* -- if driver does not support one of the configuration options.
>
>   • *octavia_lib.api.drivers.exceptions.NotFound* -- if the driver cannot find a resource.

Module contents

Module contents

octavia_lib.common package

Submodules

octavia_lib.common.constants module

Module contents

Submodules

octavia_lib.version module

octavia_lib.version.**product_string**()

octavia_lib.version.**vendor_string**()

octavia_lib.version.**version_string_with_package**()

Module contents