

OpenStack Cloud Databases

Developer Guide

API v1.0 (May 2, 2013)



OpenStack Cloud Databases Developer Guide

API v1.0 (2013-05-02)

Copyright © 2010-2013 OpenStack LLC All rights reserved.

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

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the 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.

Table of Contents

1. Overview	1
1.1. Intended Audience	3
1.2. Document Change History	4
1.3. Additional Resources	4
2. Concepts	5
2.1. Database Instance	5
2.2. Database	5
2.3. Flavor	5
2.4. Volume	5
3. General API Information	6
3.1. Authentication	6
3.2. Cloud Databases Service Versions	6
3.2.1. Contract Version	6
3.2.2. API Version	6
3.3. Date/Time Format	7
3.4. Pagination	7
3.5. Faults	11
3.5.1. Synchronous Versus Asynchronous Faults	13
3.6. Database Instance Status	14
4. API Operations	15
4.1. API Versions	16
4.1.1. List Versions	17
4.1.2. List Version Details	19
4.2. Database Instances	20
4.2.1. Create Database Instance	21
4.2.2. List All Database Instances	25
4.2.3. List Database Instance Status and Details	29
4.2.4. Delete Database Instance	32
4.2.5. Enable Root User	34
4.2.6. List Root-Enabled Status	36
4.3. Database Instance Actions	37
4.3.1. Restart Instance	38
4.3.2. Resize the Instance	40
4.3.3. Resize the Instance Volume	42
4.4. Databases	43
4.4.1. Create Database	44
4.4.2. List Databases for Instance	47
4.4.3. Delete Database	51
4.5. Users	52
4.5.1. Create User	53
4.5.2. List Users in Database Instance	57
4.5.3. Change User(s) Password	62
4.5.4. List User	65
4.5.5. Delete User	67
4.5.6. List User Access	69
4.5.7. Grant User Access	71
4.5.8. Revoke User Access	73
4.6. Flavors	74

4.6.1. List Flavors	75
4.6.2. List Flavor By ID	79
Glossary	81

List of Tables

3.1. Explanation of Date/Time Format Codes	7
4.1. Required and Optional Attributes for Create Instance	21
4.2. Required and Optional Attributes for Create Database	44
4.3. Valid Characters That Can Be Used in a Database Name	44
4.4. Characters That <i>Cannot</i> Be Used in a Database Name	44
4.5. Length Restrictions for Database Name	45
4.6. Required Attributes for Create User	53
4.7. Valid Characters That Can Be Used in a Database Name, User Name, and Password	53
4.8. Characters That <i>Cannot</i> Be Used in a Database Name, User Name, and Password	54
4.9. Length Restrictions for Database Name, User Name, and Password	54

List of Examples

3.1. Example Request URL (contract version in bold)	6
3.2. DB Service Date/Time Format	7
3.3. List Instances Paged Request: XML	8
3.4. List Instances Paged Request: JSON	8
3.5. List Instances Paged Response: XML	9
3.6. List Instances Paged Response: JSON	10
3.7. Example instanceFault Response: XML	11
3.8. Example Fault Response: JSON	12
3.9. Example badRequest Fault on Volume Size Errors: XML	12
3.10. Example badRequest Fault on Volume Size Errors: JSON	13
3.11. Example itemNotFound Fault: XML	13
3.12. Example itemNotFound Fault: JSON	13
4.1. List Versions Request: XML	17
4.2. List Versions Request: JSON	17
4.3. List Versions Response: XML	17
4.4. List Versions Response: JSON	18
4.5. List Version Details Request: XML	19
4.6. List Version Details Request: JSON	19
4.7. List Version Details Response: XML	19
4.8. List Version Details Response: JSON	20
4.9. Create Database Instance Request: XML	22
4.10. Create Database Instance Request: JSON	22
4.11. Create Database Instance Response: XML	23
4.12. Create Database Instance Response: JSON	24
4.13. List All Database Instances Request: XML	25
4.14. List All Database Instances Request: JSON	25
4.15. List All Database Instances Response: XML	25
4.16. List All Database Instances Response: JSON	26
4.17. List Database Instance Status and Details Request: XML	29
4.18. List Database Instance Status and Details Request: JSON	30
4.19. List Database Instance Status and Details Response: XML	30
4.20. List Database Instance Status and Details Response: JSON	30
4.21. Delete Database Instance Request: XML	32
4.22. Delete Database Instance Request: JSON	32
4.23. Enable Root User Request: XML	34
4.24. Enable Root User Request: JSON	34
4.25. Enable Root User Response: XML	35
4.26. Enable Root User Response: JSON	35
4.27. List Root-Enabled Status Request: XML	36
4.28. List Root-Enabled Status Request: JSON	36
4.29. List Root-Enabled Status Response: XML	36
4.30. List Root-Enabled Status Response: JSON	37
4.31. Restart Instance Request: XML	38
4.32. Restart Instance Request: JSON	38
4.33. Restart Instance Response: XML	39
4.34. Restart Instance Response: JSON	39
4.35. Resize the Instance Request: XML	40
4.36. Resize the Instance Request: JSON	40

4.37. Resize the Instance Response: XML	41
4.38. Resize the Instance Response: JSON	41
4.39. Resize the Instance Volume Request: XML	42
4.40. Resize the Instance Volume Request: JSON	42
4.41. Resize the Instance Volume Response: XML	43
4.42. Resize the Instance Volume Response: JSON	43
4.43. Create Database Request: XML	45
4.44. Create Database Request: JSON	45
4.45. Create Database Response: XML	46
4.46. Create Database Response: JSON	46
4.47. List Databases for Instance Request: XML	47
4.48. List Databases for Instance Request: JSON	47
4.49. List Databases for Instance Paged Request: XML	48
4.50. List Databases for Instance Paged Request: JSON	48
4.51. List Databases for Instance Response: XML	48
4.52. List Databases for Instance Response: JSON	48
4.53. List Databases for Instance Paged Response: XML	49
4.54. List Databases for Instance Paged Response: JSON	49
4.55. Delete Database Request: XML	51
4.56. Delete Database Request: JSON	51
4.57. Delete Database Response: XML	52
4.58. Delete Database Response: JSON	52
4.59. Create User Request: XML	54
4.60. Create User Request: JSON	55
4.61. Create User Response: XML	55
4.62. Create User Response: JSON	55
4.63. List Users in Database Instance Request: XML	57
4.64. List Users in Database Instance Request: JSON	57
4.65. List Users in Database Instance Paged Request: XML	58
4.66. List Users in Database Instance Paged Request: JSON	58
4.67. List Users in Database Instance Response: XML	58
4.68. List Users in Database Instance Response: JSON	59
4.69. List Users in Database Instance Paged Response: XML	60
4.70. List Users in Database Instance Paged Response: JSON	60
4.71. Change User(s) Password Request: XML	62
4.72. Change User(s) Password Request: JSON	62
4.73. Change User(s) Password Response: XML	63
4.74. Change User(s) Password Response: JSON	64
4.75. List User Request: XML	65
4.76. List User Request: JSON	65
4.77. List User Response: XML	66
4.78. List User Response: JSON	66
4.79. Delete User Request: XML	67
4.80. Delete User Request: JSON	67
4.81. Delete User Response: XML	68
4.82. Delete User Response: JSON	68
4.83. List User Access Request: XML	69
4.84. List User Access Request: JSON	69
4.85. List User Access Response: XML	69
4.86. List User Access Response: JSON	70
4.87. Grant User Access Request: XML	71

4.88. Grant User Access Request: JSON	71
4.89. Grant User Access Response: XML	72
4.90. Grant User Access Response: JSON	72
4.91. Revoke User Access Request: XML	73
4.92. Revoke User Access Request: JSON	73
4.93. Revoke User Access Response: XML	74
4.94. Revoke User Access Response: JSON	74
4.95. List Flavors Request: XML	75
4.96. List Flavors Request: JSON	75
4.97. List Flavors Response: XML	75
4.98. List Flavors Response: JSON	76
4.99. List Flavor By ID Request: XML	79
4.100. List Flavor By ID Request: JSON	79
4.101. List Flavor By ID Response: XML	80
4.102. List Flavor By ID Response: JSON	80

1. Overview

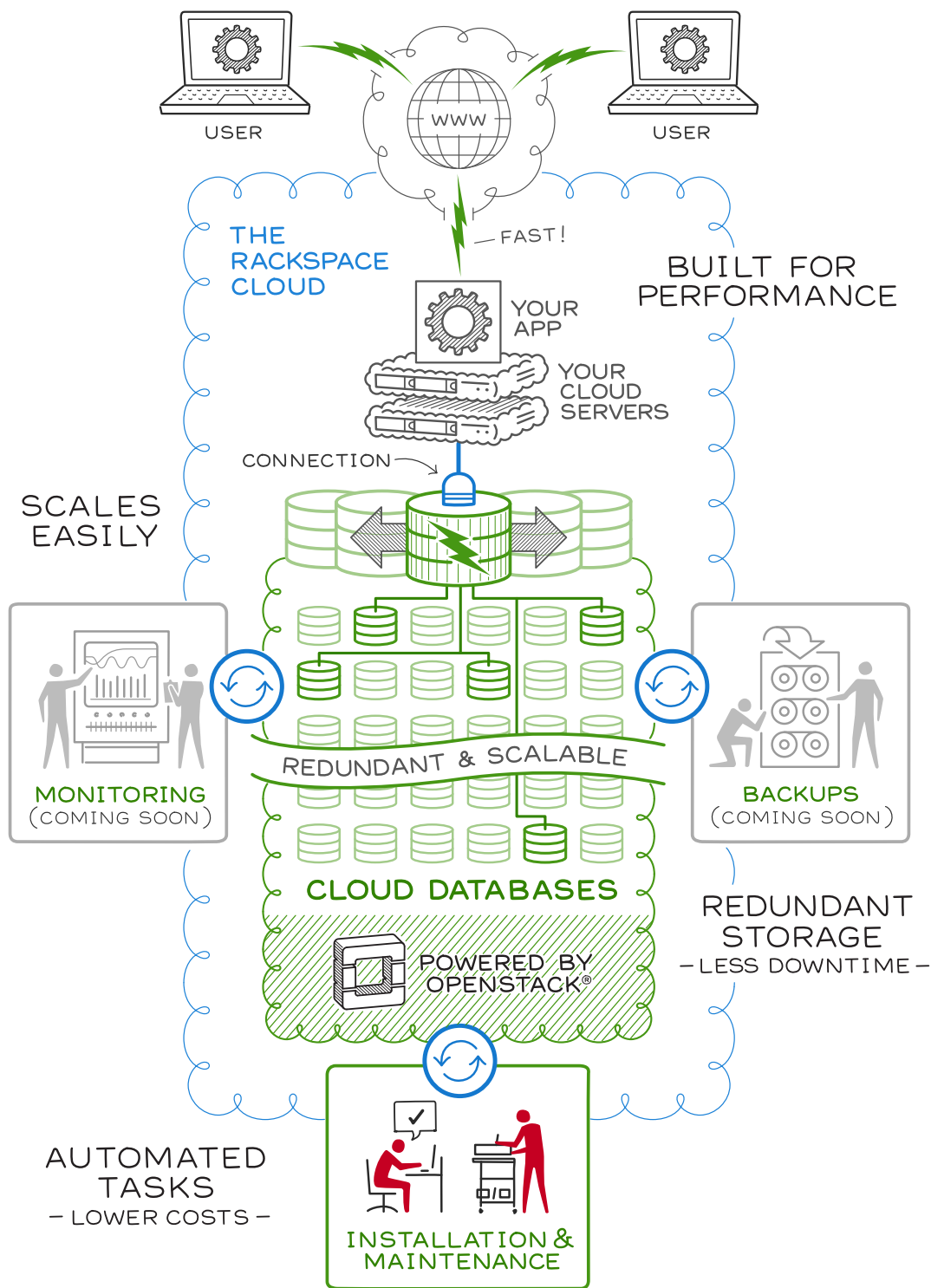
OpenStack Cloud Databases is an OpenStack-based MySQL relational database service that allows customers to easily provision database instances of varying virtual resource sizes without the need to maintain and/or update MySQL. Interactions with Cloud Databases occur programmatically via the Cloud Databases API as described in this developer guide.



Note

OpenStack recommends that Cloud Databases users back up their data using **mysqldump** until backups are supported in Cloud Databases.

The following figure shows an overview of Cloud Databases Infrastructure:



Reviewer: need to edit graphic above so it says "The Cloud" rather than "The Rackspace Cloud".

1.1. Intended Audience

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

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

1.2. Document Change History

This version of the Developer Guide replaces and obsoletes all previous versions. The most recent changes are described in the table below:

Revision Date	Summary of Changes
May 2, 2013	<ul style="list-style-type: none">• This document is for the initial OpenStack review.

1.3. Additional Resources

You can download the most current versions of this and other API-related documents from <http://docs.openstack.org/>.

We welcome feedback, comments, and bug reports at <https://bugs.launchpad.net/reddwarf>.

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

2. Concepts

To use the Cloud Databases API effectively, you should understand several key concepts:

2.1. Database Instance

A database instance is an isolated MySQL instance in a single tenant environment on a shared physical host machine.

2.2. Database

A MySQL database within a database instance.

2.3. Flavor

A flavor is an available hardware configuration for a database instance. Each flavor has a unique combination of memory capacity and priority for CPU time.

2.4. Volume

A volume is user-specified storage that contains the MySQL data directory. Volumes are automatically provisioned on shared Internet Small Computer System Interface (iSCSI) storage area networks (SAN) that provide for increased performance, scalability, availability and manageability. Applications with high I/O demands are performance optimized and data is protected through both local and network RAID-10. Additionally, network RAID provides synchronous replication of volumes with automatic failover and load balancing across available storage clusters.

3. General API Information

The Cloud Databases API is implemented using a ReSTful web service interface. Like other cloud products, the Database Service shares a common token-based authentication system that allows seamless access between products and services.



Note

All requests to authenticate against and operate the service are performed using SSL over HTTP (HTTPS) on TCP port 443.

3.1. Authentication

Each HTTP request against the Cloud Database service requires the inclusion of specific authentication credentials. A single deployment may support multiple authentication schemes (OAuth, Basic Auth, Token). The authentication scheme used is determined by the provider of the Cloud Database service. Please contact your provider to determine the best way to authenticate against this API.



Note

Some authentication schemes may require that the API operate using SSL over HTTP (HTTPS).

3.2. Cloud Databases Service Versions

The Cloud Databases version defines the contract and build information for the API.

3.2.1. Contract Version

The contract version denotes the data model and behavior that the API supports. The requested contract version is included in all request URLs. Different contract versions of the API may be available at any given time and are not guaranteed to be compatible with one another.

Example 3.1. Example Request URL (contract version in bold)

```
https://openstack.example.com/v1.0/1234
```



Note

This document pertains to contract version 1.0.

3.2.2. API Version

The API List Versions call is available to show the current API version as well as information about all versions of the API. Refer to [Section 4.1, “API Versions” \[16\]](#) for details.

3.3. Date/Time Format

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

The system timezone is in UTC. MySQL converts `TIMESTAMP` values from the current time zone to UTC for storage, and back from UTC to the current time zone for retrieval. This does not occur for other types, such as `DATETIME`.

Example 3.2. 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 3.1. 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

3.4. Pagination

To reduce load on the service, list operations return a maximum of 20 items at a time. This is referred to as *pagination*. Cloud Databases has separate paging limits for instances, databases, and users, which are currently all set to 20. If a request supplies no limit or one that exceeds the configured default limit, the default is used instead.

Pagination provides the ability to limit the size of the returned data as well as retrieve a specified subset of a large data set. Pagination has two key concepts: *limit* and *marker*. *Limit* is the restriction on the maximum number of items for that type that can be returned. *Marker* is the ID of the last item in the previous list returned. The ID is the UUID in the case of instances, and the name in the case of databases and users. For example, a query could request the next 10 instances after the instance "1234" as follows: `?limit=10&marker=1234`. Items are displayed sorted by ID.

Pagination applies only to the calls listed in the following table:

Verb	URI	Description
GET	/instances/	Lists the status and information for all database instances.

Verb	URI	Description
GET	/instances/{instanceId}/databases	Lists databases for the specified instance.
GET	/instances/{instanceId}/users	Lists the users in the specified database instance.

If the content returned by a call is paginated, the response includes a structured link much like an instance item's links, with the basic structure `{ "href": "<url>", "rel": "next" }`. Any response that is truncated by pagination will have a *next* link, which points to the next item in the collection. If there are no more items, no *next* link is returned.

See the examples of paged List Instances calls that follow.

Reviewer: Need new examples that show OpenStack host.

Example 3.3. List Instances Paged Request: XML

```
GET /v1.0/1234/instances?limit=2 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 3.4. List Instances Paged Request: JSON

```
GET /v1.0/1234/instances?limit=2 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

Notice that the paged request examples above set the limit to 2 (`?limit=2`), so the responses that follow each show 2 instances and return a *marker* set to the UUID of the last item in the returned list (`?marker=4137d6a4-03b7-4b66-b0ef-8c7c35c470d3`). Also a link is provided to retrieve the next 2 results (`limit=2`) in the link element identified by the attribute `rel="next"` (XML) or `"rel": "next"` (JSON):

Example 3.5. List Instances Paged Response: XML

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1538
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<instances xmlns="http://docs.openstack.org/database/api/v1.0">
  <instance id="098653ba-218b-47ce-936a-e0b749101f81" name=
"xml_rack_instance" status="ACTIVE">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="self"/>
      <link href="https://openstack.example.com/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="bookmark"/>
    </links>
    <volume size="2"/>
    <flavor id="1">
      <links>
        <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
        <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
      </links>
    </flavor>
  </instance>
  <instance id="44b277eb-39be-4921-be31-3d61b43651d7" name=
"json_rack_instance" status="ACTIVE">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/instances/
44b277eb-39be-4921-be31-3d61b43651d7" rel="self"/>
      <link href="https://openstack.example.com/instances/
44b277eb-39be-4921-be31-3d61b43651d7" rel="bookmark"/>
    </links>
    <volume size="2"/>
    <flavor id="1">
      <links>
        <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
        <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
      </links>
    </flavor>
  </instance>
  <links>
    <link href="https://openstack.example.com/v1.0/1234/instances?marker=
44b277eb-39be-4921-be31-3d61b43651d7&limit=2" rel="next"/>
  </links>
</instances>
```


3.5. Faults

When an error occurs, the Database 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.
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 <code>details</code> 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 Database Service is not available.

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

Example 3.7. 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

<instanceFault code="500" xmlns="http://docs.openstack.org/database/api/v1.0">
  <message>
    The server has either erred or is incapable of performing the
    requested operation.
  </message>
</instanceFault>
```

Example 3.8. 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 3.9. 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

<badRequest code="400" xmlns="http://docs.openstack.org/database/api/v1.0">
  <message>
    Volume 'size' needs to be a positive integer value, -1.0 cannot be
    accepted.
  </message>
</badRequest>
```

Example 3.10. 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 3.11. 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

<itemNotFound code="404" xmlns="http://docs.openstack.org/database/api/v1.0">
  <message>
    The resource could not be found.
  </message>
</itemNotFound>
```

Example 3.12. 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."
  }
}
```

3.5.1. Synchronous Versus Asynchronous Faults

Synchronous faults occur at request time. When a synchronous fault occurs, the fault contains an HTTP error response code, a human readable message, and optional details about the error. The following Database API calls are synchronous and may produce synchronous faults:

- List Users
- List Instances
- List Instance Details by ID
- List Databases
- Enable Root User
- List Root-Enabled Status
- List Flavors
- List Versions
- List Version Details

Asynchronous faults occur in the background while an instance, database, or user is being built or an instance is executing an action. When an asynchronous fault occurs, the system places the instance, database, or user in an ERROR state and embeds the fault in the offending instance, database, or user. When an asynchronous fault occurs, the fault contains an HTTP error response code, a human readable message, and optional details about the error. The following Database API calls are asynchronous and may produce asynchronous faults:

- Create Instance
- Delete Instance
- Create Database
- Delete Database
- Create User
- Delete User
- Resize Volume
- Resize Instance
- Restart Instance



Note

Note that an asynchronous operation, if it fails, may not give the user an error, and the operation can error out without a failure notification.

3.6. Database Instance Status

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

- BUILD – The database instance is being provisioned.
- REBOOT – The database instance is rebooting.
- ACTIVE – The database instance is online and available to take requests.
- BLOCKED – The database instance is unresponsive at the moment.
- RESIZE – The database instance is being resized at the moment.
- SHUTDOWN – The database instance is terminating services. Also, SHUTDOWN is returned if for any reason the MySQL instance is shut down but not the actual server.



Note

If MySQL has crashed (causing the SHUTDOWN status), please call support for assistance.

- ERROR – The last operation for the database instance failed due to an error.

4. API Operations



Note

Do not use trailing slashes (/) at the end of calls to API operations, since this may cause the call to fail. For example, do not use **GET** /instances/detail/ (with the trailing slash at the end). Rather, use **GET** /instances/detail instead.

Method	URI	Description
API Versions		
GET	/	Lists information about all versions of the API.
GET	{/version}	Returns detailed information about the specified version of the API.
Database Instances		
POST	/instances	Creates a new <i>database instance</i> .
GET	/instances	Lists the status and information for all database instances.
GET	/instances{/instanceId}	Lists status and details for a specified database instance.
DELETE	/instances{/instanceId}	Deletes the specified database instance.
POST	/instances{/instanceId}/root	Enables the root user for the specified database instance and returns the root password.
GET	/instances{/instanceId}/root	Returns true if root user is enabled for the specified database instance or false otherwise.
Database Instance Actions		
POST	/instances{/instanceId}/action	Restart the database service on the instance.
POST	/instances{/instanceId}/action	Resize the memory of the instance.
POST	/instances{/instanceId}/action	Resize the <i>volume</i> attached to the Instance.
Databases		
POST	/instances{/instanceId}/databases	Creates a new <i>database</i> within the specified instance.
GET	/instances{/instanceId}/databases	Lists databases for the specified instance.
DELETE	/instances{/instanceId}/ databases{/databaseName}	Deletes the specified database.
Users		
POST	/instances{/instanceId}/users	Creates a user for the specified database instance.
GET	/instances{/instanceId}/users	Lists the users in the specified database instance.
PUT	/instances{/instanceId}/users	Changes the MySQL password of one or more users.
GET	/instances{/instanceId}/users{/ name}	Lists the specified user's name and a list of databases that the user can access.
DELETE	/instances{/instanceId}/users{/ name}	Deletes the user identified by {name} for the specified database instance.
Flavors		
GET	/flavors	Lists information for all available <i>flavors</i> .
GET	/flavors{/flavorId}	Lists all flavor information about the specified flavor ID.

4.1. API Versions

This section describes the versions that are supported for the Cloud Databases API.

Method	URI	Description
GET	/	Lists information about all versions of the API.
GET	{/version}	Returns detailed information about the specified version of the API.

4.1.1. List Versions

This operation lists information about all versions of the API.

Method	URI	Description
GET	/	Lists information about all versions of the API.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.1.1.1. Request

Example 4.1. List Versions Request: XML

The following examples show the List Versions requests:

```
GET / HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.2. List Versions Request: JSON

```
GET / HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.1.1.2. Response

Example 4.3. List Versions Response: XML

The following examples show the List Versions responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 261
Date: Wed, 25 Jan 2012 21:53:04 GMT

<versions xmlns="http://docs.openstack.org/database/api/v1.0">
  <version id="v1.0" status="CURRENT" updated="2012-01-01T00:00:00Z">
```

```
<links>
  <link href="https://openstack.example.com/v1.0/" rel="self"/>
</links>
</version>
</versions>
```

Example 4.4. List Versions Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 153
Date: Wed, 25 Jan 2012 21:53:04 GMT

{
  "versions": [
    {
      "id": "v1.0",
      "links": [
        {
          "href": "https://openstack.example.com/v1.0/",
          "rel": "self"
        }
      ],
      "status": "CURRENT",
      "updated": "2012-01-01T00:00:00Z"
    }
  ]
}
```

4.1.2. List Version Details

This operation returns detailed information about the specified version of the API.

Method	URI	Description
GET	{/version}	Returns detailed information about the specified version of the API.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.1.2.1. Request

Example 4.5. List Version Details Request: XML

The following examples show the List Version Details requests:

```
GET /v1.0/ HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.6. List Version Details Request: JSON

```
GET /v1.0/ HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.1.2.2. Response

Example 4.7. List Version Details Response: XML

The following examples show the List Version Details responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 218
Date: Wed, 25 Jan 2012 21:53:04 GMT

<version id="v1.0" status="CURRENT" updated="2012-01-01T00:00:00Z" xmlns=
"http://docs.openstack.org/database/api/v1.0">
```

```
<links>
  <link href="https://openstack.example.com/v1.0/" rel="self"/>
</links>
</version>
```

Example 4.8. List Version Details Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 150
Date: Wed, 25 Jan 2012 21:53:04 GMT

{
  "version": {
    "id": "v1.0",
    "links": [
      {
        "href": "https://openstack.example.com/v1.0/",
        "rel": "self"
      }
    ],
    "status": "CURRENT",
    "updated": "2012-01-01T00:00:00Z"
  }
}
```

4.2. Database Instances

This section describes the operations that are supported for database instances.

Method	URI	Description
POST	/instances	Creates a new <i>database instance</i> .
GET	/instances	Lists the status and information for all database instances.
GET	/instances{/instanceId}	Lists status and details for a specified database instance.
DELETE	/instances{/instanceId}	Deletes the specified database instance.
POST	/instances{/instanceId}/root	Enables the root user for the specified database instance and returns the root password.
GET	/instances{/instanceId}/root	Returns true if root user is enabled for the specified database instance or false otherwise.

4.2.1. Create Database Instance

This operation asynchronously provisions a new database instance. This call requires the user to specify a *flavor* and a *volume* size. The service then provisions the instance with the requested flavor and sets up a volume of the specified size, which is the storage for the database instance.




Notes

- You can create only one database instance per **POST** request.
- You can create a database instance with one or more databases, and users associated to those databases.
- The default binding for the MySQL instance is port 3306.

The following table lists the required and optional attributes for Create Instance:

Table 4.1. Required and Optional Attributes for Create Instance

Applies To	Name	Description	Required
Instance	flavorRef	Reference (href) to a flavor as specified in the response from the List Flavors API call. This is the actual URI as specified by the href field in the link. Refer to the List Flavors response examples that follow for an example of the flavorRef.  Note Rather than the flavor URI, you can also pass the flavor id (integer) as the value for flavorRef. Refer to List Flavors [75] for details.	Yes
	(volume) size	Specifies the volume size in gigabytes (GB). The value specified must be between 1 and 50.	Yes
	name	Name of the instance to create. The length of the name is limited to 255 characters and any characters are permitted.	No
Database	name	Specifies <i>database</i> names for creating databases on instance creation. Refer to Create Database [44] for the required xml/json format.	No
	character_set	Set of symbols and encodings. The default character set is <code>utf8</code> .	No
	collate	Set of rules for comparing characters in a character set. The default value for collate is <code>utf8_general_ci</code> .	No
User	name	Specifies user name for the database on instance creation. Refer to Create User [53] for the required xml/json format.	No
	password	Specifies password for those users on instance creation. Refer to Create User [53] for the required xml/json format.	No
	(database) name	Specifies names of databases that those users can access on instance creation. Refer to Create User [53] for the required xml/json format.	No

Refer to [Section 3.6, “Database Instance Status” \[14\]](#) for a list of possible database instance statuses that may be returned.

Method	URI	Description
POST	/instances	Creates a new <i>database instance</i> .

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.2.1.1. Request

This table shows the URI parameters for the Create Database Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.

Example 4.9. Create Database Instance Request: XML

The following examples show the Create Database Instance requests and responses:

```
POST /v1.0/1234/instances HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<instance xmlns="http://docs.openstack.org/database/api/v1.0" flavorRef="1"
  name="xml_rack_instance">
  <volume size="2"/>
  <users>
    <user password="demopassword" name="demouser">
      <databases>
        <database name="sampledb"/>
      </databases>
    </user>
  </users>
  <databases>
    <database collate="utf8_general_ci" name="sampledb" character_set="utf8"/>
    <database name="nextround"/>
  </databases>
</instance>
```

Example 4.10. Create Database Instance Request: JSON

```
POST /v1.0/1234/instances HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json

{
```

```
{
  "instance": {
    "databases": [
      {
        "character_set": "utf8",
        "collate": "utf8_general_ci",
        "name": "sampledb"
      },
      {
        "name": "nextround"
      }
    ],
    "flavorRef": 1,
    "name": "json_rack_instance",
    "users": [
      {
        "databases": [
          {
            "name": "sampledb"
          }
        ],
        "name": "demouser",
        "password": "demopassword"
      }
    ],
    "volume": {
      "size": 2
    }
  }
}
```

4.2.1.2. Response

Example 4.11. Create Database Instance Response: XML

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 724
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<instance created="2013-03-18 19:09:17.441489" id="098653ba-218b-47ce-936a-e0b749101f81" name="xml_rack_instance" status="BUILD" updated="2013-03-18 19:09:17.441606" xmlns="http://docs.openstack.org/database/api/v1.0">
  <links>
    <link href="https://openstack.example.com/v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81" rel="self"/>
    <link href="https://openstack.example.com/instances/098653ba-218b-47ce-936a-e0b749101f81" rel="bookmark"/>
  </links>
  <volume size="2"/>
  <flavor id="1">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/1" rel="self"/>
      <link href="https://openstack.example.com/flavors/1" rel="bookmark"/>
    </links>
  </flavor>
</instance>
```

Example 4.12. Create Database Instance Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 591
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "instance": {
    "created": "2013-03-18T19:09:17",
    "flavor": {
      "id": "1",
      "links": [
        {
          "href": "https://openstack.example.com/v1.0/1234/flavors/1",
          "rel": "self"
        },
        {
          "href": "https://openstack.example.com/flavors/1",
          "rel": "bookmark"
        }
      ]
    },
    "id": "44b277eb-39be-4921-be31-3d61b43651d7",
    "links": [
      {
        "href": "https://openstack.example.com/v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7",
        "rel": "self"
      },
      {
        "href": "https://openstack.example.com/instances/44b277eb-39be-4921-be31-3d61b43651d7",
        "rel": "bookmark"
      }
    ],
    "name": "json_rack_instance",
    "status": "BUILD",
    "updated": "2013-03-18T19:09:17",
    "volume": {
      "size": 2
    }
  }
}
```

For convenience, notice in the response examples above that resources contain links to themselves. This allows a client to easily obtain resource URIs rather than to construct them. There are two kinds of link relations associated with resources. A `self` link contains a *versioned* link to the resource. These links should be used in cases where the link will be followed immediately. A `bookmark` link provides a permanent link to a resource that is appropriate for long term storage.

4.2.2. List All Database Instances

This operation lists the status and information for all database instances.

Refer to [Section 3.6, “Database Instance Status” \[14\]](#) for a list of possible database instance statuses that may be returned.

Method	URI	Description
GET	/instances	Lists the status and information for all database instances.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.2.2.1. Request

This table shows the URI parameters for the List All Database Instances Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.

Example 4.13. List All Database Instances Request: XML

The following examples show the List All Database Instances Detail requests:

```
GET /v1.0/1234/instances HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.14. List All Database Instances Request: JSON

```
GET /v1.0/1234/instances HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.2.2.2. Response

Example 4.15. List All Database Instances Response: XML

The following examples show the List All Database Instances responses:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml
Content-Length: 1380
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<instances xmlns="http://docs.openstack.org/database/api/v1.0">
  <instance id="098653ba-218b-47ce-936a-e0b749101f81" name=
"xml_rack_instance" status="ACTIVE">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="self"/>
      <link href="https://openstack.example.com/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="bookmark"/>
    </links>
    <volume size="2"/>
    <flavor id="1">
      <links>
        <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
        <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
      </links>
    </flavor>
  </instance>
  <instance id="44b277eb-39be-4921-be31-3d61b43651d7" name=
"json_rack_instance" status="ACTIVE">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/instances/
44b277eb-39be-4921-be31-3d61b43651d7" rel="self"/>
      <link href="https://openstack.example.com/instances/
44b277eb-39be-4921-be31-3d61b43651d7" rel="bookmark"/>
    </links>
    <volume size="2"/>
    <flavor id="1">
      <links>
        <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
        <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
      </links>
    </flavor>
  </instance>
</instances>
```

Example 4.16. List All Database Instances Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1038
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "instances": [
    {
      "flavor": {
        "id": "1",
        "links": [
          {
```

```
        "href": "https://openstack.example.com/v1.0/1234/
flavors/1",
        "rel": "self"
    },
    {
        "href": "https://openstack.example.com/flavors/1",
        "rel": "bookmark"
    }
]
},
"id": "098653ba-218b-47ce-936a-e0b749101f81",
"links": [
    {
        "href": "https://openstack.example.com/v1.0/1234/
instances/098653ba-218b-47ce-936a-e0b749101f81",
        "rel": "self"
    },
    {
        "href": "https://openstack.example.com/instances/
098653ba-218b-47ce-936a-e0b749101f81",
        "rel": "bookmark"
    }
],
"name": "xml Rack Instance",
"status": "ACTIVE",
"volume": {
    "size": 2
}
},
{
    "flavor": {
        "id": "1",
        "links": [
            {
                "href": "https://openstack.example.com/v1.0/1234/
flavors/1",
                "rel": "self"
            },
            {
                "href": "https://openstack.example.com/flavors/1",
                "rel": "bookmark"
            }
        ]
    },
    "id": "44b277eb-39be-4921-be31-3d61b43651d7",
    "links": [
        {
            "href": "https://openstack.example.com/v1.0/1234/
instances/44b277eb-39be-4921-be31-3d61b43651d7",
            "rel": "self"
        },
        {
            "href": "https://openstack.example.com/instances/
44b277eb-39be-4921-be31-3d61b43651d7",
            "rel": "bookmark"
        }
    ],
    "name": "JSON Rack Instance",
    "status": "ACTIVE",
    "volume": {
```

```
}  
  ]  
    }  
      }  
        "size": 2
```

4.2.3. List Database Instance Status and Details

This operation lists the status and details of the specified database instance.

This operation lists the volume size in gigabytes (GB) and the approximate GB used.



Note

After instance creation, the `used` size of your volume will be greater than 0. This is expected and due to the automatic creation of non-empty transaction logs for mysql optimization. The `used` attribute is *not* returned in the response when the status for the instance is BUILD, REBOOT, RESIZE, or ERROR.

Refer to [Section 3.6, “Database Instance Status” \[14\]](#) for a list of possible database instance statuses that may be returned.

The list operations return a DNS-resolvable hostname associated with the database instance instead of an IP address. Since the hostname always resolves to the correct IP address of the database instance, this relieves the user from the task of maintaining the mapping. Note that although the IP address may likely change on resizing, migrating, and so forth, the hostname always resolves to the correct database instance.

Method	URI	Description
GET	/instances{/instanceId}	Lists status and details for a specified database instance.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.2.3.1. Request

This table shows the URI parameters for the List Database Instance Status and Details Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.17. List Database Instance Status and Details Request: XML

The following examples show the List Database Instance Status and Details requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.18. List Database Instance Status and Details Request: JSON

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

This operation does not require a request body.

4.2.3.2. Response

Example 4.19. List Database Instance Status and Details Response: XML

The following examples show the List Database Instance Status and Details responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 747
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<instance created="2013-03-18 19:09:17.441489" id="098653ba-218b-47ce-936a-
e0b749101f81" name="xml Rack Instance" status="ACTIVE" updated="2013-03-18
19:09:17.513134" xmlns="http://docs.openstack.org/database/api/v1.0">
  <links>
    <link href="https://openstack.example.com/v1.0/1234/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="self"/>
    <link href="https://openstack.example.com/instances/
098653ba-218b-47ce-936a-e0b749101f81" rel="bookmark"/>
  </links>
  <volume size="2" used="0.163685983978"/>
  <flavor id="1">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
      <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
    </links>
  </flavor>
</instance>
```

Example 4.20. List Database Instance Status and Details Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 621
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "instance": {
    "created": "2013-03-18T19:09:17",
    "flavor": {
      "id": "1",
```

```
        "links": [
            {
                "href": "https://openstack.example.com/v1.0/1234/flavors/
1",
                "rel": "self"
            },
            {
                "href": "https://openstack.example.com/flavors/1",
                "rel": "bookmark"
            }
        ]
    },
    "id": "44b277eb-39be-4921-be31-3d61b43651d7",
    "links": [
        {
            "href": "https://openstack.example.com/v1.0/1234/instances/
44b277eb-39be-4921-be31-3d61b43651d7",
            "rel": "self"
        },
        {
            "href": "https://openstack.example.com/instances/
44b277eb-39be-4921-be31-3d61b43651d7",
            "rel": "bookmark"
        }
    ],
    "name": "json_rack_instance",
    "status": "ACTIVE",
    "updated": "2013-03-18T19:09:17",
    "volume": {
        "size": 2,
        "used": 0.16368598397821188
    }
}
```

4.2.4. Delete Database Instance

This operation deletes the specified database instance, including any associated data.

Refer to [Section 3.6, “Database Instance Status” \[14\]](#) for a list of possible database instance statuses that may be returned.



Note

This operation is not allowed when the instance status is `BUILD`.

Method	URI	Description
DELETE	/instances{/instanceId}	Deletes the specified database instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), unprocessableEntity (422), itemNotFound (404)

4.2.4.1. Request

This table shows the URI parameters for the Delete Database Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.21. Delete Database Instance Request: XML

The following examples show the Delete Database Instance requests:

```
DELETE /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.22. Delete Database Instance Request: JSON

```
DELETE /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.2.4.2. Response

This operation does not return a response body.

4.2.5. Enable Root User

This operation enables login from any host for the root user and provides the user with a generated root password.



Note

Changes you make as a root user may cause detrimental effects to the database instance and unpredictable behavior for API operations. When you enable the root user, you accept the possibility that we will not be able to support your database instance. While enabling root does not prevent us from a “best effort” approach to helping you if something goes wrong with your instance, we cannot ensure that we will be able to assist you if you change core MySQL settings. These changes can be (but are not limited to) turning off binlogs, removing users that we use to access your instance, and so forth.

Method	URI	Description
POST	/instances{/instanceId}/root	Enables the root user for the specified database instance and returns the root password.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.2.5.1. Request

This table shows the URI parameters for the Enable Root User Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.23. Enable Root User Request: XML

The following examples show the Enable Root User requests:

```
POST /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/root HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.24. Enable Root User Request: JSON

```
POST /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/root HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
```

```
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.2.5.2. Response

Example 4.25. Enable Root User Response: XML

The following examples show the Enable Root User responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 89
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<user name="root" password="12345" xmlns="http://docs.openstack.org/database/
api/v1.0"/>
```

Example 4.26. Enable Root User Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 47
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "user": {
    "name": "root",
    "password": "12345"
  }
}
```

4.2.6. List Root-Enabled Status

This operation checks an active specified database instance to see if root access is enabled. It returns True if root user is enabled for the specified database instance or False otherwise.

Method	URI	Description
GET	/instances{/instanceId}/root	Returns true if root user is enabled for the specified database instance or false otherwise.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.2.6.1. Request

This table shows the URI parameters for the List Root-Enabled Status Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.27. List Root-Enabled Status Request: XML

The following examples show the Check Root User Access requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/root HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.28. List Root-Enabled Status Request: JSON

```
GET /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/root HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.2.6.2. Response

Example 4.29. List Root-Enabled Status Response: XML

The following examples show the Check Root User Access responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 90
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<rootEnabled xmlns="http://docs.openstack.org/database/api/v1.0">
  True
</rootEnabled>
```

Example 4.30. List Root-Enabled Status Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 21
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "rootEnabled": true
}
```

4.3. Database Instance Actions

This section describes the actions that are supported for database instances.

Method	URI	Description
POST	/instances{/instanceId}/action	Restart the database service on the instance.
POST	/instances{/instanceId}/action	Resize the memory of the instance.
POST	/instances{/instanceId}/action	Resize the <i>volume</i> attached to the Instance.

4.3.1. Restart Instance

The restart operation will restart only the MySQL Instance. Restarting MySQL will erase any dynamic configuration settings that you have made within MySQL.



Note

The MySQL service will be unavailable until the instance restarts.

This operation returns a 202 Accepted response.

Method	URI	Description
POST	/instances{/instanceId}/action	Restart the database service on the instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

4.3.1.1. Request

This table shows the URI parameters for the Restart Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.31. Restart Instance Request: XML

The following examples show the Restart Instance requests:

```
POST /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/action HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<restart xmlns="http://docs.openstack.org/database/api/v1.0"/>
```

Example 4.32. Restart Instance Request: JSON

```
POST /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/action HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json

{
  "restart": {}
}
```

```
}
```

4.3.1.2. Response

Example 4.33. Restart Instance Response: XML

The following examples show the Restart Instance responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:18 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.34. Restart Instance Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:18 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

4.3.2. Resize the Instance

This operation changes the memory size of the instance, assuming a valid flavorRef is provided. Restarts MySQL in the process.

Method	URI	Description
POST	/instances{/instanceId}/action	Resize the memory of the instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

4.3.2.1. Request

This table shows the URI parameters for the Resize the Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.35. Resize the Instance Request: XML

The following examples show the Resize Instance requests:

```
POST /v1.0/1234/instances/5d891bb6-6c61-4b0a-8b85-26f4ee461c9d/action HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 2eeb3252-0164-40f5-8fb7-85df5faa2698
Accept: application/xml
Content-Type: application/xml

<?xml version="1.0" ?>
<resize xmlns="http://docs.openstack.org/database/api/v1.0">
  <flavorRef>https://openstack.example.com/v1.0/1234/flavors/2</
flavorRef>
</resize>
```

Example 4.36. Resize the Instance Request: JSON

```
POST /v1.0/1234/instances/23a3d4fb-3731-497b-afd4-bf25bde2b5fc/action HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 2eeb3252-0164-40f5-8fb7-85df5faa2698
Accept: application/json
Content-Type: application/json

{
  "resize": {
    "flavorRef": "https://openstack.example.com/v1.0/1234/flavors/2"
  }
}
```


4.3.2.2. Response

Example 4.37. Resize the Instance Response: XML

The following examples show the Resize Instance responses:

```
HTTP/1.1 202 Accepted
Content-Type: text/plain; charset=UTF-8
Content-Length: 58
Date: Mon, 06 Feb 2012 21:28:11 GMT
```

Example 4.38. Resize the Instance Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: text/plain; charset=UTF-8
Content-Length: 58
Date: Mon, 06 Feb 2012 21:28:10 GMT
```

4.3.3. Resize the Instance Volume

This operation supports resizing the attached volume for an instance. It supports only increasing the volume size and does not support decreasing the size. The volume size is in gigabytes (GB) and must be an integer.



Note

You cannot increase the volume to a size larger than the API volume size limit specifies.

This operation returns a 202 Accepted response.

Method	URI	Description
POST	/instances{/instanceId}/action	Resize the <i>volume</i> attached to the Instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

4.3.3.1. Request

This table shows the URI parameters for the Resize the Instance Volume Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.39. Resize the Instance Volume Request: XML

The following examples show the Resize Instance Volume requests:

```
POST /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/action HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<resize xmlns="http://docs.openstack.org/database/api/v1.0">
  <volume size="4"/>
</resize>
```

Example 4.40. Resize the Instance Volume Request: JSON

```
POST /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/action HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
```

```
Content-Type: application/json

{
  "resize": {
    "volume": {
      "size": 4
    }
  }
}
```

4.3.3.2. Response

Example 4.41. Resize the Instance Volume Response: XML

The following examples show the Resize Instance Volume responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:18 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.42. Resize the Instance Volume Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:18 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

4.4. Databases

This section describes the operations that are supported for databases.

Method	URI	Description
POST	/instances{/instanceId}/databases	Creates a new <i>database</i> within the specified instance.
GET	/instances{/instanceId}/databases	Lists databases for the specified instance.
DELETE	/instances{/instanceId}/ databases{/databaseName}	Deletes the specified database.

4.4.1. Create Database

This operation creates a new database within the specified instance.

The `name` of the database is a required attribute.

The following additional attributes can be specified for each database: `collate` and `character_set`.

Table 4.2. Required and Optional Attributes for Create Database

Name	Description	Required
<code>name</code>	Specifies the database name for creating the database. Refer to the request examples for the required <code>xml/json</code> format.	Yes
<code>character_set</code>	Set of symbols and encodings. The default character set is <code>utf8</code> .	No
<code>collate</code>	Set of rules for comparing characters in a character set. The default value for <code>collate</code> is <code>utf8_general_ci</code> .	No

See the MySQL documentation for information about supported character sets and collations at <http://dev.mysql.com/doc/refman/5.1/en/charset-mysql.html>.



Note

The following database names are reserved and cannot be used for creating databases: `lost+found`, `information_schema`, and `mysql`.

Refer to the following tables for information about characters that are valid/invalid for creating database names.

Table 4.3. Valid Characters That Can Be Used in a Database Name

Character
Letters (upper and lower cases allowed)
Numbers
'@', '?', '#', and spaces are allowed, but <i>not</i> at the beginning and end of the database name
'_' is allowed anywhere in the database name

Table 4.4. Characters That *Cannot* Be Used in a Database Name

Character
Single quotes
Double quotes
Back quotes
Semicolons
Commas
Backslashes
Forwardslashes

Table 4.5. Length Restrictions for Database Name

Restriction	Value
Database-name maximum length	64

Method	URI	Description
POST	/instances{/instanceId}/databases	Creates a new <i>database</i> within the specified instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.4.1.1. Request

This table shows the URI parameters for the Create Database Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.43. Create Database Request: XML

The following examples show the Create Database requests:

```
POST /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/databases HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<databases xmlns="http://docs.openstack.org/database/api/v1.0">
  <database collate="utf8_general_ci" name="testingdb" character_set="utf8"/>
  <database name="anotherdb"/>
  <database name="oneMoreDB"/>
</databases>
```

Example 4.44. Create Database Request: JSON

```
POST /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/databases HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json

{
  "databases": [
    {
      "character_set": "utf8",
      "collate": "utf8_general_ci",
```

```
        "name": "testingdb"
      },
      {
        "name": "anotherdb"
      },
      {
        "name": "oneMoreDB"
      }
    ]
  }
```

4.4.1.2. Response

Example 4.45. Create Database Response: XML

The following examples show the Create Database responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.46. Create Database Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

4.4.2. List Databases for Instance

This operation lists the databases for the specified instance.



Note

This operation returns only the user-defined databases, not the system databases. The system databases (mysql, information_schema, lost+found) can only be viewed by a database administrator.

Method	URI	Description
GET	/instances{/instanceId}/databases	Lists databases for the specified instance.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.4.2.1. Request

This table shows the URI parameters for the List Databases for Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.47. List Databases for Instance Request: XML

The following examples show the List Databases for Instance requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/databases HTTP/
1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.48. List Databases for Instance Request: JSON

```
GET /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/databases HTTP/
1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

Example 4.49. List Databases for Instance Paged Request: XML

The following examples show the *paginated* List Databases for Instance requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/databases?limit=
2 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.50. List Databases for Instance Paged Request: JSON

```
GET /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/databases?limit=
1 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.4.2.2. Response

Example 4.51. List Databases for Instance Response: XML

The following examples show the List Databases for Instance responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 241
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<databases xmlns="http://docs.openstack.org/database/api/v1.0">
  <database name="anotherdb"/>
  <database name="nextround"/>
  <database name="oneMoreDB"/>
  <database name="sampledb"/>
  <database name="testingdb"/>
</databases>
```

Example 4.52. List Databases for Instance Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 129
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
```



```
"databases": [
  {
    "name": "anotherdb"
  },
  {
    "name": "nextround"
  },
  {
    "name": "oneMoreDB"
  },
  {
    "name": "sampledb"
  },
  {
    "name": "testingdb"
  }
]
```

Example 4.53. List Databases for Instance Paged Response: XML

The following examples show the *paginated* List Databases for Instance responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 321
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<databases xmlns="http://docs.openstack.org/database/api/v1.0">
  <database name="anotherdb"/>
  <database name="nextround"/>
  <links>
    <link href="https://openstack.example.com/v1.0/1234/instances/
098653ba-218b-47ce-936a-e0b749101f81/databases?marker=nextround&limit=2"
rel="next"/>
  </links>
</databases>
```

Example 4.54. List Databases for Instance Paged Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 192
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "databases": [
    {
      "name": "anotherdb"
    }
  ],
  "links": [
    {
      "href": "https://openstack.example.com/v1.0/1234/instances/
44b277eb-39be-4921-be31-3d61b43651d7/databases?marker=anotherdb&limit=1",
      "rel": "next"
    }
  ]
}
```

```
} ]
```

4.4.3. Delete Database

This operation deletes the requested database within the specified database instance. Note that all data associated with the database is also deleted.

Method	URI	Description
DELETE	/instances{/instanceId}/databases{/databaseName}	Deletes the specified database.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.4.3.1. Request

This table shows the URI parameters for the Delete Database Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{databaseName}	String	The name for the specified database.

Example 4.55. Delete Database Request: XML

The following examples show the Delete Database requests:

```
DELETE /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/databases/
oneMoreDB HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.56. Delete Database Request: JSON

```
DELETE /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/databases/
testingdb HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.4.3.2. Response

Example 4.57. Delete Database Response: XML

The following examples show the Delete Database responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.58. Delete Database Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

4.5. Users

This section describes the operations that are supported for managing database users.

In this section, "user has access to a database" means that the user has full create, read, update, and delete access to the given database. In other words, on a cloud database instance, a user named USER and a database named DATABASE exist, and within MySQL, a GRANT ALL ON DATABASE.* TO USER has been issued on the instance.



Warning

There is a bug in a python library that development is using that may cause incorrect user deletions to occur if a period (.) is used in the user name. In this case, the user name is truncated to remove the portion of the name from the period to the end, leaving only the portion from the beginning up to the period. For example, for a user named "my.userA", the bug would truncate the user name to "my", and if the user "my" exists, that user will be incorrectly deleted. To avoid the problem, do not use periods in user names.

Method	URI	Description
POST	/instances{/instanceId}/users	Creates a user for the specified database instance.
GET	/instances{/instanceId}/users	Lists the users in the specified database instance.
PUT	/instances{/instanceId}/users	Changes the MySQL password of one or more users.
GET	/instances{/instanceId}/users{/name}	Lists the specified user's name and a list of databases that the user can access.
DELETE	/instances{/instanceId}/users{/name}	Deletes the user identified by {name} for the specified database instance.

4.5.1. Create User

This operation asynchronously provisions a new user for the specified database instance based on the configuration defined in the request object. Once the request is validated and progress has started on the provisioning process, a 202 Accepted response object is returned.

Writer: please add the following note back into the doc once the List User Details call is added back into the API: Using the identifier, the caller can check on the progress of the operation by performing a GET on users/name (for more details on this operation see the "List User Details" section of this document).

If the corresponding request cannot be fulfilled due to insufficient or invalid data, an HTTP 400 "Bad Request" error response is returned with information regarding the nature of the failure. Failures in the validation process are non-recoverable and require the caller to correct the cause of the failure and POST the request again.

The following table lists the required attributes for Create User. Refer to the request examples for the required xml/json format:

Table 4.6. Required Attributes for Create User

Applies To	Name	Description	Required
User	name	Name of the user for the database.	Yes
	password	User password for database access.	Yes
	(database) name	Name of the database that the user can access. One or more database names must be specified.	No



Notes

- A user is granted all privileges on the specified databases.
- The following user name is reserved and cannot be used for creating users: root.

Refer to the following tables for information about characters that are valid/invalid for creating database names, user names, and passwords.

Table 4.7. Valid Characters That Can Be Used in a Database Name, User Name, and Password

Character
Letters (upper and lower cases allowed)
Numbers
'@', '?', '#', and spaces are allowed, but <i>not</i> at the beginning and end of the database name, user name, and password
"_" is allowed anywhere in the database name, user name, and password

Table 4.8. Characters That *Cannot* Be Used in a Database Name, User Name, and Password

Character
Single quotes
Double quotes
Back quotes
Semicolons
Commas
Backslashes
Forwardslashes
Spaces at the front or end of the user name or password

Table 4.9. Length Restrictions for Database Name, User Name, and Password

Restriction	Value
Database name maximum length	64
User name maximum length	16
Password maximum length	unlimited (no restrictions)

Method	URI	Description
POST	/instances{/instanceId}/users	Creates a user for the specified database instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.1.1. Request

This table shows the URI parameters for the Create User Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.59. Create User Request: XML

The following examples show the Create User requests:

```
POST /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/users HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<users xmlns="http://docs.openstack.org/database/api/v1.0">
  <user password="password" name="dbuser1" database="databaseA"/>
  <user password="password" name="dbuser2">
    <databases>
```

```
<database name="databaseB"/>
<database name="databaseC"/>
</databases>
</user>
<user password="password" name="dbuser3" database="databaseD"/>
</users>
```

Example 4.60. Create User Request: JSON

```
POST /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/users HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

```
{
  "users": [
    {
      "database": "databaseA",
      "name": "dbuser1",
      "password": "password"
    },
    {
      "databases": [
        {
          "name": "databaseB"
        },
        {
          "name": "databaseC"
        }
      ],
      "name": "dbuser2",
      "password": "password"
    },
    {
      "database": "databaseD",
      "name": "dbuser3",
      "password": "password"
    }
  ]
}
```

4.5.1.2. Response

Example 4.61. Create User Response: XML

The following examples show the Create User responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.62. Create User Response: JSON

```
HTTP/1.1 202 Accepted
```

```
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```


4.5.2. List Users in Database Instance

This operation lists the users in the specified database instance, along with the associated databases for that user.



Note

This operation does not return the system users (database administrators that administer the health of the database). Also, this operation returns the "root" user only if "root" user has been enabled.

The following notes apply to MySQL users:

- User names can be up to 16 characters long.
- When you create accounts with INSERT, you must use FLUSH PRIVILEGES to tell the server to reload the grant tables.
- For additional information, refer to: <http://dev.mysql.com/doc/refman/5.1/en/user-account-management.html>

Method	URI	Description
GET	/instances{/instanceId}/users	Lists the users in the specified database instance.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.2.1. Request

This table shows the URI parameters for the List Users in Database Instance Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.63. List Users in Database Instance Request: XML

The following examples show the List Users in Database Instance requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/users HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.64. List Users in Database Instance Request: JSON

```
GET /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/users HTTP/1.1
```

```
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

Example 4.65. List Users in Database Instance Paged Request: XML

The following examples show the *paginated* List Users in Database Instance requests:

```
GET /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/users?limit=2
HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.66. List Users in Database Instance Paged Request: JSON

```
GET /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/users?limit=2
HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.5.2.2. Response

Example 4.67. List Users in Database Instance Response: XML

The following examples show the List Users in Database Instance responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 468
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<users xmlns="http://docs.openstack.org/database/api/v1.0">
  <user name="dbuser1">
    <databases/>
  </user>
  <user name="dbuser2">
    <databases>
      <database name="databaseB"/>
      <database name="databaseC"/>
    </databases>
  </user>
  <user name="dbuser3">
    <databases/>
  </user>
  <user name="demouser">
    <databases>
      <database name="sampledb"/>
    </databases>
  </user>
</users>
```

Example 4.68. List Users in Database Instance Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 228
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "users": [
    {
      "databases": [],
      "name": "dbuser1"
    },
    {
      "databases": [
        {
          "name": "databaseB"
        },
        {
          "name": "databaseC"
        }
      ],
      "name": "dbuser2"
    },
    {
      "databases": [],
      "name": "dbuser3"
    },
    {
      "databases": [
```

```
{
  {
    "name": "sampledb"
  },
  {
    "name": "demouser"
  }
}
```

Example 4.69. List Users in Database Instance Paged Response: XML

The following examples show the *paginated* List Users in Database Instance responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 461
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<users xmlns="http://docs.openstack.org/database/api/v1.0">
  <user name="dbuser1">
    <databases/>
  </user>
  <user name="dbuser2">
    <databases>
      <database name="databaseB"/>
      <database name="databaseC"/>
    </databases>
  </user>
  <links>
    <link href="https://openstack.example.com/v1.0/1234/instances/
098653ba-218b-47ce-936a-e0b749101f81/users?marker=dbuser2&limit=2" rel=
"next"/>
  </links>
</users>
```

Example 4.70. List Users in Database Instance Paged Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 279
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "links": [
    {
      "href": "https://openstack.example.com/v1.0/1234/instances/
44b277eb-39be-4921-be31-3d61b43651d7/users?marker=dbuser2&limit=2",
      "rel": "next"
    }
  ],
  "users": [
    {
      "databases": [],
      "name": "dbuser1"
    },
    {
      "databases": [
```

```
{
  {
    "name": "databaseB"
  },
  {
    "name": "databaseC"
  }
],
"name": "dbuser2"
}
]
```

4.5.3. Change User(s) Password

This operation changes the MySQL password of one or more users.



Note

For information about choosing a valid password, please refer to [Create User \[53\]](#) for details.

Method	URI	Description
PUT	/instances{/instanceId}/users	Changes the MySQL password of one or more users.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.3.1. Request

This table shows the URI parameters for the Change User(s) Password Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.

Example 4.71. Change User(s) Password Request: XML

The following examples show the Change User(s) Password requests:

```
PUT /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<users xmlns="http://docs.openstack.org/database/api/v1.0">
  <user password="5" name="exampleuser"/>
</users>
```

Example 4.72. Change User(s) Password Request: JSON

```
PUT /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json

{
  "users": [
    {
      "name": "dbuser1",
```

```
        "password": "newpassword"
      },
      {
        "name": "dbuser2",
        "password": "anotherpassword"
      }
    ]
  }
```

4.5.3.2. Response

Example 4.73. Change User(s) Password Response: XML

The following examples show the Change User(s) Password responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 501
Date: Wed, 27 Jun 2012 21:56:06 GMT
```

Example 4.74. Change User(s) Password Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 152
Date: Wed, 21 Mar 2012 17:46:46 GMT
```


4.5.4. List User

This operation lists the specified user's name and a list of databases that the user can access.

Method	URI	Description
GET	/instances{/instanceId}/users{/name}	Lists the specified user's name and a list of databases that the user can access.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.4.1. Request

This table shows the URI parameters for the List User Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{name}	String	The name for the specified user.

Example 4.75. List User Request: XML

The following examples show the List User requests:

```
GET /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/testuser
HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.76. List User Request: JSON

```
GET /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/
exampleuser HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.5.4.2. Response

Example 4.77. List User Response: XML

The following examples show the List User responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT

<user name="testuser" xmlns="http://docs.openstack.org/database/api/v1.0">
  <databases>
    <database name="exampledb"/>
  </databases>
</user>
```

Example 4.78. List User Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT

{
  "user": {
    "name": "exampleuser",
    "databases": [
      {
        "name": "databaseA"
      },
      {
        "name": "databaseB"
      }
    ]
  }
}
```

4.5.5. Delete User

This operation deletes the specified user for the specified database instance.



Warning

There is a bug in a python library that development is using that may cause incorrect user deletions to occur if a period (.) is used in the user name. In this case, the user name is truncated to remove the portion of the name from the period to the end, leaving only the portion from the beginning up to the period. For example, for a user named "my.userA", the bug would truncate the user name to "my", and if the user "my" exists, that user will be incorrectly deleted. To avoid the problem, do not use periods in user names.

Method	URI	Description
DELETE	/instances{/instanceId}/users{/name}	Deletes the user identified by {name} for the specified database instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.5.1. Request

This table shows the URI parameters for the Delete User Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{name}	String	The name for the specified user.

Example 4.79. Delete User Request: XML

The following examples show the Delete User requests:

```
DELETE /v1.0/1234/instances/098653ba-218b-47ce-936a-e0b749101f81/users/
testuser HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.80. Delete User Request: JSON

```
DELETE /v1.0/1234/instances/44b277eb-39be-4921-be31-3d61b43651d7/users/
testuser HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
```

```
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.5.5.2. Response

Example 4.81. Delete User Response: XML

The following examples show the Delete User responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

Example 4.82. Delete User Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)
```

4.5.6. List User Access

This operation shows a list of all databases a user has access to.

Method	URI	Description
GET	/instances{/instanceId}/users{/name}/databases	Shows a list of all databases a user has access to.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.6.1. Request

This table shows the URI parameters for the List User Access Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{name}	String	The name for the specified user.

Example 4.83. List User Access Request: XML

The following examples show the List User Access requests:

```
GET /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/testuser/
databases HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.84. List User Access Request: JSON

```
GET /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/
exampleuser/databases HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.5.6.2. Response

Example 4.85. List User Access Response: XML

The following examples show the List User Access responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT

<databases xmlns="http://docs.openstack.org/database/api/v1.0">
  <database name="exampledb"/>
</databases>
```

Example 4.86. List User Access Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT

{
  "databases": [
    {
      "name": "databaseA"
    },
    {
      "name": "databaseB"
    }
  ]
}
```

4.5.7. Grant User Access

This operation grants access for the specified user to one or more databases for the specified instance. The user is granted ALL privileges on the database. Refer to the information at the beginning of [Section 4.5, “Users” \[52\]](#) for more details on access.

Method	URI	Description
PUT	/instances{/instanceId}/users{/name}/databases	Grant access for the specified user to one or more databases for the specified instance.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.7.1. Request

This table shows the URI parameters for the Grant User Access Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{name}	String	The name for the specified user.

Example 4.87. Grant User Access Request: XML

The following examples show the Grant User Access requests:

```
PUT /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/testuser/
databases HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml

<databases xmlns="http://docs.openstack.org/database/api/v1.0">
  <database name="extradb"/>
</databases>
```

Example 4.88. Grant User Access Request: JSON

```
PUT /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/
exampleuser/databases HTTP/1.1
User-Agent: python-example-client
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json

{
  "databases": [
    {
```

```
    "name": "databaseC"  
  },  
  {  
    "name": "databaseD"  
  }  
]  
}
```

4.5.7.2. Response

Example 4.89. Grant User Access Response: XML

The following examples show the Grant User Access responses:

```
HTTP/1.1 202 Accepted  
Content-Type: application/xml  
Content-Length: 0  
Date: Wed, 27 Jun 2012 23:11:19 GMT
```

Example 4.90. Grant User Access Response: JSON

```
HTTP/1.1 202 Accepted  
Content-Type: application/json  
Content-Length: 0  
Date: Wed, 27 Jun 2012 23:11:19 GMT
```


4.5.8. Revoke User Access

This operation removes access to the specified database for the specified user.

Method	URI	Description
DELETE	/instances{/instanceId}/users{/name}/databases{/databaseName}	Remove access to the specified database for the specified user.

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.5.8.1. Request

This table shows the URI parameters for the Revoke User Access Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{instanceId}	String	The instance ID for the specified database instance.
{name}	String	The name for the specified user.
{databaseName}	String	The name for the specified database.

Example 4.91. Revoke User Access Request: XML

The following examples show the Revoke User Access requests:

```
DELETE /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/  
testuser/databases/extradb HTTP/1.1  
User-Agent: python-example-client  
Host: openstack.example.com  
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7  
Accept: application/xml  
Content-Type: application/xml
```

Example 4.92. Revoke User Access Request: JSON

```
DELETE /v1.0/1234/instances/692d8418-7a8f-47f1-8060-59846c6e024f/users/  
exampleuser/databases/databaseC HTTP/1.1  
User-Agent: python-example-client  
Host: openstack.example.com  
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7  
Accept: application/json  
Content-Type: application/json
```

This operation does not require a request body.

4.5.8.2. Response

Example 4.93. Revoke User Access Response: XML

The following examples show the Revoke User Access responses:

```
HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT
```

Example 4.94. Revoke User Access Response: JSON

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 0
Date: Wed, 27 Jun 2012 23:11:19 GMT
```

4.6. Flavors

This section describes the operations that are supported for flavors.

Method	URI	Description
GET	/flavors	Lists information for all available <i>flavors</i> .
GET	/flavors{/flavorId}	Lists all flavor information about the specified flavor ID.

4.6.1. List Flavors

This operation lists information for all available flavors.

This resource is identical to the flavors found in the OpenStack Nova API, but without the disk property.

Method	URI	Description
GET	/flavors	Lists information for all available <i>flavors</i> .

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.6.1.1. Request

This table shows the URI parameters for the List Flavors Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.

Example 4.95. List Flavors Request: XML

The following examples show the List Flavors requests:

```
GET /v1.0/1234/flavors HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.96. List Flavors Request: JSON

```
GET /v1.0/1234/flavors HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
Content-Type: application/json
```

This operation does not require a request body.

4.6.1.2. Response

Example 4.97. List Flavors Response: XML

The following examples show the List Flavors responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1600
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<flavors xmlns="http://docs.openstack.org/database/api/v1.0">
  <flavor id="1" name="512MB Instance" ram="512">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/1"
rel="self"/>
      <link href="https://openstack.example.com/flavors/1" rel=
"bookmark"/>
    </links>
  </flavor>
  <flavor id="2" name="1GB Instance" ram="1024">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/2"
rel="self"/>
      <link href="https://openstack.example.com/flavors/2" rel=
"bookmark"/>
    </links>
  </flavor>
  <flavor id="3" name="2GB Instance" ram="2048">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/3"
rel="self"/>
      <link href="https://openstack.example.com/flavors/3" rel=
"bookmark"/>
    </links>
  </flavor>
  <flavor id="4" name="4GB Instance" ram="4096">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/4"
rel="self"/>
      <link href="https://openstack.example.com/flavors/4" rel=
"bookmark"/>
    </links>
  </flavor>
  <flavor id="5" name="8GB Instance" ram="8192">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/5"
rel="self"/>
      <link href="https://openstack.example.com/flavors/5" rel=
"bookmark"/>
    </links>
  </flavor>
  <flavor id="6" name="16GB Instance" ram="16384">
    <links>
      <link href="https://openstack.example.com/v1.0/1234/flavors/6"
rel="self"/>
      <link href="https://openstack.example.com/flavors/6" rel=
"bookmark"/>
    </links>
  </flavor>
</flavors>
```

Example 4.98. List Flavors Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1186
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "flavors": [
    {
      "id": 1,
      "links": [
        {
          "href": "https://openstack.example.com/v1.0/1234/flavors/1",
          "rel": "self"
        },
        {
          "href": "https://openstack.example.com/flavors/1",
          "rel": "bookmark"
        }
      ],
      "name": "512MB Instance",
      "ram": 512
    },
    {
      "id": 2,
      "links": [
        {
          "href": "https://openstack.example.com/v1.0/1234/flavors/2",
          "rel": "self"
        },
        {
          "href": "https://openstack.example.com/flavors/2",
          "rel": "bookmark"
        }
      ],
      "name": "1GB Instance",
      "ram": 1024
    },
    {
      "id": 3,
      "links": [
        {
          "href": "https://openstack.example.com/v1.0/1234/flavors/3",
          "rel": "self"
        },
        {
          "href": "https://openstack.example.com/flavors/3",
          "rel": "bookmark"
        }
      ],
      "name": "2GB Instance",
      "ram": 2048
    },
    {
      "id": 4,
      "links": [
        {
```

```
4",
    "href": "https://openstack.example.com/v1.0/1234/flavors/4",
    "rel": "self"
  },
  {
    "href": "https://openstack.example.com/flavors/4",
    "rel": "bookmark"
  }
],
"name": "4GB Instance",
"ram": 4096
},
{
  "id": 5,
  "links": [
    {
      "href": "https://openstack.example.com/v1.0/1234/flavors/5",
      "rel": "self"
    },
    {
      "href": "https://openstack.example.com/flavors/5",
      "rel": "bookmark"
    }
  ],
  "name": "8GB Instance",
  "ram": 8192
},
{
  "id": 6,
  "links": [
    {
      "href": "https://openstack.example.com/v1.0/1234/flavors/6",
      "rel": "self"
    },
    {
      "href": "https://openstack.example.com/flavors/6",
      "rel": "bookmark"
    }
  ],
  "name": "16GB Instance",
  "ram": 16384
}
]
```

4.6.2. List Flavor By ID

This operation lists all information for the specified flavor ID with details of the RAM.

This resource is identical to the flavors found in the OpenStack Nova API, but without the disk property.



Note

The flavorId parameter should be an integer. If a floating point value is used for the flavorId parameter, the decimal portion is truncated and the integer portion is used as the value of the flavorId.

Reviewer / Writer: need to confirm that this behavior is not changed in subsequent releases, and if it is prevented, remove the Note above.

Method	URI	Description
GET	/flavors/{flavorId}	Lists all flavor information about the specified flavor ID.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), unprocessableEntity (422), instanceFault (500), notImplemented (501), serviceUnavailable (503), itemNotFound (404)

4.6.2.1. Request

This table shows the URI parameters for the List Flavor By ID Request:

Name	Type	Description
{tenantId}	String	The tenant ID of the owner of the specified instance.
{flavorId}	String	The flavor ID for the specified flavor.

Example 4.99. List Flavor By ID Request: XML

The following examples show the List Flavor By ID requests:

```
GET /v1.0/1234/flavors/1 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/xml
Content-Type: application/xml
```

Example 4.100. List Flavor By ID Request: JSON

```
GET /v1.0/1234/flavors/1 HTTP/1.1
User-Agent: python-reddwarfclient
Host: openstack.example.com
X-Auth-Token: 87c6033c-9ff6-405f-943e-2deb73f278b7
Accept: application/json
```

```
Content-Type: application/json
```

This operation does not require a request body.

4.6.2.2. Response

Example 4.101. List Flavor By ID Response: XML

The following examples show the List Flavor By ID responses:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 283
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

<flavor id="1" name="512MB Instance" ram="512" xmlns="http://docs.openstack.org/database/api/v1.0">
  <links>
    <link href="https://openstack.example.com/v1.0/1234/flavors/1" rel="self"/>
    <link href="https://openstack.example.com/flavors/1" rel="bookmark"/>
  </links>
</flavor>
```

Example 4.102. List Flavor By ID Response: JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 206
Date: Mon, 18 Mar 2013 19:09:17 GMT
Server: Jetty(8.0.y.z-SNAPSHOT)

{
  "flavor": {
    "id": 1,
    "links": [
      {
        "href": "https://openstack.example.com/v1.0/1234/flavors/1",
        "rel": "self"
      },
      {
        "href": "https://openstack.example.com/flavors/1",
        "rel": "bookmark"
      }
    ],
    "name": "512MB Instance",
    "ram": 512
  }
}
```


Glossary

database

A MySQL database within a database instance.

database instance

A database instance is an isolated MySQL instance in a single tenant environment on a shared physical host machine. Also referred to as instance.

flavor

A flavor is an available hardware configuration for a database instance. Each flavor has a unique combination of memory capacity and priority for CPU time.

volume

A volume is user-specified storage that contains the MySQL data directory. Volumes are automatically provisioned on shared Internet Small Computer System Interface (iSCSI) storage area networks (SAN) that provide for increased performance, scalability, availability and manageability. Applications with high I/O demands are performance optimized and data is protected through both local and network RAID-10. Additionally, network RAID provides synchronous replication of volumes with automatic failover and load balancing across available storage clusters.