

Sun Cloud Storage Administration API Specification
Version 0.2
Last Updated: 24 April 2009

Copyright © 2008-2009 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2008-2009 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays.

L'utilisation est soumise aux termes de la Licence.

Cette distribution peut comprendre des composants développés par des tierces parties.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font l'objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Table of Contents

Document Conventions.....	4
1 Introduction.....	4
1.1 Overview.....	4
1.2 Summary of Changes.....	4
2 Request and Response Messages.....	5
2.1 Transport Protocol.....	5
2.2 Media Types.....	5
2.3 Request Headers.....	6
2.4 Request Parameters.....	7
2.5 Response Headers.....	7
2.6 Message Bodies.....	8
2.7 Error Handling.....	8
3 Message Body Data Models.....	9
3.1 Top-Level Data Models.....	9
3.1.1 Resources Data Model.....	9
3.1.2 Resource Data Model.....	10
3.1.3 Version Data Model.....	11
3.2 Enclosed Data Models.....	11
3.2.1 Clone Data Model.....	11
3.2.2 Clones Data Model.....	11
3.2.3 Link Data Model.....	12
3.2.4 Metadata Data Model.....	13
3.2.5 Origin Data Model.....	13
3.2.6 Snapshot Data Model.....	13
3.2.7 Snapshots Data Model.....	14
3.2.8 Source Data Model.....	14
3.3 Error Message Data Models.....	14
4 Administrative Request Types.....	16
4.1 Create Volume.....	18
4.2 Delete Volume.....	20
4.3 Get Volume.....	21
4.4 Get Volumes.....	23
4.5 Create Snapshot.....	26
4.6 Delete Snapshot.....	28
4.7 Create Clone.....	30
4.8 Get Version Information.....	32

Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119 \(http://www.ietf.org/rfc/rfc2119.txt\)](http://www.ietf.org/rfc/rfc2119.txt).

1 Introduction

1.1 Overview

This document describes an HTTP/1.1-based RESTful API for requesting information and administrative services from the Cloud Storage Service. Through this API, you MAY administer server-side objects of the following types:

- *Volume* – The root directory of a server-side filesystem, into which you MAY store and retrieve directories (including subdirectories), and files.
- *Snapshot* – A named “backup copy” of the content of a volume, at a particular time.
- *Clone* – A volume whose contents were initialized from the contents of a particular snapshot.

This API supports the following request types:

- Create volume
- Delete volume
- Get volume
- Get volumes
- Create snapshot
- Delete snapshot
- Create clone
- Get version information

1.2 Summary of Changes

The following list identifies significant changes since the previous version of this specification:

- The “Get Volume Statistics” request type, and associated data model, has been removed. It will be replaced by the new Metering API, which will be common across all Sun Cloud services.
- A new “Get Version Information” request type, and associated data model, to support retrieving specification the version or versions supported by the server, as well as its implementation version.

2 Request and Response Messages

This section describes the general characteristics of requests that MAY be sent to the Cloud Storage Administration Service and responses that the service returns.

2.1 Transport Protocol

As mentioned previously, this API is based on version 1.1 of the *Hypertext Transport Protocol*¹, so any use of the API must conform to HTTP/1.1 requirements. In addition, the following additional requirement is imposed: All requests MUST be sent using the **https** protocol only.

2.2 Media Types

This API supports a number of top-level data models that can be transmitted in a request or received in a response. In addition, the service can accept and produce such data models in one of two syntaxes:

- Extensible Markup Language (XML)
- JavaScript Object Notation (JSON)²

You specify the combination of data model and representation syntax using an HTTP *media type*, which is used in two different headers on a request:

- As the value of a *Content-Type* header, describing the data model and representation syntax of the request body (if any).
- As the value (or one of the comma-delimited values) of an *Accept* header, describing the data model and representation syntax combination (or combinations) that you are prepared to receive as a response body. The server picks the response media type according to the rules in the HTTP/1.1 Specification, and indicates that choice in the *Content-Type* header included in the response.

Each of these top-level data models is identified by a *media type* specific to this API, which describes the combination of which data model, and which syntax, you are using. The valid media types are described in the following table.

1 <http://www.ietf.org/rfc/rfc2616.txt>

2 <http://www.ietf.org/rfc/rfc4627.txt>

Table 1: Media Types

Top Level Data Model	JSON Media Type	XML Media Type
Resources	x-network/resources+json	x-network/resources+xml
Resource	x-network/resource+json	x-network/resource+xml
Version	x-network/version+json	x-network/version+xml

For historical continuity and the widest possible interoperability, the service also supports the following generic media types:

- application/json – Any data model, in JSON syntax
- application/xml – Any data model, in XML syntax)
- text/xml – Any data model, in XML syntax)

However, clients SHOULD use the service specific media types described in Table 1.

2.3 Request Headers

In requests made with this API, several specific HTTP headers are used as described in the following table:

Table 2: Request Headers

Header	Supported Values	Description of Use	Required
Authorization	“Basic“ plus user name and password, encoded as required by HTTP Basic Authentication ³ .	Identifies authorized user making this request, determines access privileges	Yes, on all requests.
Content-Length	Length (in bytes) of message body.	Describes size of the included message body.	Yes, on requests that contain a message body or do a POST.
Content-Type	See Section 2.2.	Describes representation included in the message body of this request.	Yes, on requests that contain a message body.
X-Storage-Client-Specification-Version	“0.1”, “0.2”	Specifies the version of this API that this client was designed to support.	No. Current version assumed if not present.

Note that interactions with this API are stateless, so no Cookie header (or any other mechanism to provide a session identifier) is used.

2.4 Request Parameters

Certain request types support a set of optional request parameters that MAY be added to the request URI, and used for filtering or otherwise modifying the data to be returned by this request. The valid request parameters, and their effects on that request type, are included in the relevant request type descriptions in section 4.

2.5 Response Headers

In responses returned by the service, several specific HTTP headers are used as described in the following table:

³ <http://www.ietf.org/rfc/rfc2617.txt>

Table 3: Response Headers

Header	Supported Values	Description of Use	Required
Content-Length	Length (in bytes) of message body.	Describes size of the included message body.	Yes, on responses that contain a message body.
Content-Type	See Section 2.2.	Describes representation included in the message body of this response.	Yes, on responses that contain a message body.
Location	Canonical URI of a newly created resource.	Returns new URI on successful creation requests.	Yes, on requests that create new server-side resources.

2.6 Message Bodies

Based on the request type being performed (see section 4), a message body might be required. The nature of this request body is described by the *Content-Type* header included in the request, which must be one of the appropriate media types (see section 2.2) for this request type.

If you perform a request that returns a message body in the response (see section 4), your request should include an *Accept* header describing the media type or types that you are willing to accept. The response is returned with the selected media type in the *Content-Type* header.

2.7 Error Handling

Successful requests generally return an HTTP status code of 200 (OK), 201 (Created), or 204 (No Content) to indicate that the requested action has been performed. However, a number of things could go wrong. The various underlying causes are described by various HTTP status codes in the range 400-499 (for client side errors) and 500-599 (for server-side errors). The description of each request type (see section 4) contains a table describing the possible status codes that MAY be returned by that request type.

If a response is returned with an error status code (400-499 or 500-599), the server also returns a message body containing zero or more *messages* describing what went wrong. The text values of such messages might be used, for example, to communicate with a human user of the client-side application.

See Section 3.3 for details of the data models used to report error messages.

3 Message Body Data Models

This section contains abstract definitions of the *data models* relevant to this API, where a *data model* can be described as a specified set of *fields* whose values comprise the state of a particular server-side resource. Such data models can be organized hierarchically. That is, a field value in one data model might be a different data model, rather than a simple value like a String or an Integer.

As described in section 2.2, all of the request types supported by this API handle message bodies in either XML or JSON formats. The top-level data models in the next section describe those data models that might be the “outermost” in a request or response message body, and therefore correspond to an application-specific *media type*. The subsequent section describes the enclosed data models, which appear only as nested field values inside another data model.

3.1 Top-Level Data Models

3.1.1 Resources Data Model

A *resources* data model contains zero or more *resource* data models returned by the Get Volumes request type. When used as the outer element in a message body, it corresponds to a media type of *x-network/resources+json* or *x-network/resources+xml* depending on the desired representation. This data model contains the following fields:

Table 4: Resources Data Model Fields

Field Name	Type	Occurs ⁴	Description
link	Link	0..n	Hyperlink reference to related data. See section 3.1.1.1.
resource	Resource	0..n	Array of <i>resource</i> data models for the matching volumes.

3.1.1.1 Link rel Attribute Values

If your request included “pagination” request parameters (*count* and *offset*), the response includes *link* elements with *rel* (relationship) attribute values as follows:

- *next* – The *href* attribute can be used to retrieve the next logical “page” of results
- *previous* – The *href* attribute can be used to retrieve the previous “page” of results, if any. No previous page is returned if the current offset is zero).

⁴ In the data model descriptions, the *Occurs* column defines a range of the number of occurrences of this field that MAY be present in a response from the server. See the individual request type descriptions for details on which fields MUST, or MAY, be included in a request message body.

3.1.2 Resource Data Model

A *resource* data model represents the characteristics of an individual storage *volume* on the service. If used as the outer element in a message body, the resource data model corresponds to a media type of *x-network/resource+json* or *x-network/resource+xml* depending upon the desired representation. This data model contains the following fields:

Table 5: Resource Data Model Fields

Field Name	Type	Occurs	Description
name	String	1	Name of this volume.
links	Link[]	0..1	List of hyperlink references to related data. See section 3.1.2.1. Zero or more <i>link</i> data models may be nested inside.
clones	Clones	0..1	If a snapshot of this volume has been used as the source of one or more Create Clone requests, the resulting cloned volumes are listed here.
creation-date	Timestamp ⁵	1	Date and time this volume was created.
metadata	Metadata	1	Metadata associated with this volume
origin	Origin	0..1	If this volume was created through the Create Clone request type, the <i>origin</i> element describes the volume and snapshot from which it was created.
snapshots	Snapshots	0..1	If one or more snapshots of this volume have been created but not deleted, information about those snapshots is listed here.
source	Source	0	Never returned in a response. Used on a request to distinguish between “move” and “copy” operations.

3.1.2.1 Link rel Attribute Values

A returned *resource* data element contains zero or more *link* data models with *rel* (relationship) attribute values as follows:

- *self* – The *href* attribute can be used to perform an HTTP GET request for the information about this resource.
- *edit* – The *href* attribute can be used to perform an HTTP DELETE operation to delete this resource, or (if supported) an HTTP PUT operation to update it.
- *webdav* – The *href* attribute contains the absolute URI to use for accessing this volume

⁵ All date/time values in the data models of this API are expressed as the number of seconds since 01 January 1970 at 00:00:00 GMT.

through WebDAV APIs.

3.1.3 Version Data Model

A *version* data model contains information about the API specification version or versions supported by this instance of the server, as well as the implementation version of this instance. This data model includes the following fields:

Table 6: Version Data Model Fields

Field Name	Type	Occurs	Description
implementation-version	String	1	Implementation version number of this instance of this service.
specification-version	String	1	Comma-delimited list of API specification version numbers supported by this instance of this service.

3.2 Enclosed Data Models

The following data models do not appear as the outermost element in a message body. Therefore, they are not associated with any media types. These data models exist solely as nested field values within another data model.

3.2.1 Clone Data Model

A *clone* data model contains information about a clone that was created from the enclosing *resource*.

Table 7: Clone Data Model Fields

Field Name	Type	Occurs	Description
name	String	1	Name of the clone volume.
creation-date	Timestamp	1	Date and time this clone was created.
origin	Origin	1	Origin data model defining the volume and snapshot that were the source of this clone.
links	Link[]	0..1	List of hyperlink references to related data. Zero or more <i>Link</i> data models MAY be nested inside.

3.2.2 Clones Data Model

A *clones* data model contains zero or more *clone* data models representing the clones that have been created from snapshots of this volume. This data model contains the following fields:

Table 8: Clones Data Model Fields

Field Name	Type	Occurs	Description
clone	Clone	0..n	Array of <i>clone</i> data models for the clones of the enclosing <i>resource</i> .

3.2.3 Link Data Model

A *link* data model describes a relationship between the enclosing data model and a resource available at a specified URI. Where used in other data models, zero or more links MAY be present to describe different types of relationships. The set of valid relationship types is listed where this arrangement is possible. A *link* contains the following fields:

Table 9: Link Data Model Fields

Field Name	Type	Occurs	Description
href	String	1	URI this link points at for a relationship described by the <i>rel</i> field.
hreflang	String	0..1	Language code for the human language used in the resource found through the <i>href</i> field.
length	Integer	0..1	A hint about the size of the resource found through the <i>href</i> field.
rel	String	1	The type of relationship expressed by this link, between the enclosing data model and the resource found through the <i>href</i> field.
title	String	0..1	Human readable title describing the resource found through the <i>href</i> field.
type	String	0..1	The media type of the resource found through the <i>href</i> field.

The XML and JSON representations of a *link* data model were inspired by the way that <link> elements are used in the Atom Syndication Format, as well as in XHTML. Therefore, unlike other XML data models defined in this specifications, the fields are denoted by XML attributes, rather than subelement values. For example, a link in XML might appear like this:

```
<link rel="self" href="https://storage.cloud.sun.com/storage-server/resources/home"/>
```

The same link in JSON might appear like the following:

```
"link" : { "@rel" : "self",
           "@href" : "https://storage.cloud.sun.com/storage-server/resources/home"
```

}

3.2.4 Metadata Data Model

A *metadata* data model contains filesystem information about the volume described by the containing *resource* data model. This data model contains the following fields:

Table 10: Metadata Data Model Fields

Field Name	Type	Occurs	Description
md5	String	0..1	MD5 digest of this resource.
modification-date	Timestamp	0..1	Date and time this resource was last modified.
read-date	Timestamp	0..1	Date and time this resource was last modified.
size	Integer	0..1	Size of this resource, in bytes.
type	String	0..1	Type of this resource. For this version of the API, type must be “directory”.

The fields that are actually included in a returned *metadata* element depend on the value, if any, of the “metadata” request parameter included with this request. If no such parameter is included, the size and type fields are returned by default.

3.2.5 Origin Data Model

An *origin* data model describes the the original volume *resource*, and the original *snapshot*, from which a cloned volume was created. This data model contains the following fields:

Table 11: Origin Data Model Fields

Field Name	Type	Occurs	Description
creation-date	Timestamp	1	Date and time this clone was created.
name	String	1	Name of the original source volume.
snapshot	Snapshot	1	<i>Snapshot</i> data model describing the snapshot from which the containing clone volume was created.
links	Link[]	0..1	List of hyperlink references to related data. Zero or more <i>Link</i> data models MAY be nested inside.

3.2.6 Snapshot Data Model

A *snapshot* data model represents an individual snapshot that currently exists for the containing volume *resource*. This data model contains the following fields:

Table 12: Snapshot Data Model Fields

Field Name	Type	Occurs	Description
creation-date	Timestamp	1	Date and time this snapshot was created.
name	String	1	Name of this snapshot.
links	Link[]	0..1	List of hyperlink references to related data. Zero or more <i>Link</i> data models MAY be nested inside.

3.2.7 Snapshots Data Model

A *snapshots* data model contains zero or more *snapshot* data models representing the snapshots that have been created of this volume. This data model contains the following fields:

Table 13: Snapshots Data Model Fields

Field Name	Type	Occurs	Description
snapshot	Snapshot	0..n	Array of <i>snapshot</i> data models for the snapshots of the enclosing <i>resource</i> .

3.2.8 Source Data Model

A *source* data model describes the source of a rename volume, copy volume, or create clone request. This data model contains the following fields:

Table 14: Source Data Model Fields

Field Name	Type	Occurs	Description
name	String	1	Name of the source volume.
rename	Boolean	0..1	Flag describing whether this operation is a rename (true) or a move (false). Defaults to true.
snapshot	Snapshot	0..1	<i>Snapshot</i> data model describing the source snapshot for a Create Clone request.

3.3 Error Message Data Models

As described in section 2.7, responses with an HTTP status code of 4xx (client error) or 5xx (server error) include a set of zero or more messages describing what happened in more detail. The entire set of messages is contained in a *messages* data model with the following fields:

Table 15: Messages Data Model Fields

Field Name	Type	Occurs	Description
message	Message	0..n	Zero or more individual messages.

An individual *message* data model contains the following fields:

Table 16: Message Data Model Fields

Field Name	Type	Occurs	Description
code	String	0..1	Error code identifying the type of error documented by this message.
field	String	0..1	Name of the field (from the request data model) that this message is associated with. If not specified, this message should be considered global to the entire request.
text	String	1	Localized text describing the nature of the problem reported by this <i>message</i> .
hint	String	0..1	Optional localized text further describing the nature of the problem, possibly including potential workarounds that the client could try.
stack-trace	String	0..1	Stack trace associated with this message ⁶ .

⁶ For security reasons, stack traces SHOULD NOT be included in responses for public APIs. The *message* data model supports this field because it can also be used for internal services.

4 Administrative Request Types

The following sections describe patterns for performing various types of requests supported by this API. In all cases, two formats for the request URI are specified:

- *Canonical Form* – References resources for a specified user's *access key*⁷, where the access key might be your own (to access your resources), or the access key of a different user (to access that user's resources). Access to a resource for a different user is dependent on that user authorizing the type of access in which you want to engage.
- *Local Form* – References resources that belong to the user submitting the request, that is, the user whose credentials are included in the Authorization header.

For brevity, the following examples include only the local form, but either form is acceptable.

The example URIs in the descriptions of each request type include placeholders for several variable aspects that depend on the way that the Storage Service is configured. These variables are defined as follows:

- *{ServiceProtocol}* – “http” or “https”, defining the protocol used for interactions with the Storage Service.
- *{ServiceHost}* – Fully qualified hostname of the host providing the Storage Service.
- *{ServicePort}* – TCP/IP port number of the endpoint on *{ServiceHost}* for this service. Defaults to 80 or 443, depending on whether *{ServiceProtocol}* is “http” or “https”.
- *{ServicePrefix}* – The context path on which the Storage Service is deployed *{ServiceHost}*. If no such prefix exists, this is effectively be a zero length string.
- *{ServiceURI}* – The absolute URI of the storage service that you are contacting. This is a concatenation of the previously described variables in the following format (“.” and *{ServicePort}* are optional if the default value is used):
 - *{ServiceProtocol}://{ServiceHost}[:{ServicePort}][/{ServicePrefix}]*

The entire request URI depends on the form desired (canonical or local), as well as the details of the request type:

- Canonical Form – *{ServiceURI}/users/{AccessKey}/resources/{VolumeName}*
- Local Form – *{ServiceURI}/resources/{VolumeName}*

The example HTTP requests and responses shown in the request type descriptions in sections 4.1 through 4.8 have the following characteristics:

- A backslash (“\”) character at the end of a line indicates that the following content should actually appear on the same line as the current text.
- In both JSON and XML representations, whitespace is not significant. The example

⁷ The access key for a particular user is defined when that user registers with the Storage Service.

message bodies are formatted for readability, but actual response message bodies MAY or MAY NOT contain any whitespace between elements.

4.1 Create Volume

This request type causes a new volume named {VolumeName} to be created for the specified (or default) user. Volume names must be unique per user.

Synopsis (Canonical): POST {ServiceURI}/users/{AccessKey}/resources/{VolumeName}

Synopsis (Local): POST {ServiceURI}/resources/{VolumeName}

Request Headers: Authorization, Content-Length, Content-Type, X-Storage-Client-Specification-Version.

Request Parameters: N/A.

Request Message Body: An empty *resource* data model, with no required fields.

Response Headers: Location.

Response Message Body: N/A.

Response Status:

Table 17: Create Volume Response Status Codes

HTTP Status	Description
201 Created	New volume was successfully created at the URI returned in the <i>Location</i> header.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
409 Conflict	This volume name already exists.

Example: Create a new volume named “second”.

Table 18: Create Volume Example

Request	Response
<pre>POST \ {ServicePrefix}/resources/second Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Content-Length: nnn Content-Type: \ x-network/resource+xml X-Storage-Client-Specification- Version: 0.2 <?xml version="1.0"?> <resource/></pre>	<pre>HTTP/1.1 201 Created Location: {URI of new volume}</pre>

4.2 Delete Volume

This request type causes an existing volume named {VolumeName} to be deleted for the specified (or default) user.

Synopsis (Canonical): DELETE {ServiceURI}/users/{AccessKey}/resources/{VolumeName}

Synopsis (Local): DELETE {ServiceURI}/resources/{VolumeName}

Request Headers: Authorization, X-Storage-Client-Specification-Version.

Request Parameters: N/A.

Request Message Body: N/A.

Response Headers: N/A.

Response Message Body: N/A.

Response Status:

Table 19: Delete Volume Response Status Codes

HTTP Status	Description
204 No Content	Existing volume was successfully deleted.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	No volume with this name currently exists.
409 Conflict	Existing volume could not be deleted.

Example: Delete an existing volume named “second”.

Table 20: Delete Volume Example

Request	Response
DELETE \ {ServicePrefix}/resources/second Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx X-Storage-Client-Specification- Version: 0.2	HTTP/1.1 204 No Content

4.3 Get Volume

This request type retrieves information about an existing volume named {VolumeName}.

Synopsis (Canonical): GET {ServiceURI}/users/{AccessKey}/resources/{VolumeName}

Synopsis (Local): GET {ServiceURI}/resources/{VolumeName}

Request Headers: Accept, Authorization, X-Storage-Client-Specification-Version.

Request Parameters: The following request parameters are supported:

Table 21: Get Volume Request Parameters

Name	Type	Default	Description
metadata	String	size,type	Comma-delimited list of the metadata fields that you want to have retrieved. Options are: md5, modified-date, read-date, size, and type.

Request Message Body: N/A.

Response Headers: Content-Length, Content-Type.

Response Message Body: *Resource* data model.

Response Status:

Table 22: Get Volume Response Status Codes

HTTP Status	Description
200 OK	Information retrieval was successful.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	No volume with this name currently exists.

Example: Retrieve information about an existing volume named “home”. Of the available metadata, you want the MD5 checksum in addition to the default values.

Table 23: Get Volume Example

Request	Response
<pre>GET \ {ServicePrefix}/resources/home\ ?metadata=md5,size,type Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Accept: x-network/resource+xml X-Storage-Client-Specification- Version: 0.2</pre>	<pre>HTTP/1.1 200 OK Content-Type: \ x-network/resource+xml Content-Length: nnn <?xml version="1.0" \ encoding="UTF-8"?> <resource> <name>home</name> <links> <link rel="self" href="xxxxx"/> <link rel="edit" href="xxxxx"/> <link rel="webdav" href="xxxxx"/> </links> <metadata> <md5>xxxxxxxxxxxxxxxxxx</md5> <size>nnn</size> <type>directory</type> </metadata> </resource></pre>

4.4 Get Volumes

This request type retrieves information about zero or more existing volumes that match the specified filter criteria. If no filter criteria are specified, all volumes for the specified user (or you, if the local form is used) are returned (which could be a zero-length list if the user has no existing volumes).

Synopsis (Canonical): GET {ServiceURI}/users/{AccessKey}/resources

Synopsis (Local): GET {ServiceURI}/resources

Request Headers: Accept, Authorization, X-Storage-Client-Specification-Version.

Request Parameters: The following request parameters are supported:

Table 24: Get Volumes Request Parameters

Name	Type	Default	Description
count	Integer	No Limit	Maximum number of resources to return.
metadata	String	size,type	Comma-delimited list of the metadata fields that you want to have retrieved. Options are: md5, modified-date, read-date, size, and type.
offset	Integer	0	Zero-relative offset of the first resource to return.
order	String	N/A	Comma-delimited list of <i>resource</i> field names on which to sort the returned resources. Options are: name, creation-date.
regex	String	N/A	Regular expression against which to match names of resources to be returned.

Note that the *count* and *offset* request parameters support a client that wants to supply a paginated view of the returned resources. In this case, the returned *resources* data model includes two *link* elements with relationship values of “previous” and “next” that can be used to return the previous or next “page” of results. For example, to retrieve the second page of matching volumes displayed 10 at a time, you would use a URI like “{ServiceURI}/resources?offset=10&count=10”, and the response would include something like this:

```
<resources>
  <link rel="previous" href="{ServiceURI}/resources?offset=0&count=10"/>
  <link rel="next" href="{ServiceURI}/resources?offset=20&count=10"/>
  <resource>...</resource>
```

...

</resources>

When multiple request parameters are specified, the operations are applied in the following order:

- Filtering (metadata, regex)
- Sorting (order)
- Pagination (count, offset)

Request Message Body: N/A.

Response Headers: Content-Length, Content-Type.

Response Message Body: *Resources* data model.

Response Status:

Table 25: Get Volumes Response Status Codes

HTTP Status	Description
200 OK	Information retrieval was successful.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	No user for the specified access key currently exists (canonical form only).

Example: Retrieve information about all volumes that you own whose names start with a lowercase letter.

Table 26: Get Volumes Example

Request	Response
<pre>GET \ {ServicePrefix}/resources\ ?regex=[a-z] Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Accept: x-network/resources+xml X-Storage-Client-Specification- Version: 0.2</pre>	<pre>HTTP/1.1 200 OK Content-Type: \ x-networks/resource+xml Content-Length: nnn <?xml version="1.0" \ encoding="UTF-8"?> <resources> <resource> <name>home</name> <links> <link rel="self" href="xxxxx"/> <link rel="edit" href="xxxxx"/> <link rel="webdav" href="xxxxx"/> </links> <metadata> <size>nnn</size> <type>directory</type> </metadata> </resource> <resource> <name>second</name> <links> <link rel="self" href="xxxxx"/> <link rel="edit" href="xxxxx"/> <link rel="webdav" href="xxxxx"/> </links> <metadata> <size>nnn</size> <type>directory</type> </metadata> </resource> </resources></pre>

4.5 Create Snapshot

This request type causes a new snapshot named {SnapshotName} of an existing volume named {VolumeName} to be created for the specified (or default) user. Snapshot names must be unique per user, and must not match the name of any existing volumes.

Synopsis (Canonical): POST {ServiceURI}/users/{AccessKey}/snapshots/{VolumeName}\n/{SnapshotName}

Synopsis (Local):

POST {ServiceURI}/snapshots/{VolumeName}/{SnapshotName}

Request Headers: Authorization, Content-Length⁸, X-Storage-Client-Specification-Version.

Request Parameters: N/A.

Request Message Body: N/A.

Response Headers: Location.

Response Message Body: N/A.

Response Status:

Table 27: Create Snapshot Response Status Codes

HTTP Status	Description
201 Created	New snapshot was successfully created at the URI returned in the <i>Location</i> header.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	The volume name that you specified does not exist.
409 Conflict	This snapshot name already exists.

Example: Create a new snapshot named “today” for a volume named “second”.

⁸ Per the requirements described in section 2.3, a Content-Length header is required to satisfy HTTP/1.1 requirements on POST requests, even though no message body exists.

Table 28: Create Snapshot Example

Request	Response
POST \ {ServicePrefix}/snapshots\ /second/today Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Content-Length: 0 X-Storage-Client-Specification- Version: 0.2	HTTP/1.1 201 Created Location: {URI of new snapshot}

4.6 Delete Snapshot

This request type causes an existing snapshot named {SnapshotName} of an existing volume named {VolumeName} to be deleted for the specified (or default) user.

Synopsis (Canonical): DELETE {ServiceURI}/users/{AccessKey}/snapshots/{VolumeName}\

/ {SnapshotName}

Synopsis (Local):

DELETE {ServiceURI}/snapshots/{VolumeName}/{SnapshotName}

Request Headers: Authorization, X-Storage-Client-Specification-Version.

Request Parameters: N/A.

Request Message Body: N/A.

Response Headers: N/A.

Response Message Body: N/A.

Response Status:

Table 29: Delete Snapshot Response Status Codes

HTTP Status	Description
204 No Content	Delete operation was successful.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	The volume name that you specified does not exist.
409 Conflict	Existing snapshot could not be deleted.

Example: Delete an existing snapshot named “today” for a volume named “second”.

Table 30: Delete Snapshot Example

Request	Response
<pre>DELETE \ {ServicePrefix}/snapshots\ /second/today Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx X-Storage-Client-Specification- Version: 0.2</pre>	<pre>HTTP/1.1 204 No Content</pre>

4.7 Create Clone

This request type causes a new volume named {CloneName} to be created, based on the contents of snapshot {SnapshotName} of volume {VolumeName} for the specified (or default) user. Volume names must be unique per user.

Synopsis (Canonical): POST {ServiceURI}/users/{AccessKey}/resources/{CloneName}

Synopsis (Local): POST {ServiceURI}/resources/{CloneName}

Request Headers: Authorization, Content-Length, Content-Type, X-Storage-Client-Specification-Version.

Request Parameters: N/A.

Request Message Body: A *resource* data model, which must contain a *source* data model that includes the names of the volume and snapshot from which this clone will be created. See the next example.

Response Headers: Location.

Response Message Body: N/A.

Response Status:

Table 31: Create Clone Response Status Codes

HTTP Status	Description
201 Created	New volume was successfully created at the URI returned in the <i>Location</i> header.
400 Bad Request	Invalid parameter or field values on the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	You lack authorization to perform this request.
404 Not Found	The volume or snapshot name that you specified does not exist.
409 Conflict	The {CloneName} specified is the name of an existing volume.

Example: Create a new volume named “back” based on existing snapshot “today” of the existing volume “home”.

Table 32: Create Clone Example

Request	Response
<pre> POST \ {ServicePrefix}/resources/back Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Content-Length: nnn Content-Type: \ x-network/resource+xml X-Storage-Client-Specification- Version: 0.2 <?xml version="1.0"?> <resource> <source> <name>home</name> <snapshot> <name>today</name> </snapshot> </source> </resource> </pre>	<pre> HTTP/1.1 201 Created Location: {URI of new volume} </pre>

4.8 Get Version Information

This request type retrieves implementation and specification version information about this instance of this service.

Synopsis: GET {ServiceURI}/versions

Request Headers: Accept, Authorization, X-Storage-Client-Specification-Version

Request Parameters: N/A.

Request Message Body: N/A.

Response Headers: Content-Length, Content-Type

Response Message Body: *Versions* data model.

Response Status:

Table 33: Get Version Information Response Status Codes

HTTP Status	Description
200 OK	Information retrieval was successful.

Example:

Table 34: Get Version Information Example

Request	Response
<pre>GET \ {ServicePrefix}/versions Host: {ServiceHost} Authorization: Basic xxxxxxxxxxxxxx Accept: x-network/versions+xml X-Storage-Client-Specification- Version: 0.2</pre>	<pre>HTTP/1.1 200 OK Content-Type: \ x-network/versions+xml Content-Length: nnn <?xml version="1.0" \ encoding="UTF-8"?> <versions> <implementation-version> 12345 </implementation-version> <specification-version> 0.1,0.2 </specification-version> </versions></pre>