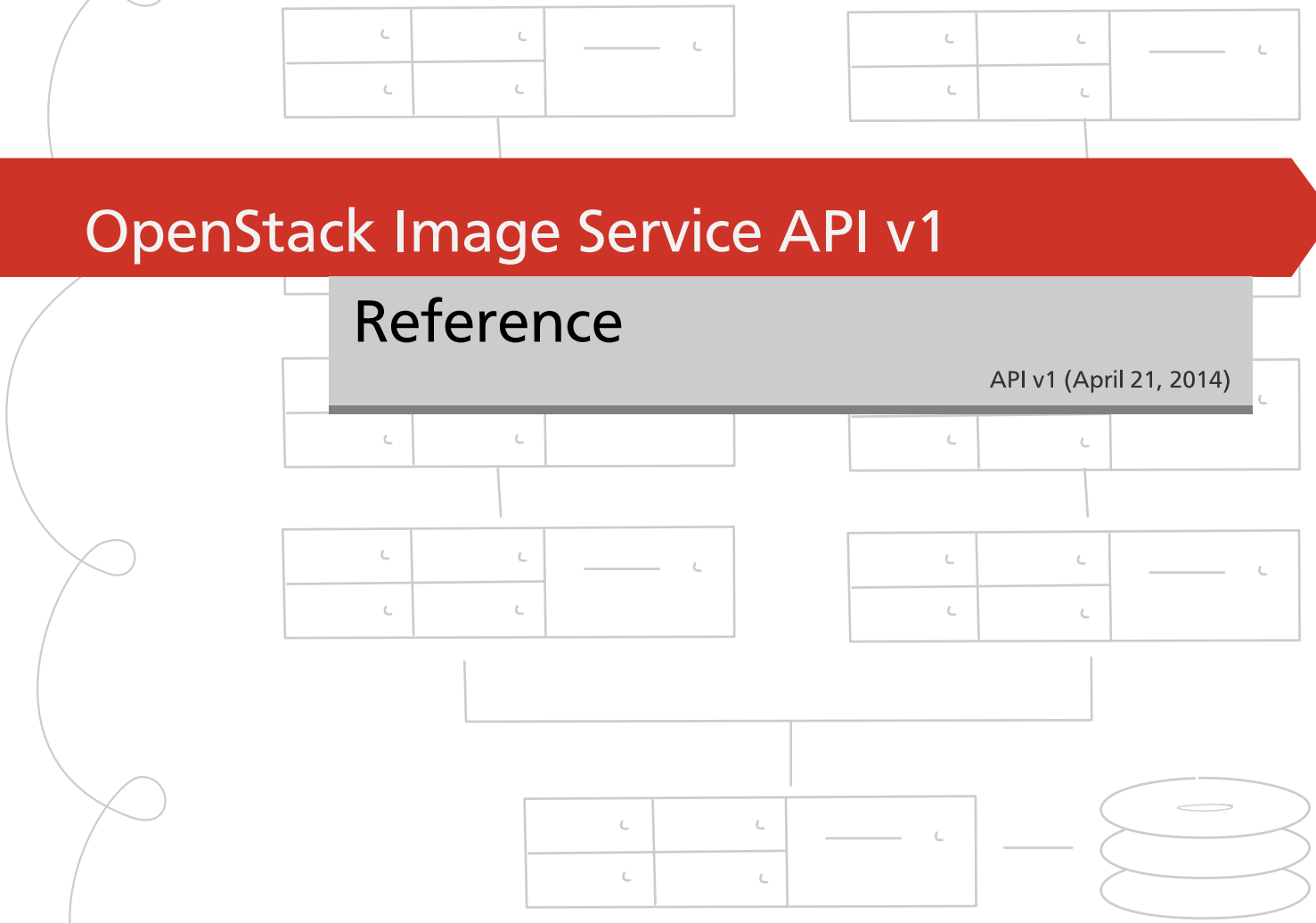
OpenStack Image Service API v1

Reference

API v1 (April 21, 2014)



BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack Image Service API v1 Reference

API v1 (2014-04-21)

This document is for software developers who develop applications by using the OpenStack™ Image Service Application Programming Interface (API) v1.

Table of Contents

Preface

OpenStack Image Service offers retrieval, storage, and metadata assignment for your images that you want to run in your OpenStack cloud. The project is code-named Glance.

OpenStack Image service enables users to store and retrieve images through a simple Web Service (ReST: Representational State Transfer) interface.

For more details on the OpenStack Image service, please refer to docs.openstack.org/developer/glance/

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

Intended Audience

This guide is intended to assist software developers who want to develop applications using the OpenStack Image Service API. It fully documents the ReST application programming interface (API) that allows developers to interact with the storage components of the OpenStack Image system. To use the information provided here, you should first have a general understanding of the OpenStack Image Service and have access to an installation of OpenStack Image Service. You should also be familiar with:

- ReSTful web services
- HTTP/1.1

Document Change History

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

Revision Date	Summary of Changes
May 30, 2013	• Updated the book title for consistency.

1. General API Information

Table of Contents

Authentication

You can optionally integrate Glance with the OpenStack Identity Service project. Setting this up is relatively straightforward: the Identity Service distribution at http://github.com/openstack/keystone includes the requisite middleware and examples of appropriately modified glance-api.conf and glance-registry.conf configuration files in the examples/paste directory. Once you have installed Keystone and edited your configuration files, newly created images will have their owner attribute set to the tenant of the authenticated users, and the is_public attribute will cause access to those images for which it is false to be restricted to only the owner.

The exception is those images for which owner is set to null, which may only be done by those users having the Admin role. These images may still be accessed by the public, but will not appear in the list of public images. This allows the Glance Registry owner to publish images for beta testing without allowing those images to show up in lists, potentially confusing users.

It is possible to allow a private image to be shared with one or more alternate tenants. This is done through image memberships, which are available via the members resource of images. (For more details, see the next chapter.) Essentially, a membership is an association between an image and a tenant which has permission to access that image. These membership associations may also have a can_share attribute, which, if set to true, delegates the authority to share an image to the named tenant.

2. The Glance REST API

Table of Contents

Glance has a RESTful API that exposes both metadata about registered virtual machine images and the image data itself.

A host that runs the `bin/glance-api` service is said to be a *Glance API Server*.

Assume there is a Glance API server running at the URL `http://glance.example.com`.

Let's walk through how a user might request information from this server.

Requesting a List of Public VM Images

We want to see a list of available virtual machine images that the Glance server knows about.

We issue a GET request to `http://glance.example.com/images/` to retrieve this list of available *public* images. The data is returned as a JSON-encoded mapping in the following format:

```
{'images': [
  {'status: 'active',
   'name': 'Ubuntu 10.04 Plain',
   'disk_format': 'vhd',
   'container_format': 'ovf',
   'size': '5368709120'}
  ...]}
```

```
All images returned from the above `GET` request are public images
```

Requesting Detailed Metadata on Public VM Images

We want to see more detailed information on available virtual machine images that the Glance server knows about.

We issue a GET request to `http://glance.example.com/images/detail` to retrieve this list of available *public* images. The data is returned as a JSON-encoded mapping in the following format:

```
{'images': [
  {'name': 'Ubuntu 10.04 Plain 5GB',
   'disk_format': 'vhd',
   'container_format': 'ovf',
   'size': '5368709120',
   'checksum': 'c2e5db72bd7fd153f53ede5da5a06de3',
   'location': 'swift://account:key/container/image.tar.gz.0',
   'created_at': '2010-02-03 09:34:01',
   'updated_at': '2010-02-03 09:34:01',
   'deleted_at': '',
   'status': 'active',
   'is_public': true,
   'owner': null,
   'properties': {'distro': 'Ubuntu 10.04 LTS'}},
  ...]}
```

```
All images returned from the above `GET` request are public images

All timestamps returned are in UTC

The `updated_at` timestamp is the timestamp when an image's metadata
was last updated, not its image data, as all image data is immutable
once stored in Glance

The `properties` field is a mapping of free-form key/value pairs that
have been saved with the image metadata

The `checksum` field is an MD5 checksum of the image file data

The `is_public` field is a boolean indicating whether the image is
publicly available

The `owner` field is a string which may either be null or which will
indicate the owner of the image
```

Filtering Images Returned via `GET /images` and `GET /images/detail`

Both the `GET /images` and `GET /images/detail` requests take query parameters that serve to filter the returned list of images. The following list details these query parameters.

- `name=NAME`

  Filters images having a `name` attribute matching `NAME`.

- `container_format=FORMAT`

  Filters images having a `container_format` attribute matching `FORMAT`

  For more information, see :doc:`About Disk and Container Formats <formats>`

- `disk_format=FORMAT`

  Filters images having a `disk_format` attribute matching `FORMAT`

  For more information, see :doc:`About Disk and Container Formats <formats>`

- `status=STATUS`

  Filters images having a `status` attribute matching `STATUS`

  For more information, see :doc:`About Image Statuses <statuses>`

- `size_min=BYTES`

  Filters images having a `size` attribute greater than or equal to `BYTES`

- `size_max=BYTES`

  Filters images having a `size` attribute less than or equal to `BYTES`

These two resources also accept sort parameters:

- `sort_key=KEY`

  Results will be ordered by the specified image attribute `KEY`. Accepted values include `id`, `name`, `status`, `disk_format`, `container_format`, `size`, `created_at` (default) and `updated_at`.

- `sort_dir=DIR`

  Results will be sorted in the direction `DIR`. Accepted values are `asc` for ascending or `desc` (default) for descending.

Requesting Detailed Metadata on a Specific Image

We want to see detailed information for a specific virtual machine image that the Glance server knows about.

We have queried the Glance server for a list of public images and the data returned includes the `uri` field for each available image. This `uri` field value contains the exact location needed to get the metadata for a specific image.

Continuing the example from above, in order to get metadata about the first public image returned, we can issue a HEAD request to the Glance server for the image's URI.

We issue a HEAD request to `http://glance.example.com/images/1` to retrieve complete metadata for that image. The metadata is returned as a set of HTTP headers that begin with the prefix `x-image-meta-`. The following shows an example of the HTTP headers returned from the above HEAD request:

```
x-image-meta-name            Ubuntu 10.04 Plain 5GB
x-image-meta-disk-format     vhd
```

```
x-image-meta-container-format ovf
x-image-meta-size             5368709120
x-image-meta-checksum         c2e5db72bd7fd153f53ede5da5a06de3
x-image-meta-location         swift://account:key/container/image.tar.gz.0
x-image-meta-created_at       2010-02-03 09:34:01
x-image-meta-updated_at       2010-02-03 09:34:01
x-image-meta-deleted_at
x-image-meta-status           available
x-image-meta-is-public        true
x-image-meta-owner            null
x-image-meta-property-distro  Ubuntu 10.04 LTS
```

All timestamps returned are in UTC The `x-image-meta-updated_at` timestamp is the timestamp when an image's metadata was last updated, not its image data, as all image data is immutable once stored in Glance There may be multiple headers that begin with the prefix `x-image-meta-property-`. These headers are free-form key/value pairs that have been saved with the image metadata. The key is the string after `x-image-meta-property-` and the value is the value of the header The response's `ETag` header will always be equal to the `x-image-meta-checksum` value The response's `x-image-meta-is-public` value is a boolean indicating whether the image is publicly available The response's `x-image-meta-owner` value is a string which may either be null or which will indicate the owner of the image

Retrieving a Virtual Machine Image

We want to retrieve that actual raw data for a specific virtual machine image that the Glance server knows about.

We have queried the Glance server for a list of public images and the data returned includes the `uri` field for each available image. This `uri` field value contains the exact location needed to get the metadata for a specific image.

Continuing the example from above, in order to get metadata about the first public image returned, we can issue a HEAD request to the Glance server for the image's URI.

We issue a GET request to `http://glance.example.com/images/1` to retrieve metadata for that image as well as the image itself encoded into the response body.

The metadata is returned as a set of HTTP headers that begin with the prefix `x-image-meta-`. The following shows an example of the HTTP headers returned from the above GET request:

```
x-image-meta-name             Ubuntu 10.04 Plain 5GB
x-image-meta-disk-format      vhd
x-image-meta-container-format ovf
x-image-meta-size             5368709120
x-image-meta-checksum         c2e5db72bd7fd153f53ede5da5a06de3
x-image-meta-location         swift://account:key/container/image.tar.gz.0
x-image-meta-created_at       2010-02-03 09:34:01
x-image-meta-updated_at       2010-02-03 09:34:01
x-image-meta-deleted_at
x-image-meta-status           available
x-image-meta-is-public        true
x-image-meta-owner            null
x-image-meta-property-distro  Ubuntu 10.04 LTS
```

```
All timestamps returned are in UTC

The `x-image-meta-updated_at` timestamp is the timestamp when an
image's metadata was last updated, not its image data, as all
image data is immutable once stored in Glance

There may be multiple headers that begin with the prefix
`x-image-meta-property-`.  These headers are free-form key/value pairs
that have been saved with the image metadata. The key is the string
after `x-image-meta-property-` and the value is the value of the header

The response's `Content-Length` header shall be equal to the value of
the `x-image-meta-size` header

The response's `ETag` header will always be equal to the
`x-image-meta-checksum` value

The response's `x-image-meta-is-public` value is a boolean indicating
whether the image is publicly available

The response's `x-image-meta-owner` value is a string which may either
be null or which will indicate the owner of the image

The image data itself will be the body of the HTTP response returned
from the request, which will have content-type of
`application/octet-stream`.
```

Adding a New Virtual Machine Image

We have created a new virtual machine image in some way (created a "golden image" or snapshotted/backed up an existing image) and we wish to do two things:

- Store the disk image data in Glance
- Store metadata about this image in Glance

We can do the above two activities in a single call to the Glance API. Assuming, like in the examples above, that a Glance API server is running at `glance.example.com`, we issue a POST request to add an image to Glance:

```
POST http://glance.example.com/images/
```

The metadata about the image is sent to Glance in HTTP headers. The body of the HTTP request to the Glance API will be the MIME-encoded disk image data.

Adding Image Metadata in HTTP Headers

Glance will view as image metadata any HTTP header that it receives in a

```
``POST`` request where the header key is prefixed with the strings
``x-image-meta-`` and ``x-image-meta-property-``.
```

The list of metadata headers that Glance accepts are listed below.

- `x-image-meta-name`

This header is optional . Its value should be the name of the image.

Note that the name of an image *is not unique to a Glance node*. It would be an unrealistic expectation of users to know all the unique names of all other user's images.

- `x-image-meta-id`

  This header is optional.

  When present, Glance will use the supplied identifier for the image. If the identifier already exists in that Glance node, then a **409 Conflict** will be returned by Glance.

  When this header is *not* present, Glance will generate an identifier for the image and return this identifier in the response (see below)

- `x-image-meta-store`

  This header is optional. Valid values are one of `file`, `s3`, or `swift`

  When present, Glance will attempt to store the disk image data in the backing store indicated by the value of the header. If the Glance node does not support the backing store, Glance will return a **400 Bad Request**.

  When not present, Glance will store the disk image data in the backing store that is marked default. See the configuration option `default_store` for more information.

- `x-image-meta-disk-format`

  This header is optional. Valid values are one of `aki`, `ari`, `ami`, `raw`, `iso`, `vhd`, `vdi`, `qcow2`, or `vmdk`.

  For more information, see :doc:`About Disk and Container Formats <formats>`

- `x-image-meta-container-format`

  This header is optional. Valid values are one of `aki`, `ari`, `ami`, `bare`, or `ovf`.

  For more information, see :doc:`About Disk and Container Formats <formats>`

- `x-image-meta-size`

  This header is optional.

  When present, Glance assumes that the expected size of the request body will be the value of this header. If the length in bytes of the request body *does not match* the value of this header, Glance will return a **400 Bad Request**.

  When not present, Glance will calculate the image's size based on the size of the request body.

- `x-image-meta-checksum`

  This header is optional. When present it shall be the expected **MD5** checksum of the image file data.

When present, Glance will verify the checksum generated from the backend store when storing your image against this value and return a **400 Bad Request** if the values do not match.

- `x-image-meta-is-public`

  This header is optional.

  When Glance finds the string "true" (case-insensitive), the image is marked as a public image, meaning that any user may view its metadata and may read the disk image from Glance.

  When not present, the image is assumed to be *not public* and specific to a user.

- `x-image-meta-owner`

  This header is optional and only meaningful for admins.

  Glance normally sets the owner of an image to be the tenant or user (depending on the "owner_is_tenant" configuration option) of the authenticated user issuing the request. However, if the authenticated user has the Admin role, this default may be overridden by setting this header to null or to a string identifying the owner of the image.

- `x-image-meta-property-*`

  When Glance receives any HTTP header whose key begins with the string prefix `x-image-meta-property-`, Glance adds the key and value to a set of custom, free-form image properties stored with the image. The key is the lower-cased string following the prefix `x-image-meta-property-` with dashes and punctuation replaced with underscores.

  For example, if the following HTTP header were sent:

```
x-image-meta-property-distro  Ubuntu 10.10
```

  Then a key/value pair of "distro"/"Ubuntu 10.10" will be stored with the image in Glance.

  There is no limit on the number of free-form key/value attributes that can be attached to the image. However, keep in mind that the 8K limit on the size of all HTTP headers sent in a request will effectively limit the number of image properties.

Updating an Image

Glance will view as image metadata any HTTP header that it receives in a

```
``PUT`` request where the header key is prefixed with the strings
``x-image-meta-`` and ``x-image-meta-property-``.
```

If an image was previously reserved, and thus is in the `queued` state, then image data can be added by including it as the request body. If the image already as data associated with it (e.g. not in the `queued` state), then including a request body will result in a **409 Conflict** exception.

On success, the PUT request will return the image metadata encoded as HTTP headers.

See more about image statuses here: :doc:`Image Statuses <statuses>`

Requesting Image Memberships

We want to see a list of the other system tenants (or users, if "owner_is_tenant" is False) that may access a given virtual machine image that the Glance server knows about. We take the `uri` field of the image data, append `/members` to it, and issue a GET request on the resulting URL.

Continuing from the example above, in order to get the memberships for the first public image returned, we can issue a GET request to the Glance server for `http://glance.example.com/images/1/members`. What we will get back is JSON data such as the following:

```
{'members': [
 {'member_id': 'tenant1',
  'can_share': false}
 ...]}
```

The `member_id` field identifies a tenant with which the image is shared. If that tenant is authorized to further share the image, the `can_share` field is `true`.

Requesting Shared Images

We want to see a list of images which are shared with a given tenant. We issue a GET request to `http://glance.example.com/shared-images/tenant1`. We will get back JSON data such as the following:

```
{'shared_images': [
 {'image_id': 1,
  'can_share': false}
 ...]}
```

The `image_id` field identifies an image shared with the tenant named by *member_id*. If the tenant is authorized to further share the image, the `can_share` field is `true`.

Adding a Member to an Image

We want to authorize a tenant to access a private image. We issue a PUT request to `http://glance.example.com/images/1/members/tenant1`. With no body, this will add the membership to the image, leaving existing memberships unmodified and defaulting new memberships to have `can_share` set to `false`. We may also optionally attach a body of the following form:

```
{'member':
 {'can_share': true}
}
```

If such a body is provided, both existing and new memberships will have `can_share` set to the provided value (either `true` or `false`). This query will return a 204 ("No Content") status code.

Removing a Member from an Image

We want to revoke a tenant's right to access a private image. We issue a DELETE request to `http://glance.example.com/images/1/members/tenant1`. This query will return a 204 ("No Content") status code.

Replacing a Membership List for an Image

The full membership list for a given image may be replaced. We issue a PUT request to `http://glance.example.com/images/1/members` with a body of the following form:

```
{'memberships': [
 {'member_id': 'tenant1',
  'can_share': false}
 ...]}
```

All existing memberships which are not named in the replacement body are removed, and those which are named have their `can_share` settings changed as specified. (The `can_share` setting may be omitted, which will cause that setting to remain unchanged in the existing memberships.) All new memberships will be created, with `can_share` defaulting to `false` if it is not specified.