Gluster Management Gateway 1.0.0alpha

REST API Guide

Red Hat Engineering Content Services

Gluster Management Gateway 1.0.0alpha REST API Guide

Author Red Hat Engineering Content Services `docfeedback@gluster.com`

1801 Varsity DriveRaleigh, NC27606-2072USAPhone: +1 919 754 3700Phone: 888 733 4281Fax: +1 919 754 3701

This document describes the Representational State Transfer (REST) API for Gluster Management Gateway.

Preface

Gluster Management Gateway provides simple and powerful **Representational State Transfer** (REST) APIs for managing your Gluster storage cluster. It also hosts the Gluster Management Console binaries, which can be run from a browser using the Java Webstart. Gluster Management Console uses the REST APIs for performing the administrative tasks on your Gluster storage cluster.

GlusterFS is an open source, distributed file system capable of scaling to several petabytes and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design and can deliver exceptional performance for diverse workloads.

This guide explains how to use Gluster Management Gateway's REST API. This document covers the fundamentals of the REST architectural concepts in the context of a cloud storage environment and provides examples of the APIs in operation.

1. Audience

This guide is intended for developers and system administrators who aim to integrate their storage environment with third-party applications and scripts.

This document assumes that you are familiar with the Linux operating system, concepts of File System, GlusterFS concepts, REST web services, and HTTP/1.1.

2. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

2.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

[1] https://fedorahosted.org/liberation-fonts/

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

2.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

2.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

3. Feedback

Red Hat welcomes your comments and suggestions on the quality and usefulness of its documentation. If you find any errors or have any other suggestions, write to us at `docfeedback@gluster.com` for clarification and provide the chapter, section, and page number, if available.

Red Hat offers a range of resources related to GlusterFS software:

- Discuss technical problems and solutions on the Discussion Forum (*http://community.gluster.org*)
- Get hands-on step-by-step tutorials (*http://www.gluster.org/community/documentation/index.php/Main_Page*)
- Reach Support (*http://www.redhat.com/support/*)

Introduction

Gluster Management Gateway provides **Representational State Transfer (REST)** API which provides software developers and system administrators with control over their cloud storage environment outside of the standard web interface. The REST API is useful for developers and administrators who aim to integrate the functionality of GlusterFS environment with custom scripts or external applications that access the API through the standard Hypertext Transfer Protocol (HTTP).

This documentation acts as a reference to the Gluster Management Gateway REST API. It aims to provide developers and administrators with instructions and examples to help harness the functionality of their virtualized cloud environments environment through the REST API

The API supports Extensible Markup Language (XML) and JSON (JavaScript Object Notation). The following sections provide an overview of REST architecture and explains key terms of the REST API.

Note

The Gluster Management Gateway works on SSL certificate. The product is shipped with a self signed SSL certificate. Users must procure their own certificate from a Certificate Authority (CA) and install it.

1.1. Representational State Transfer

Representational State Transfer (REST) is a design architecture that focuses on resources for a specific service and their representations. A resource representation is a key abstraction of information that corresponds to one specific managed element on a server. A client sends a request to a server element located at a Uniform Resource Identifier (URI) and performs operations with standard HTTP methods, such as **GET**, **POST**, **PUT**, and **DELETE**. This provides a stateless communication between the client and server where each request acts independent of any other request and contains all necessary information to complete the request.

1.2. Resources

A REST API focuses primarily on resources of a specific service. A resource is a key abstraction of information and in the context of virtualization represents an individual component, capability or data set in the virtualized cloud infrastructure.

The API structures resource representations in the following XML document structure:

```
<resource>
    <name>Resource-Name</name>
     ...
</resource>
```

All resource representations contain a set of common attributes

Table 1.1. Common elements to resource representations

Element	Type	Description
name	string	A user-supplied human readable name for the resource. The **name** is unique across all resources of its type.

1.3. Uniform Resource Identifiers (URI)

A user performs a request on any part of the API model to control the virtualization environment. For example, access to all resources in a collection requires only a request to the collection itself. Access to a specific resource requires a request to the resource within the context of its collection. An API user targets these requests to a specific location called a Uniform Resource Identifier (URI).

The REST API combines our virtualization model with the use of HTTP to construct a series of Uniform Resource Identifiers (URIs) for collections and resources. This means an API user accesses collections and resources in a manner similar to loading a web page on the Internet.

For example, an API user accesses a collection group called clusters from a REST API installed at **`www.example.com`** with the following URI:

`http://www.example.com/glustermg/1.0.0alpha/clusters`

Access to a resource called **`mycluster`** within clusters requires the following URI:

`http://www.example.com/glustermg/1.0.0alpha/clusters/mycluster`

1.4. Representations

Access to a resource URI in a RESTful web service does not mean direct access to the resource itself but to an abstraction of the resource. This abstraction is called a representation. The API constructs representations using XML and JSON formats but this documentation focuses primarily on XML representations.

When working with a specific resource, the API constructs a representation using elements. When an API user accesses the following URL: **`http://www.example.com/glustermg/1.0.0alpha/clusters/mycluster/volumes/myvolume`**, an XML resource representation structures elements in the following way:

```
<volume>
    <name>myvolume</name>
    <volumeType>DISTRIBUTED_REPLICATE</volumeType>
    <transportType>ETHERNET</transportType>
    <status>ONLINE</status>
   ...
</volume>
```

A virtual machine resource contains elements such as machine type, operational status, and memory in bytes, CPUs and creation time. These elements form a basic XML structure for a virtual machine resource representation:

```
<server>
       <name>glusterserver1</name>
       <uuid>uuid</uuid>
       <status>ONLINE</Status>
       <NumOfCPUs>4</numOfCPUs>
       <cpuUsage>72.63</cpuUsage>
       <totalMemory>8192.00</totalMemory>
       <memoryInUse>1384.92</memoryInUse>
    <networkInterfaces>
              <networkInterface>
              <name>eth0</name>
              <hwAddr>0a:00:27:00:00:00</ hwAddr>
              <speed>speed</speed>
              <model>ETHERNET</model>
```

```
               <ipAddress>192.168.1.123</ipAdress>
               <netMask>255.255.255.0</netmask>
               <defaultGateway>192.168.1.1</defaultGateway>
               </networkInterface>
         </networkInterfaces>
               <disks>
                      <disk>
                      <name>sdb</name>
                 <description>ATA ST9320423AS</description>
           <uuid>06da4fde-01fe-4c7a-9918-80dc86695fea</uuid>
                     <status>INITIALIZED</status>
                     <type>DATA</type>
                     <interface>pci</interface>
                     <fsType>ext3</fsType>
                     <mountPoint>/export/sdb</mountPoint>
                     <space>2097152.00</space>
                     <spaceInUse>281617.72</spaceInUse>
                     </disk>
                </disks>
         </server>
```

This example shows how the resource representation portrays complex elements, such as network interfaces and disk information. Complex elements contain sub-elements to depict multiple properties of a single element. This demonstrates how REST representations use XML to depict very specific aspects of resources and their elements.

1.5. Requests

The REST API uses HTTP as a protocol to transfer a request for a resource representation from a client to the Gluster Management Gateway server.

```
GET /glustermg/1.0.0alpha/clusters/mycluster/volumes/myvolume HTTP/1.1
Host: www.example.com
Accept: application/xml
```

This example follows the standard HTTP request message format including:

- A **Request-Line**, which requires
  - a **method** e.g. **`GET`**
  - a **URI** e.g **`/glustermg/1.0.0alpha/clusters/mycluster/volumes/myvolume`**; and
  - a **HTTP** version e.g **`HTTP/1.1`**
- A **Header** with fields to define parameters that process a request e.g. **`Host:`** and **`Accept:`**; and
- An optional **Message Entity** depending on the request **method**

Requests from a client to a server contain all necessary information to process the request without depending on any previous request or stored context on the server, which makes the REST API a stateless communication.

1.6. Parameters

QUERY parameters are the most common type of parameter that is appended to the path of the URL when submitting a request such as **`GET`** and **`DELETE`**. **`POST`** and **`PUT`** are form parameters.

1.7. Methods

The benefit of using HTTP to communicate with the REST API is the ability to use a method to access Gluster cloud resources. A method defines the type of request to a resource. The REST API uses four HTTP methods:

Method	Description
GET	Retrieves a resource or collection representation
POST	Creates a resource
PUT	Updates a resource
DELETE	Removes a resource

The default method is **`GET`** but an API user has a choice of any of the four methods to access and control resources in their clustered cloud storage environment

1.8. Headers

The HTTP request contains a header to define its parameters. The header uses specific header fields to define these parameters. The table below lists the two header fields which is required in the context of the REST API:

Header	Description
Host	The target host of the URI i.e. the location of the cloud environment and REST API
Accept	The accepted format for the representation. This documentation uses the **`application/xml`** to define the representation structure as XML format.
Authorization	The username and password of the Gluster Management Gateway user.

1.9. Response Formats

Gluster Management Gateway API supports XML and JSON output types. To instruct the API calls to return the desired format, the client can set appropriate value (**`application/xml`** or **`application/json`**) in the HTTP header “Accept”. When no format is specified, XML will be used as the default output type.

HTTP Status Codes

This chapter provides descriptions of the HTTP success and error status codes used by Gluster Management Gateway.

Note

The HTTP codes listed below are compatible with CDMI specifications.

Table 2.1, "HTTP Success Status Codes" lists the HTTP success status codes.

Table 2.1. HTTP Success Status Codes

Code	HTTP Name	Description
200	OK	Resource Retrieved Successfully (All **GET** request)
201	Created	Resource Created Successfully (All **POST** request)
202	Accepted	Long Running Operation Accepted for Processing (Few **PUT** request)
204	No Content	Operation Successful, No Data (Most **PUT**, all **DELETE** requests)

Table 2.2, "HTTP Error Status Codes" lists the HTTP error status codes.

Table 2.2. HTTP Error Status Codes

Code	HTTP Name	Description
400	Bad Request	Missing or invalid request contents
401	Unauthorized	Invalid authentication/ authorization credentials
403	Forbidden	This user is not allowed to perform this request
404	Not Found	Requested resource not found
405	Method Not Allowed	Requested HTTP verb not allowed on this resource
406	Not Acceptable	No content type can be produced at this **URI** that matches the request
500	Internal Server Error	An unexpected error during processing request

Authentication

Authentication is the process of proving your identity to the system. An API user submits a mandatory Gluster Management Console username and password with all requests to the API and uses HTTP Basic Authentication to encode these credentials.

If a request does not include an appropriate **`Authorization`** header, the API sends a **`401 Authorization Required`** as a result:

Example 3.1. Access to the REST API without appropriate credentials

```
HEAD [base] HTTP/1.1
Host: [host]
HTTP/1.1 401 Authorization Required
```

Request are issued with an **`Authorization`** header. An API user encodes the supplied credentials with the **`username@domain:password`** convention.

Important

Basic authentication involves potentially sensitive information, such as passwords, sent as plain text. REST API requires Hypertext Transfer Protocol Secure (HTTPS) for transport-level encryption of plain-text requests.

API Operations for User

This chapter describes the user operation of the Gluster Management Gateway API.

4.1. Changing User Password

The API changes the default user password of the Gluster Management Gateway with a **`PUT`** request to the URI.

```
PUT /glustermg/1.0.0alpha/users/userName HTTP/1.1
```

cURL command:

`curl --request PUT --user` *`[USER:PASS]`* `--data "oldPassword=`*`oldPassword`*`&newPassword=`*`newPassword`*`" https://`*`[HOST:PORT]`*`/glustermg/1.0.0alpha/users/`*`userName`*

```
HTTP/1.1 204 No Content
Location: https://host:port/glustermg/1.0.0alpha/users/userName
```

API Operations for Clusters

This chapter describes the cluster operations of the Gluster Management Gateway API. These APIs enable you to create, register or unregister a cluster and retrieve cluster list.

5.1. Listing of Clusters

A listing of clusters registered in your cloud environment is obtained by issuing a **GET** request on the resource **URI** obtained from the entry point.

```
GET /glustermg/1.0.0alpha/clusters HTTP/1.1
Accept: application/xml
```

cURL command:

curl --request GET --header "Accept: application/xml" --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<clusters>
    <cluster>clusterName1</cluster>
    <cluster>clusterName2</cluster>
</cluster>
```

5.2. Creating a Cluster

The API creates a new cluster with a **POST** request to the URI containing a representation of the new resource.

A **POST** request requires a **Content-Type: application/x-www-form-urlencoded** header.

```
POST /glustermg/1.0.0alpha/clusters HTTP/1.1
Content-Type: application/x-www-form-urlencoded

clusterName=myCluster
```

cURL command:

curl --request POST --user *[USER:PASS]* --data "clusterName=*myCluster*" https://*[host:port]*/glustermg/1.0.0alpha/clusters

```
HTTP/1.1 201 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster
```

5.3. Registering Clusters

An existing cluster is registered to your cloud environment by issuing a **PUT** request to the resource **URI**.

```
PUT /glustermg/1.0.0alpha/clusters HTTP/1.1
```

```
Content-Type: application/x-www-form-urlencoded

clusterName=myCluster&serverName=myServer
```

cURL command:

`curl --request PUT --user [USER:PASS] --data "clusterName=myCluster&serverName=myServer" https://[host:port]/glustermg/1.0.0alpha/clusters`

```
HTTP/1.1 204 No Content
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster
```

5.4. Unregistering a Cluster

Unregistering of a cluster requires a **`DELETE`** request to its resource **`URI`**.

```
DELETE /glustermg/1.0.0alpha/clusters/myCluster HTTP/1.1
```

cURL command:

`curl --request DELETE --user [USER:PASS] https://[host:port]/glustermg/1.0.0alpha/clusters/myCluster`

```
HTTP/1.1 204 No Content
```

API Operations for Servers

This chapter describes the discovered server and server operations of the Gluster Management Gateway API.

6.1. Listing All Discovered Servers

A listing of all discovered Gluster servers available in your network is obtained by issuing a **`GET`** request on the resource **`URI`** and the **`Accept`** header includes the **`details`** parameter. These servers run GlusterFS, but are not part of any storage cloud environment.

```
GET /glustermg/1.0.0alpha/discoveredservers?details=true/false HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user` ***`[USER:PASS]`*** **`https://`*****`[HOST:PORT]`*****`/glustermg/1.0.0alpha/discoveredservers?details=`*****`true/false`***

The API returns the following representation, if the **`details`** parameter value is **`true`**:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<servers>
    <server>
           <name>serverName</name>
           <numOfCPUs>numOfCPUs</numOfCPUs>
           <cpuUsage>currentCPUUsage</cpuUsage>
           <totalMemory>totalMemory</totalMemory>
           <memoryInUse>memoryInUse</memoryInUse>
      <networkInterfaces>
             <networkInterface>
                    <name>name</name>
                    <hwAddr>macAddress</hwAddr>
                    <speed>speed</speed>
                    <model>model</model>
                    <ipAddress>ipAddress</ipAddress>
                    <netMask>netMask</netMask>
                    <defaultGateway/>defaultGateway</defaultGateway>
             </networkInterface>
            ...
      </networkInterfaces>
      <disks>
          <disk>...</disk>
           ...
       </disks>
    </server>
    ...
</servers>
```

If the **`details`** parameter value is **`false`**, the API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<servers>
     <server>serverName1</server>
     <server>serverName2</server>
```

```
    ...
</servers>
```

6.2. Retrieving Discovered Server Details

The API retrieves the details of the discovered server in your cluster with a **`GET`** request on the resource **`URI`**.

```
GET /glustermg/1.0.0alpha/discoveredservers/serverName HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user [USER:PASS] https://[HOST:PORT]/glustermg/1.0.0alpha/discoveredservers/serverName`

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<server>
      <name>serverName</name>
      <numOfCPUs>numOfCPUs</numOfCPUs>
      <cpuUsage>currentCPUUsage</cpuUsage>
      <totalMemory>totalMemory</totalMemory>
      <memoryInUse>memoryInUse</memoryInUse>
      <networkInterfaces>
            <networkInterface>
                 <name>name</name>
                 <hwAddr>macAddress</hwAddr>
                 <speed>speed</speed>
                 <model>model</model>
                 <ipAddress>ipAddress</ipAddress>
                 <netMask>netMask</netMask>
                 <defaultGateway/>defaultGateway</defaultGateway>
            </networkInterface>
            ...
      </networkInterfaces>
      <disks>
          <disk>...</disk>
          ...
      </disks>
</server>
```

6.3. Listing of Servers

A listing of servers in your cluster is obtained by issuing a **`GET`** request on the resource **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/servers?details=true/
false&nextTo=srvr40&maxCount=20 HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user [USER:PASS] https://[HOST:PORT]/glustermg/1.0.0alpha/clusters/myCluster/servers?details=true/false&nextTo=srvr40&maxCount=20`

Table 6.1. Parameters

Parameter	Description
details	value = `true/false`; optional; false by default
nextTo	optional; first server in the response will be the one '*next to*' the given server name
maxCount	optional; maximum number of servers to be returned

If the **`details`** parameter value is **`true`**, the API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<servers>
    <server>
         <name>serverName</name>
         <uuid>uuid</uuid>
         <status>status</status>
         <numOfCPUs>numOfCPUs</numOfCPUs>
         <cpuUsage>currentCPUUsage</cpuUsage>
         <totalMemory>totalMemory</totalMemory>
         <memoryInUse>memoryInUse</memoryInUse>
         <networkInterfaces>
                   <networkInterface>
                        <name>name</name>
                        <hwAddr>macAddress</hwAddr>
                        <speed>speed</speed>
                        <model>model</model>
                        <ipAddress>ipAddress</ipAddress>
                        <netMask>netMask</netMask>
                        <defaultGateway/>defaultGateway</defaultGateway>
                  </networkInterface>
                  ...
         </networkInterfaces>
         <disks>
            <disk>...</disk>
         ...
       </disks>
    </server>
...
</servers>
```

If the **`details`** parameter value is **`false`**, the API returns the following representation:

```
<servers>
     <server>serverName1</server>
     <server>serverName2</server>
     ...
</servers>
```

6.4. Retrieving Details of a Server

The API retrieves the details of a particular server in a cluster with a **`GET`** request on the resource **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/servers/serverName HTTP/1.1
Accept: application/xml
```

cURL command:

curl --request GET --header "Accept: application/xml" --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/servers/*serverName*

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<server>
     <name>serverName</name>
     <uuid>uuid</uuid>
     <status>status</status>
     <numOfCPUs>numOfCPUs</numOfCPUs>
     <cpuUsage>currentCPUUsage</cpuUsage>
     <totalMemory>totalMemory</totalMemory>
     <memoryInUse>memoryInUse</memoryInUse>
     <networkInterfaces>
                <networkInterface>
                         <name>name</name>
                         <hwAddr>macAddress</hwAddr>
                         <speed>speed</speed>
                         <model>model</model>
                         <ipAddress>ipAddress</ipAddress>
                         <netMask>netMask</netMask>
                         <defaultGateway/>defaultGateway</defaultGateway>
                </networkInterface>
                ...
     </networkInterfaces>
     <disks>
         <disk>...</disk>
         ...
     </disks>
</server>
```

6.5. Adding Servers

Adding servers to the cluster is obtained by issuing a **POST** request on the resource **URI** obtained from the entry point.

```
POST /glustermg/1.0.0alpha/clusters/myCluster/servers HTTP/1.1
Content-Type: application/x-www-form-urlencoded

serverName=hostName
```

cURL command:

curl --request POST --user *[USER:PASS]* --data "*serverName=hostName*" https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/servers

```
HTTP/1.1 201 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/servers/serverName
```

6.6. Deleting a Server

The API deletes a server with a **DELETE** request sent to its resource **URI**.

```
DELETE /glustermg/1.0.0alpha/clusters/myCluster/servers/serverName HTTP/1.1
```

cURL command:

`curl --request DELETE --user *[USER:PASS]* https://glustermg/1.0.0alpha/clusters/*myCluster*/servers/*serverName*`

```
HTTP/1.1 204 No Content
```

6.7. Initializing a Server Disk

The API initializes a server disk by issuing a **`PUT`** request to its **`URI`**.

```
PUT /glustermg/1.0.0alpha/clusters/myCluster/servers/serverName/disks/diskName  HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

cURL command:

`curl --request PUT --user *[USER:PASS]* --data "fsType=*[fsType]*" https://*[HOST:PORT]*/glustermg/1.0.0alpha/clusters/*myCluster*/servers/*serverName*/disks/*diskName*`

The valid values for *`fsType`* (file system types) are **`ext3`**, **`ext4`**, and **`xfs`**.

```
HTTP/1.1 202 No Content
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/tasks/taskID
```

API Operations for Volumes

This chapter describes the volume operations of the Gluster Management Gateway API.

7.1. Listing of Volumes

A listing of volumes in your cluster is obtained by issuing a **`GET`** request on the resource **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/volumes?nextTo=volume40&maxCount=20 HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user` ***`[USER:PASS]`*** **`https://`*****`[host:port]`*****`/glustermg/1.0.0alpha/clusters/myCluster/volumes?nextTo=`*****`volume40`*****`&maxCount=`*****`20`***

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<volumes>
     <volume>
          <name>volumeName</name>
          <volumeType>volumeType</volumeType>
          <transportType>transportType</transportType>
          <status>volumeStatus</status>
          <replicaCount>replicaCount</replicaCount>
          <stripeCount>stripeCount</stripeCount>
          <bricks>
               <brick>
                  <serverName>serverName</serverName>
                  <brickDirectory>brickDir</brickDirectory>
                  <diskName>diskName</diskName>
                  <status>brickStatus</status>
               </brick>
               ...
          </bricks>
             <accessProtocols>
                 <accessProtocol>GLUSTERFS</accessProtocol>
                 <accessProtocol>CIFS</accessProtocol>
                  ...
             </accessProtocols>
             <cifsUsers>user1</cifsUsers>
             <cifsUsers>user2</cifsUsers>
          <options>
             <option>
               <key>optionKey</key>
               <value>optionValue</value>
             </option>
             ...
          </options>
     </volume>
    ...
</volumes>
```

7.2. Retrieving a Volume Detail

The API retrieves the details of a particular volume in a cluster with a **`GET`** request on the resource **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user [USER:PASS] https://[host:port]/glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName`

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<volume>
      <name>volumeName</name>
      <volumeType>volumeType</volumeType>
      <transportType>transportType</transportType>
      <status>volumeStatus</status>
      <replicaCount>replicaCount</replicaCount>
      <stripeCount>stripeCount</stripeCount>
      <bricks>
          <brick>
             <serverName>serverName</serverName>
             <brickDirectory>brickDir</brickDirectory>
             <diskName>diskName</diskName>
             <status>brickStatus</status>
          </brick>
          ...
       </bricks>
            <accessProtocols>
                <accessProtocol>GLUSTERFS</accessProtocol>
                 ...
            <accessProtocols>
                     <options>
                         <option>
                           <key>optionKey</key>
                           <value>optionValue</value>
                          </option>
                          ...
                     </options>
</volume>
```

7.3. Creating a Volume

The API creates a new volume with a **`POST`** request to the **`URI`**. A **`POST`** request requires a **`Content-Type: application/x-www-form-urlencoded`** header.

```
POST /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName HTTP/1.1
Content-Type: application/x-www-form-urlencoded

volumeName=volumeName&volumeType=DISTRIBUTE&accessProtocols=GLUSTER,NFS&options=auth.allow=*
 &bricks=srvr1:dir1,srvr2:dir2
```

If your volumeType parameter is **`DISTRIBUTE`**, the cURL command is as follows:

`curl --request POST --user [USER:PASS] --data "volumeName=volumeName&volumeType=DISTRIBUTE&accessProtocols=GLUSTER,NFS&options=auth.allow=*&bricks=srvr1:dir1,srvr2:dir2" https://[host:port]/glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName`

Table 7.1. Parameters

Parameter	Description
volumeName	Name of the volume to be created
volumeType	**`DISTRIBUTE,REPLICATE, DISTRIBUTED_REPLICATE,STRIPE, DISTRIBUTED_STRIPE`**
replicaCount	mandatory only if volumeType = **`DISTRIBUTED_REPLICATE`**
stripeCount	mandatory only if volumeType = **`DISTRIBUTED_STRIPE`**
bricks	comma separated list of bricks; for example: **`server1:dir1,server2:dir2,… servern:dirn`**
accessProtocols	comma separated list of access protocols; such as **`GLUSTER, NFS, CIFS`**
cifsUsers	mandatory only if accessProtocols=CIFS; a comma separated list of CIFS users who should be allowed access to the volume (applicable only when enableCifs=true)
options	comma separated list of key-value pairs; optional; value must not contain '='

The API returns the following representation

```
HTTP/1.1 201 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName
```

7.4. Managing a Volume

The API starts, stops, re-balances the volume or rotates the logs of your volume in your cluster by issuing a **`PUT`** request to the **`URI`**.

```
PUT /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

cURL command:

`curl --request PUT --user [USER:PASS] --data "operation=[OPERATION]" https://[HOST:PORT]/glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName`

Table 7.2. Parameters

Parameter	Description
operation	**`start, stop, rebalanceStart, rebalanceStop,cifsConfig, logRotate`**.
force	true/false; optional – false by default; indicates if force option is to be used with the operation.
For operation=**`cifsConfig`**, the additional parameters required are:	

Parameter	Description
enableCifs	Flag indicating whether CIFS support is to be enabled/disabled for the volume. value = *`true/false`*
cifsUsers	Comma separated list of CIFS users who should be allowed access to the volume (applicable only when enableCifs=true)
For operation=**`rebalanceStart`**, the additional parameters required are:	
fixLayout	true/false; optional – false by default
migrateData	true/false; optional – true by default
forcedDataMigrate	true/false; optional – false by default

For long running operations such as **`rebalanceStart`**, the API returns the following representation:

```
HTTP/1.1 202 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/tasks/newTaskId
```

For other operations, the API returns the following representation:

```
HTTP/1.1 204 OK
```

7.5. Deleting a Volume

The API deletes a volume with a **`DELETE`** request sent to its **`URI`**.

```
DELETE /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName HTTP/1.1
```

cURL command:

`curl --request DELETE --user` ***`[USER:PASS]`*** **`--data "`*****`deleteData=true/false`*****`" https://`*****`[host:port]`*****`/glustermg/1.0.0alpha/clusters/`*****`myCluster`*****`/volumes/`*****`volumeName`***

Note

Setting the **`deleteData`** parameter to **`true`**, all data will be deleted from the volume. This can impact the performance of the system while the deletion is in progress, hence must be used with caution.

```
HTTP/1.1 204 No Content
```

7.6. Listing Volume Options Information

The API list information about all volume options with a **`GET`** request sent to its **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/volumes/options HTTP/1.1
```

```
Accept: application/xml
```

cURL command:

curl --request GET --header "Accept: application/xml" --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/volumes/options

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<options>
       <option>
          <name>optionName</name>
          <description>description</description>
          <defaultValue>defaultValue</defaultValue>
       </option>
       …
</options>
```

7.7. Setting Options on a Volume

The API sets options on a specified volume with a **POST** request sent to its **URI**.

```
POST /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/options HTTP/1.1
Content-Type: application/x-www-form-urlencoded

key=optionKey&value=optionValue
```

cURL command:

curl --request POST --user *[USER:PASS]* --data "key=*optionKey*&value=*optionValue*" https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/volumes/*volumeName*/options

```
HTTP/1.1 201 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/
options/optionKey
```

7.8. Reseting Volume Options

The API resets options on a specified volume with a **PUT** request sent to its **URI**.

```
PUT /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/options HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

cURL command:

curl --request PUT --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/volumes/*volumeName*/options

7.9. Retrieving Volume Logs

The API fetches the volume logs for a given criteria with a **GET** request sent to its **URI**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/logs HTTP/1.1
Accept: application/xml
```

cURL command:

`curl --request GET --header "Accept: application/xml" --user` *`[USER:PASS]`* `--data "brickName=`*`serverName:brickDirectory`*`&severity=`*`[SEVERITY]`*`&fromTimeStamp=`*`01/01/2010 00:00:00`*`&toTimeStamp=`*`02/01/2010 23:59:59`*`&lineCount=`*`100`*`" https://`*`[host:port]`*`/glustermg/1.0.0alpha/clusters/`*`myCluster`*`/volumes/volumeName/logs`

The valid values for severity are *EMERGENCY*, *ALERT*, *CRITICAL*, *ERROR*, *WARNING*, *NOTICE*, *INFO*, *DEBUG*, *TRACE*.

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<logMessages>
        <logMessage>
             <timeStamp>timestamp</timeStamp>
             <brick>serverName:brickDirectory</brick>
             <severity>severity</severity>
             <message>logMessage</message>
          </logMessage>
          ...
</logMessages>
```

7.10. Downloading Volume Logs

The API retrieves and downloads log files of all bricks of the volume in a compressed format with a **`GET`** request on the **`URI`**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/logs/download HTTP/1.1
application/octet-stream
```

cURL command:

`curl --request GET --header "Accept: application/octet-stream" --user` *`[USER:PASS]`* `https://`*`[host:port]`*`/glustermg/1.0.0alpha/clusters/`*`myCluster`*`/volumes/`*`volumeName`*`/logs/download`

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream

byte stream output of .tar.gz log files from all bricks of the volume
```

API Operations for Bricks

This chapter describes the brick operations of the Gluster Management Gateway API such as adding, migrating, and removing them from a volume.

8.1. Adding a Brick

The API adds a new brick to a volume in your cluster with a **`POST`** request to the **`URI`**.

```
POST /glustermg/1.0.0alpha/clusters/clusterName/volumes/volumeName/bricks HTTP/1.1
Content-Type: application/x-www-form-urlencoded

bricks=srvr1:dir1,srvr2:dir2,srvr3:dir3
```

cURL command:

```
curl -request POST --user [USER:PASS] --data
"bricks=srvr1:dir1,srvr2:dir2,srvr3:dir3" https://[host:port]/
glustermg/1.0.0alpha/clusters/clusterName/volumes/volumeName/bricks
```

The API returns the following representation:

```
HTTP/1.1 201 OK
Location:https://host:port/glustermg/1.0.0alpha/clusters/clusterName/volumes/volumeName/
bricks
```

8.2. Migrating a Brick

The API migrates data from one brick to another in your cluster by issuing a **`PUT`** request to the **`URI`**.

```
PUT /glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/bricks HTTP/1.1
Content-Type: application/x-www-form-urlencoded

source=srvr1:dir1&target=srvr2:dir2&autoCommit=true
```

cURL command:

```
curl --request PUT --user [USER:PASS] --data
"source=srvr1:dir1&target=srvr2:dir2&autoCommit=true" https://[HOST:PORT]/
glustermg/1.0.0alpha/clusters/myCluster/volumes/volumeName/bricks
```

Table 8.1. Parameters

Parameter	Description
source	source brick that you want to migrate, for example *`server:brickDirectory`*
target	destination brick, for example *`server:brickDirectory`*
autoCommit	optional. **`false`** by default.

For long running operations such as **`migrateStart`**, the API returns the following representation:

```
HTTP/1.1 202 OK
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/tasks/newTaskID
```

8.3. Removing a Brick

The API removes a brick(s) from a volume with a **`DELETE`** request sent to its **`URI`**.

```
DELETE /glustermg/1.0.0alpha/clusters/myCluster/volume/volumeName/bricks HTTP/1.1
```

cURL command:

```
curl --request DELETE --user [USER:PASS] --data
"bricks=srvr1:dir1,srvr2:dir2,srvr3:dir3&deleteData=true/false"
https://[host:port]/glustermg/1.0.0alpha/clusters/myCluster/
volumes/volumeName/bricks
```

Note

If you set the **`deleteData`** parameter to **`true`**, it will physically delete all data of the brick. It can impact performance of the system while the deletion is in progress, and hence must be used with caution.

```
HTTP/1.1 204 No Content
```

API Operations for Tasks

This chapter describes the tasks operations of the Gluster Management Gateway API.

9.1. Listing Tasks

A listing of all tasks running in your cluster is obtained by issuing a **GET** request on the resource **URI**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/tasks HTTP/1.1
Accept: application/xml
```

cURL command:

curl --request GET --header "Accept: application/xml" --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/tasks

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<tasks>
    <task>
        <id>taskId</id>
        <type>taskType</type>
        <reference>referenceEntityName</reference>
         <description>taskDescription</description>
         <pauseSupported>true/false</pauseSupported>
         <stopSupported>true/false</stopSupported>
         <commitSupported>false</commitSupported>
         <status>
            <code>statusCode</code>
            <message>statusMessage</message>
            <percentageSupported>true/false</percentageSupported>
            <percentCompleted>%completed</percentCompleted>
         </status>
    </task>
    ...
</tasks>
```

9.2. Retrieving Task Details

The API retrieves the details of a specified task by issuing a **GET** request on the resource **URI**.

```
GET /glustermg/1.0.0alpha/clusters/myCluster/tasks/taskID HTTP/1.1
Accept: application/xml
```

cURL command:

curl --request GET --header "Accept: application/xml" --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/tasks/*taskID*

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml
```

```
<task>
   <id>taskId<id>
   <type>taskType</type>
   <reference>referenceEntityName</reference>
   <description>taskDescription</description>
   <pauseSupported>true/false</pauseSupported>
   <stopSupported>true/false</stopSupported>
   <commitSupported>false</commitSupported>
   <status>
       <code>statusCode</code>
       <message>statusMessage</message>
       <percentageSupported>true/false</percentageSupported>
       <percentCompleted>%completed</percentCompleted>
   </status>
</task>
```

9.3. Managing a Task

The API manages a task such as pause or resume a task by issuing a **PUT** request sent to its resource **URI**.

```
PUT /glustermg/1.0.0alpha/clusters/myCluster/tasks/taskID HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

cURL command:

curl --request PUT --user *[USER:PASS]* --data "operation=*pause/resume*" https://*[HOST:PORT]*/glustermg/1.0.0alpha/clusters/*myCluster*/tasks/*taskID*

```
HTTP/1.1 204 No Content
Location: https://host:port/glustermg/1.0.0alpha/clusters/myCluster/tasks/taskID
```

9.4. Deleting a Task

The API aborts a running task with a **DELETE** request sent to its resource **URI**.

```
DELETE /glustermg/1.0.0alpha/clusters/myCluster/tasks/taskID HTTP/1.1
```

cURL command:

curl --request DELETE --user *[USER:PASS]* https://*[host:port]*/glustermg/1.0.0alpha/clusters/*myCluster*/tasks/*taskID*

```
HTTP/1.1 204 No Content
```

Appendix A. Revision History

Revision 1-0 **Tue Dec 13 2011**

Alpha Release Documentation

Index