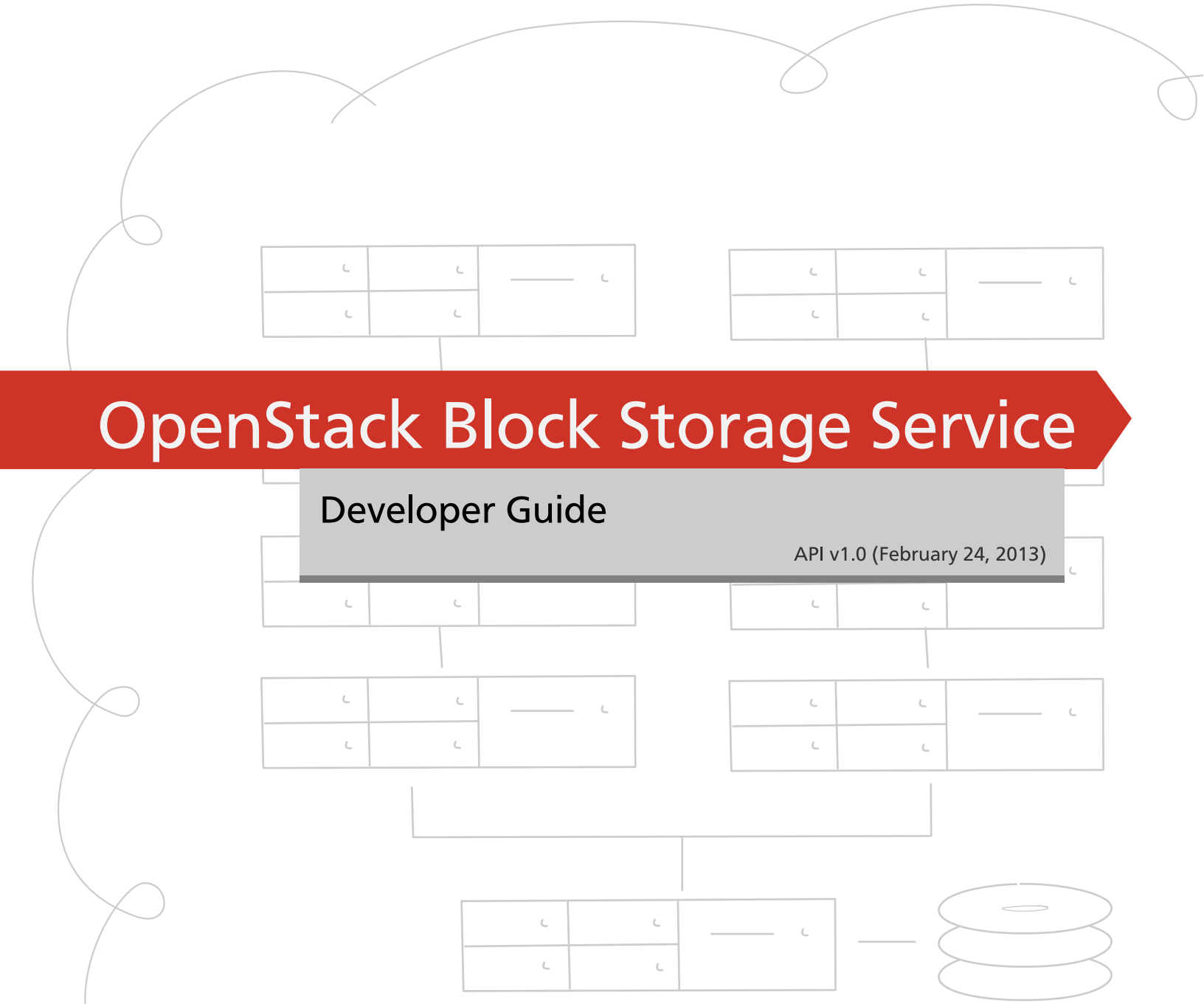

OpenStack Block Storage Service

Developer Guide

API v1.0 (February 24, 2013)

BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack Block Storage Service Developer Guide

API v1.0 (2013-02-24)

This document is intended for software developers interested in developing applications using the OpenStack Block Storage Service Application Programming Interface (API).

List of Tables

List of Examples

Preface

Table of Contents

The Cinder project provides volume management with the OpenStack compute service.

This document describes the features available with the Cinder API v1.0.

We welcome feedback, comments and bug reports at bugs.launchpad.net/Cinder.

1. Intended Audience

This Guide is intended to assist software developers who want to develop applications using the Cinder API v1.0. It assumes the reader has a general understanding of storage and is familiar with:

- ReSTful web services
- HTTP/1.1 conventions
- JSON and/or XML data serialization formats

2. Additional Resources

You can download the most current versions of the API-related documents from docs.openstack.org/api/.

This API uses standard HTTP 1.1 response codes as documented at: www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

Overview

Table of Contents

OpenStack Block Storage Service is a block-level storage solution that allows customers to mount drives or volumes to their OpenStack Compute servers™. The two primary use cases are (1) to allow customers to scale their storage independently from their compute resources, and (2) to allow customers to utilize high performance storage to serve database or I/O-intensive applications.

Interactions with Block Storage occur programmatically via the Block Storage API as described in this Developer Guide.

Highlights of OpenStack Block Storage Service include:

- Mount a drive to a Compute server to scale storage without paying for more compute capability.

Notes

- OpenStack Block Storage Service is an add-on feature to OpenStack Nova Compute in Folsom versions and earlier.
- Block Storage is multi-tenant rather than dedicated.
- Block Storage allows you to create snapshots that you can save, list, and restore.

1. Glossary

To use the Block Storage API effectively, you should understand several key concepts:

1.1. Volume

A volume is a detachable block storage device. You can think of it as a USB hard drive. It can only be attached to one instance at a time.

1.2. Snapshot

A snapshot is a point in time copy of the data contained in a volume.

1.3. Volume Type

The volume type is the type of a block storage volume. You may define whatever types work best for you, such as SATA, SCSCI, SSD, etc. These can be customized or defined by the OpenStack admin.

You may also define extra_specs associated with your volume types. For instance, you could have a VolumeType=SATA, with extra_specs (RPM=10000, RAID-Level=5) . Extra_specs are defined and customized by the admin.

1.4. Instance

An instance is a virtual machine that runs inside the cloud.

2. High-Level Task Flow

The high-level task flow for Cinder is as follows:

1. The tenant creates a volume.

   For example, the tenant creates a 30G volume called vol1.

   ```
   $cinder create --display-name vol1 30
   ```

2. This gives the tenant a volume id 521752a6-acf6-4b2d-bc7a-119f9148cd8c. The tenant attaches that volume to a virtual machine (VM) 616fb98f-46ca-475e-917e-2563e5a8cd19:

   For example A:

   ```
   $ nova volume-attach 616fb98f-46ca-475e-917e-2563e5a8cd19 521752a6-
   acf6-4b2d-bc7a-119f9148cd8c /dev/vdb
   ```

1. General API Information

Table of Contents

1.1. Authentication

You can use cURL to try the authentication process in two steps: get a token; send the token to a service.

1. Get an authentication token by providing your username and either your API key or your password. Here are examples of both approaches:

   You can request a token by providing your username and your password.

   ```
   curl -X POST https://auth.api.openstackcloud.com/v2.0/tokens -d '{"auth":
   {"passwordCredentials":{"username": "joecool", "password":"coolword"},
    "tenantId":"5"}}' -H 'Content-type: application/json'
   ```

   Successful authentication returns a token which you can use as evidence that your identity has already been authenticated. To use the token, pass it to other services as an `X-Auth-Token` header.

   Authentication also returns a service catalog, listing the endpoints you can use for Cloud services.

2. Use the authentication token to send a GET to a service you would like to use.

Authentication tokens are typically valid for 24 hours. Applications should be designed to re-authenticate after receiving a 401 (Unauthorized) response from a service endpoint.

Important

If you are programmatically parsing an authentication response, please be aware that service names are stable for the life of the particular service and can be used as keys. You should also be aware that a user's service catalog can include multiple uniquely-named services which perform similar functions.

1.2. Request/Response Types

The Block Storage API supports both the JSON and XML data serialization formats. The request format is specified using the `Content-Type` header and is required for calls that have a request body. The response format can be specified in requests either by using the

`Accept` header or by adding an `.xml` or `.json` extension to the request URI. Note that it is possible for a response to be serialized using a format different from the request. If no response format is specified, JSON is the default. If conflicting formats are specified using both an `Accept` header and a query extension, the query extension takes precedence.

Table 1.1. Response Formats

Format	Accept Header	Query Extension	Default
JSON	application/json	.json	Yes
XML	application/xml	.xml	No

In the request example below, notice that *`Content-Type`* is set to *`application/json`*, but *`application/xml`* is requested via the *`Accept`* header:

Example 1.1. Request with Headers (Getting Volume Types)

```
GET /v1/441446/types HTTP/1.1
Host: dfw.blockstorage.api.openstackcloud.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Accept: application/xml
```

Therefore an XML response format is returned:

Example 1.2. Response with Headers

```
HTTP/1.1 200 OK
Date: Fri, 20 Jul 2012 20:32:13 GMT
Content-Length: 187
Content-Type: application/xml
X-Compute-Request-Id: req-8e0295cd-a283-46e4-96da-cae05cbfd1c7

<?xml version='1.0' encoding='UTF-8'?>
<volume_types>
    <volume_type id="1" name="SATA">
        <extra_specs/>
    </volume_type>
    <volume_type id="2" name="SSD">
        <extra_specs/>
    </volume_type>
</volume_types>
```

1.3. Limits

All accounts, by default, have a preconfigured set of thresholds (or limits) to manage capacity and prevent abuse of the system. The system recognizes two kinds of limits: *rate limits* and *absolute limits*. Rate limits are thresholds that are reset after a certain amount of time passes. Absolute limits are fixed.

1.3.1. Absolute Limits

Refer to the following table for the absolute limits that are set.

Table 1.2. Absolute Limits

Name	Description	Limit
Block Storage	Maximum amount of block storage (in gigabytes)	1 TB

1.4. Date/Time Format

The Block Storage Service uses an ISO-8601 compliant date format for the display and consumption of date/time values.

Example 1.3. DB Service Date/Time Format

```
yyyy-MM-dd'T'HH:mm:ss.SSSZ
```

See the table below for a description of the date/time format codes.

May 19th, 2011 at 8:07:08 AM, GMT-5 would have the following format:

```
2011-05-19T08:07:08-05:00
```

Table 1.3. Explanation of Date/Time Format Codes

Code	Description
yyyy	Four digit year
MM	Two digit month
dd	Two digit day of month
T	Separator for date/time
HH	Two digit hour of day (00-23)
mm	Two digit minutes of hour
ss	Two digit seconds of the minute
SSS	Three digit milliseconds of the second
Z	RFC-822 timezone

1.5. Faults

When an error occurs, the Block Storage Service returns a fault object containing an HTTP error response code that denotes the type of error. In the body of the response, the system will return additional information about the fault.

The following table lists possible fault types with their associated error codes and descriptions.

Fault Type	Associated Error Code	Description
`badRequest`	400	There was one or more errors in the user request.
`unauthorized`	401	The supplied token is not authorized to access the resources, either it's expired or invalid.
`forbidden`	403	Access to the requested resource was denied.
`itemNotFound`	404	The back-end services did not find anything matching the Request-URI.
`badMethod`	405	The request method is not allowed for this resource.

Fault Type	Associated Error Code	Description
overLimit	413	Either the number of entities in the request is larger than allowed limits, or the user has exceeded allowable request rate limits. See the details element for more specifics. Contact support if you think you need higher request rate limits.
badMediaType	415	The requested content type is not supported by this service.
unprocessableEntity	422	The requested resource could not be processed on at the moment.
instanceFault	500	This is a generic server error and the message contains the reason for the error. This error could wrap several error messages and is a catch all.
notImplemented	501	The requested method or resource is not implemented.
serviceUnavailable	503	The Block Storage Service is not available.

The following two instanceFault examples show errors when the server has erred or cannot perform the requested operation:

Example 1.4. Example instanceFault Response: XML

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/xml
Content-Length: 121
Date: Mon, 28 Nov 2011 18:19:37 GMT
```

```
         <?xml version="1.0" encoding="UTF-8"?>
<instanceFault code="500"
    xmlns="http://docs.rackspace.com/cbs/api/v1.0">
    <message> The server has either erred or is incapable of
        performing the requested operation. </message>
</instanceFault>
```

Example 1.5. Example Fault Response: JSON

```
HTTP/1.1 500 Internal Server Error
Content-Length: 120
Content-Type: application/json; charset=UTF-8
Date: Tue, 29 Nov 2011 00:33:48 GMT
```

```
         {
    "instanceFault": {
        "code": 500,
        "message": "The server has either erred or is incapable of performing
 the requested operation."
    }
}
```

The error code (code) is returned in the body of the response for convenience. The message element returns a human-readable message that is appropriate for display to the end user. The details element is optional and may contain information that is useful for

tracking down an error, such as a stack trace. The `details` element may or may not be appropriate for display to an end user, depending on the role and experience of the end user.

The fault's root element (for example, `instanceFault`) may change depending on the type of error.

The following two `badRequest` examples show errors when the volume size is invalid:

Example 1.6. Example badRequest Fault on Volume Size Errors: XML

```
HTTP/1.1 400 None
Content-Type: application/xml
Content-Length: 121
Date: Mon, 28 Nov 2011 18:19:37 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<badRequest code="400"
    xmlns="http://docs.openstack.org/api/openstack-volume/1.0/content">
    <message> Volume 'size' needs to be a positive integer value, -1.0
        cannot be accepted. </message>
</badRequest>
```

Example 1.7. Example badRequest Fault on Volume Size Errors: JSON

```
HTTP/1.1 400 None
Content-Length: 120
Content-Type: application/json; charset=UTF-8
Date: Tue, 29 Nov 2011 00:33:48 GMT
```

```
{
    "badRequest": {
        "code": 400,
        "message": "Volume 'size' needs to be a positive integer value, -1.0
 cannot be accepted."
    }
}
```

The next two examples show `itemNotFound` errors:

Example 1.8. Example itemNotFound Fault: XML

```
HTTP/1.1 404 Not Found
Content-Length: 147
Content-Type: application/xml; charset=UTF-8
Date: Mon, 28 Nov 2011 19:50:15 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<itemNotFound code="404"
    xmlns="http://docs.openstack.org/api/openstack-volume/1.0/content">
    <message> The resource could not be found. </message>
</itemNotFound>
```

Example 1.9. Example itemNotFound Fault: JSON

```
HTTP/1.1 404 Not Found
```

```
Content-Length: 78
Content-Type: application/json; charset=UTF-8
Date: Tue, 29 Nov 2011 00:35:24 GMT
```

```
{
    "itemNotFound": {
        "code": 404,
        "message": "The resource could not be found."
    }
}
```

2. API Operations

Table of Contents

2.1. Volumes

A volume is a detachable block storage device. You can think of it as a USB hard drive. It can only be attached to one instance at a time.

When making an API call to create, list, or delete volume(s), the following status values are possible:

- CREATING – The volume is being created.
- AVAILABLE – The volume is read to be attached to an instance.
- ATTACHING – The volume is attaching to an instance.
- IN-USE – The volume is attached to an instance.
- DELETING – The volume is being deleted.
- ERROR – An error has occurred with the volume.
- ERROR_DELETING – There was an error deleting the volume.

Verb	URI	Description
GET	/volumes	Lists a summary of all volumes defined in Cinder that are accessible to the tenant who submits the request.
GET	/volumes/*volume_id*	Lists detailed information for the specified volume.
POST	/volumes	Creates a new Cinder volume.
PUT	/volumes/*volume_id*	Updates the specified volume.
DELETE	/volumes/*volume_id*	Destroys the specified volume.

2.1.1. Create Volume

Verb	URI	Description
POST	/volumes	Create a new volume.

Example 2.1. Create Volumes: JSON Request

```
GET /v1/volumes
Content-Type: application/json
Accept: application/json
```

```
        {
    "volume": {
        "display_name": "vol-001",
        "display_description": "Another volume.",
        "size": 30,
        "volume_type": "289da7f8-6440-407c-9fb4-7db01ec49164",
        "metadata": {"contents": "junk"},
        "availability_zone": "us-east1"
     }
}
```

Example 2.2. Create Volumes: JSON Response

```
        {
    "volume": {
        "attachments": [],
        "availability_zone": "us-east1",
        "bootable": "false",
        "created_at": "2013-02-20T07:36:44.893964",
        "display_description": "Another volume.",
        "display_name": "vol-001",
        "id": "2402b902-0b7a-458c-9c07-7435a826f794",
        "metadata": {
            "contents": "junk"
        },
        "size": 30,
        "snapshot_id": null,
        "source_volid": null,
        "status": "creating",
        "volume_type": "289da7f8-6440-407c-9fb4-7db01ec49164"
    }
}
```

Returns status code 200 on success

2.1.2. List Volume Summaries

Verb	URI	Description
GET	/volumes	Lists a summary of all volumes defined in Cinder that are accessible to the tenant who submits the request.

Example 2.3. List Volumes: Request

```
GET /v1/volumes
Accept: application/json
```

Example 2.4. List Volumes: JSON Response

```
        {
    "volumes": [
        {
            "attachments": [],
            "availability_zone": "us-east1",
            "bootable": "false",
            "created_at": "2013-02-18T23:47:37.000000",
            "display_description": "yet another volume",
            "display_name": "vol-002",
            "id": "dfd9dd41-1353-450a-b5ac-a6037ee43cd8",
            "metadata": {
                "contents": "some more junk"
            },
            "size": 30,
            "snapshot_id": null,
            "source_volid": null,
            "status": "available",
            "volume_type": "SATA"
        },
        {
            "attachments": [],
            "availability_zone": "us-east1",
            "bootable": "false",
            "created_at": "2013-02-18T23:44:47.000000",
            "display_description": "another volume",
            "display_name": "vol-001",
            "id": "da450d0c-0920-4785-80dd-b024515200ce",
            "metadata": {
                "contents": "junk"
            },
            "size": 10,
            "snapshot_id": null,
            "source_volid": null,
            "status": "available",
            "volume_type": "SATA"
        }
    ]
}
```

Returns status code 200 on success

Note

The response from list volume summary is the same as list volume details in v1. This is corrected in v2.

2.1.3. List Volume Details

Verb	URI	Description
GET	/volumes/details	List detailed information of all volumes defined in Cinder that are accessible to the tenant who submits the request.

Example 2.5. List Volumes: Request

```
GET /v1/volumes/details
Accept: application/json
```

Example 2.6. List Volumes: JSON Response

```
      {
    "volumes": [
        {
            "attachments": [],
            "availability_zone": "us-east1",
            "bootable": "false",
            "created_at": "2013-02-18T23:47:37.000000",
            "display_description": "yet another volume",
            "display_name": "vol-002",
            "id": "dfd9dd41-1353-450a-b5ac-a6037ee43cd8",
            "metadata": {
                "contents": "some more junk"
            },
            "size": 30,
            "snapshot_id": null,
            "source_volid": null,
            "status": "available",
            "volume_type": "SATA"
        },
        {
            "attachments": [],
            "availability_zone": "us-east1",
            "bootable": "false",
            "created_at": "2013-02-18T23:44:47.000000",
            "display_description": "another volume",
            "display_name": "vol-001",
            "id": "da450d0c-0920-4785-80dd-b024515200ce",
            "metadata": {
                "contents": "junk"
            },
            "size": 10,
            "snapshot_id": null,
            "source_volid": null,
            "status": "available",
            "volume_type": "SATA"
        }
    ]
}
```

Returns status code 200 on success

2.1.4. Show Volume

Verb	URI	Description
GET	/volumes/*volume-id*	Lists detailed information for the specified volume ID.

Example 2.7. Show Volume: Request

```
GET /v1/volumes/2402b902-0b7a-458c-9c07-7435a826f794
Accept: application/json
```

Example 2.8. Show Volume: JSON Response

```
        {
    "volume": {
        "attachments": [],
        "availability_zone": "nova",
        "bootable": "false",
        "created_at": "2013-02-20T07:36:44.000000",
        "display_description": "another volume foo",
        "display_name": "vol-002",
        "id": "2402b902-0b7a-458c-9c07-7435a826f794",
        "metadata": {
            "content": "junk"
        },
        "os-vol-host-attr:host": "precise64",
        "os-vol-tenant-attr:tenant_id": "dee102070e654627a1f96acc5dcad496",
        "size": 1,
        "snapshot_id": null,
        "source_volid": null,
        "status": "available",
        "volume_type": "None"
    }
}
```

Returns status code 200 on success

2.1.5. Update Volume

Verb	URI	Description
PUT	/volumes/*volume-id*	Update the specified volume.

Example 2.9. Update Volume: JSON Request

```
PUT /v1/volumes2402b902-0b7a-458c-9c07-7435a826f794
Content-Type: application/json
Accept: application/json
```

```
        {
   "volume":{
      "display_name":"vol-002",
      "display_description":"This is yet, another volume.",
      "metadata":{
         "contents":"junk"
      }
   }
}
```

Example 2.10. Update Volume: JSON Response

```
        {
    "volume": {
        "attachments": [],
        "availability_zone": "nova",
        "bootable": "false",
        "created_at": "2013-02-20T07:36:44.000000",
        "display_description": "This is yet, another volume.",
        "display_name": "vol-002",
        "id": "2402b902-0b7a-458c-9c07-7435a826f794",
        "metadata": {
            "tier": "1"
        },
        "size": 1,
        "snapshot_id": null,
        "source_volid": null,
        "status": "available",
        "volume_type": "None"
    }
}
```

Returns status code 200 on success

2.1.6. Delete Volume

Verb	URI	Description
DELETE	/volumes/*volume-id*	Destroys the specified volume.

Example 2.11. Delete Volume: Request

```
DELETE /v1/volumes/521752a6-acf6-4b2d-bc7a-119f9148cd8c
```

Example 2.12. Delete Volume: Response

```
The response body will be empty with status code 202.
```

Caution

You cannot delete a volume if it has snapshots associated with it.

2.2. Snapshots

A snapshot is a point in time copy of the data contained in a volume.

When making an API call to create, list, or delete snapshot(s), the following status values are possible:

- CREATING – The snapshot is being created.
- AVAILABLE – The snapshot is ready to be used.
- DELETING – The snapshot is being deleted.
- ERROR – An error occurred with the snapshot.
- ERROR_DELETING – There was an error deleting the snapshot.

2.2.1. Create Snapshot

Verb	URI	Description
POST	/snapshots	Create a snapshot of specified volume.

Example 2.13. Create Snapshot: JSON Request

```
POST /v1/snapshots
Content-Type: application/json
Accept: application/json
```

```
          {
   "snapshot": {
       "display_name": "snap-001",
       "display_description": "Daily backup",
       "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
       "force": true
    }
}
```

Example 2.14. Create Snapshot: JSON Response

```
            {
    "snapshot": {
        "created_at": "2013-02-19T00:24:54.179025",
        "display_description": "Daily backup",
        "display_name": "snap-001",
        "id": "2d843196-eff1-4a0c-980e-10fee5df2bea",
        "size": 10,
        "status": "creating",
        "volume_id": "da450d0c-0920-4785-80dd-b024515200ce"
    }
}
```

Returns status code 202 on success

2.2.2. List Snapshot Summaries

Verb	URI	Description
GET	/snapshots	Lists a summary of all snapshots defined in Cinder that are accessible to the tenant who submits the request.

Example 2.15. List Snapshots: Request

```
GET /v1/snapshots
Accept: application/json
```

Example 2.16. List Snapshots: JSON Response

```
          {
    "snapshots": [
        {
            "id": "3fbbcccf-d058-4502-8844-6feeffdf4cb5",
            "display_name": "snap-001",
            "display_description": "Daily backup",
            "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
            "status": "available",
            "size": 10,
            "created_at": "2012-02-29T03:50:07Z"
        },
        {
            "id": "e479997c-650b-40a4-9dfe-77655818b0d2",
            "display_name": "snap-002",
            "display_description": "Weekly backup",
            "volume_id": "76b8950a-8594-4e5b-8dce-0dfa9c696358",
            "status": "available",
            "size": 25,
            "created_at": "2012-03-19T01:52:47Z"
        }
    ]
}
```

Returns status code 200 on success

Note

The response from list snapshots summary is the same as list snapshot details in v1.

2.2.3. List Snapshot Details

Verb	URI	Description
GET	/snapshots/details	List detailed information of all snapshots defined in Cinder that are accessible to the tenant who submits the request.

Example 2.17. List Snapshots Details: Request

```
GET /v1/snapshots/details
Accept: application/json
```

Example 2.18. List Snapshots Details: JSON Response

```
        {
    "snapshots": [
        {
            "id": "3fbbcccf-d058-4502-8844-6feeffdf4cb5",
            "display_name": "snap-001",
            "display_description": "Daily backup",
            "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
            "status": "available",
            "size": 10,
            "created_at": "2012-02-29T03:50:07Z"
        },
        {
            "id": "e479997c-650b-40a4-9dfe-77655818b0d2",
            "display_name": "snap-002",
            "display_description": "Weekly backup",
            "volume_id": "76b8950a-8594-4e5b-8dce-0dfa9c696358",
            "status": "available",
            "size": 25,
            "created_at": "2012-03-19T01:52:47Z"
        }
    ]
}
```

Returns status code 200 on success

2.2.4. Show Snapshot

Verb	URI	Description
GET	/ snapshots/ *snapshot-id*	Lists detailed information for the specified snapshot ID.

Example 2.19. Show Snapshot: Request

```
GET /v1/snapshots/3fbbcccf-d058-4502-8844-6feeffdf4cb5
Accept: application/json
```

Example 2.20. Show Snapshot: JSON Response

```
        {
    "snapshot": {
        "id": "3fbbcccf-d058-4502-8844-6feeffdf4cb5",
        "display_name": "snap-001",
        "display_description": "Daily backup",
        "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
        "status": "creating",
        "size": 30,
        "created_at": "2012-02-29T03:50:07Z"
     }
}
```

Returns status code 200 on success

2.2.5. Update Snapshot

Verb	URI	Description
PUT	/ snapshots/ *snapshot-id*	Update the specified snapshot.

Example 2.21. Update Snapshot: JSON Request

```
PUT /v1/snapshots/2402b902-0b7a-458c-9c07-7435a826f794
Content-Type: application/json
Accept: application/json
```

```
        {
   "snapshot":{
      "display_name":"snap-002",
      "display_description":"This is yet, another snapshot."
   }
}
```

Example 2.22. Update Snapshot: JSON Response

```
        {
    "snapshot": {
        "created_at": "2013-02-20T08:11:34.000000",
        "display_description": "This is yet, another snapshot",
        "display_name": "vol-002",
        "id": "4b502fcb-1f26-45f8-9fe5-3b9a0a52eaf2",
        "size": 1,
        "status": "available",
        "volume_id": "2402b902-0b7a-458c-9c07-7435a826f794"
    }
}
```

Returns status code 200 on success

2.2.6. Delete Snapshot

Verb	URI	Description
DELETE	/ snapshots/ *snapshot-id*	Destroys the specified snapshot.

Example 2.23. Delete Snapshot: Request

```
DELETE /v1/snapshots/3fbbcccf-d058-4502-8844-6feeffdf4cb5
```

Example 2.24. Delete Snapshot: Response

```
The response body will be empty with status code 202.
```

2.3. Volume Types

2.3.1. List Volume Types

Verb	URI	Description
GET	/types	List all volume types and their information.

Example 2.25. List Volume Types: Request

```
GET /v1/types
Accept: application/json
```

Example 2.26. List Volume Types: JSON Response

```
            {
    "volume_types": [
        {
            "extra_specs": {},
            "id": "6685584b-1eac-4da6-b5c3-555430cf68ff",
            "name": "SSD"
        },
        {
            "extra_specs": {},
            "id": "8eb69a46-df97-4e41-9586-9a40a7533803",
            "name": "SATA"
        }
    ]
}
```

Returns status code 200 on success

2.3.2. Show Volume Type

Verb	URI	Description
GET	/types/*type-id*	Show information on a specified volume type.

Example 2.27. Show Volume Type: Request

```
GET /v1/types/6685584b-1eac-4da6-b5c3-555430cf68ff
Accept: application/json
```

Example 2.28. Show Volume Type: JSON Response

```
            {
   "volume_type":{
      "id":"6685584b-1eac-4da6-b5c3-555430cf68ff",
      "name":"SSD",
      "extra_specs":{

      }
   }
}
```

Returns status code 200 on success