



Virtuozzo Hybrid Infrastructure 4.0

OBJECT STORAGE ORCHESTRATION API REFERENCE

Table of contents

1 About the guide	4
2 Authentication	5
3 User management	6
3.1 GET service ostor-users	6
3.1.1 Description	6
3.1.2 Requests	6
3.1.3 Responses	7
3.2 PUT service ostor-users	9
3.2.1 Description	9
3.2.2 Requests	9
3.2.3 Responses	10
3.3 DELETE service ostor-users	12
3.3.1 Description	12
3.3.2 Requests	12
3.3.3 Responses	13
3.4 GET service ostor-limits	13
3.4.1 Description	13
3.4.2 Requests	14
3.4.3 Responses	14
3.5 PUT service ostor-limits	16
3.5.1 Description	16
3.5.2 Requests	16
3.5.3 Responses	18
3.6 DELETE service ostor-limits	20
3.6.1 Description	20
3.6.2 Requests	20
3.6.3 Responses	21
3.7 GET service ostor-buckets	22
3.7.1 Description	22
3.7.2 Requests	23
3.7.3 Responses	24
4 Usage statistics	26
4.1 GET service ostor-usage	26
4.1.1 Description	26
4.1.2 Requests	26

4.1.3 Responses	27
4.2 DELETE service ostor-usage	29
4.2.1 Description	29
4.2.2 Requests	29
4.2.3 Responses	30

1 About the guide

The guide explains how to use the REST API to manage S3 clusters based on Virtuozzo Hybrid Infrastructure. The system API enables storage administrators to manage users, limits, and billing statistics. The system REST API enables remote execution of operations similar to `ostor-s3-admin` functionality.

2 Authentication

Management request must be authenticated with the AWS Access Key ID corresponding to the S3 system user. You can create system users with the `ostor-s3-admin create-user -S` command.

3 User management

3.1 GET service ostor-users

3.1.1 Description

Lists information about all users or the user specified by either email or ID.

3.1.2 Requests

Syntax

```
GET /?ostor-users HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-users&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-users&id=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service ostor-users parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	No*
id	User ID. Type: string. Default value: none.	No*

* Only one of the required parameters can be set in a single request.

If neither `emailAddress` nor `id` are set, the response is information about all users, otherwise the response is information about the user with the specified email or ID.

Headers

This implementation uses only common request headers.

3.1.3 Responses

Headers

This implementation uses only common response headers.

Body

AJSON dictionary with user information in the following format:

```
{
  "UserEmail" : "<email>"
  "UserId" : "<id>",
  "AWSAccessKeys" : [
    {
      "AWSAccessKeyId" : "<access_key>",
      "AWSSecretAccessKey" : "<secret_key>"
    }
  ]
}
```

Errors

Returns Error Code 400 if more than one parameter is set.

Examples

Sample request #1

Returns information about all users.

```
GET /?ostor-users HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 30 Apr 2016 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : keep-alive
x-amz-request-id : 800000000000000030003c6b538eadd95
Date: Wed, 30 Apr 2016 22:32:00 GMT
Connection:keep-alive
```

```
Content-type : application/json
[
{
  "UserEmail": "user@email.com",
  "UserId": "c5bf3c29f0a86585",
  "AWSAccessKeys": [
  {
    "AWSAccessKeyId": "c5bf3c29f0a865851KPQ",
    "AWSSecretAccessKey": "yqt3or2xMFn6mtvPH5Fdr9nbp2foDCKOCLYjCTb"
  }
  ],
  },
{
  "UserEmail": "root2@email.com",
  "UserId": "da2ccd035ce34bc3",
  "AWSAccessKeys": [
  {
    "AWSAccessKeyId": "da2ccd035ce34bc3XD5P",
    "AWSSecretAccessKey": "wHfEBQF07HN7fhoHx45lHHyBInA00CZTHtvveY1B"
  }
  ],
  },
{
  "UserEmail": "root0@email.com",
  "UserId": "f82c23f7823589eb",
  "AWSAccessKeys": [
  {
    "AWSAccessKeyId": "f82c23f7823589ebN4KD",
    "AWSSecretAccessKey": "MbKetIRMW8rrZh16yfb2dMj16ejHuBHfOa37bp5V"
  }
  ],
  },
{
  "UserEmail": "root1@email.com",
  "UserId": "fc06056891f36588",
  "AWSAccessKeys": [
  {
    "AWSAccessKeyId": "fc06056891f36588RMOE",
    "AWSSecretAccessKey": "HHD59Sf9KB4fG0xrjqhzyLBeHsODXD40QZeomKfy"
  }
  ]
  }
  ]
}
```

Sample request #2

Returns information about the user with the ID fc06056891f36588.

```
GET /?ostor-users&id=fc06056891f36588 HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 30 Apr 2016 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #2


```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : keep-alive
x-amz-request-id : 800000000000000030003c6b538eedd95
Date: Wed, 30 Apr 2016 22:32:00 GMT
Connection:keep-alive
Content-type : application/json
{
  "UserEmail": "root1@email.com",
  "UserId": "fc06056891f36588",
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "fc06056891f36588RMOE",
      "AWSSecretAccessKey": "HHD59Sf9KB4fG0xrjqhzyLBeHsODXD40QZeomKfy"
    }
  ]
}
```

3.2 PUT service ostor-users

3.2.1 Description

Creates a new user or generates/revokes access key pairs of existing users.

3.2.2 Requests

Syntax

```
PUT /?ostor-users&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
PUT /?ostor-users&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
PUT /?ostor-users&emailAddress=<value>&revokeKey=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

PUT service ostor-users parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	Yes
genKey	Generates a new access key pair for the user. A user can only have two key pairs. Type: flag. Default value: none.	No
revokeKey	Removes the access key pair that corresponds to the specified access key. Type: string. Default value: none.	No

If neither genKey nor revokeKey are set, a new user with the specified email will be created.

Headers

This implementation uses only common request headers.

3.2.3 Responses

Headers

This implementation uses only common response headers.

Body

If a new user is created or a key is generated, the body is a JSON dictionary with user information.

```
{
  "UserEmail" : "<email>"
  "UserId" : "<id>",
  "AWSAccessKeys" : [
    {
      "AWSAccessKeyId" : "<access_key>",
      "AWSSecretAccessKey" : "<secret_key>"
    }
  ]
}
```

If a key is revoked, the body is empty.

Examples

Sample request #1

Creates a user with the email test@test.test.

```
PUT /?ostor-users&emailAddress=test@test.test HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 16:01:03 GMT +3:00
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
x-amz-req-time-micros : 186132
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : keep-alive
X-amz-request-id : 800000000000000030003746059efad68
Date : Thu, 07 Apr 2016 13:01:08 GMT
Content-type : application/json

{
  "UserEmail": "test@test.test",
  "UserId": "a721fc1a64f13a05",
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "a721fc1a64f13a050QF4",
      "AWSSecretAccessKey": "VtzYY4ZHWYzbWLUrRMSzVhB07UvD6Z5nGsAPtESV"
    }
  ]
}
```

Sample request #2

Generates a new key pair for the user with the email user1@email.com.

```
PUT /?ostor-users&emailAddress=user1@email.com&genKey HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 15:51:13 GMT +3:00
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
x-amz-req-time-micros : 384103
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : closed
x-amz-request-id : 80000000000000003000374603639905b
Date : Thu, 07 Apr 2016 12:51:09 GMT
```

```
Content-type : application/json
```

```
{
  "UserEmail": "user1@email.com",
  "UserId": "8eaa6ab4749a29b4",
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "8eaa6ab4749a29b4034G",
      "AWSSecretAccessKey": "7spuMfShCI12tX6dFtS17TEP7ZQbIG11GgE0Emdy"
    },
    {
      "AWSAccessKeyId": "8eaa6ab4749a29b4EJUY",
      "AWSSecretAccessKey": "ELzQ8CTMfcYQCGSP51nGvmJxFC9xXrEJ4CjBAA2k"
    }
  ]
}
```

3.3 DELETE service ostor-users

3.3.1 Description

Deletes the user specified by email or ID.

3.3.2 Requests

Syntax

```
DELETE /?ostor-users&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

DELETE service ostor-users parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	Yes*
id	User ID. Type: string. Default value: none.	Yes*

* Only one of the required parameters can be set in a single request.

Headers

This implementation uses only common request headers.

3.3.3 Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Errors

Returns Error Code 400 if more than one required parameter is set.

Note

If a user is successfully deleted, Status204NoContent is returned.

Examples

Sample request

Deletes the user with the email test@test.test.

```
DELETE /?ostor-users&emailAddress=test@test.test HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 30 Apr 2016 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response

```
HTTP/1.1 203 No Content
x-amz-req-time-micros : 172807
Server : nginx/1.8.1
Connection : closed
x-amz-request-id : 800000000000000030005c8ca5862476a
Date : Wed, 30 Apr 2016 22:32:03 GMT
Content-type : application/xml
```

3.4 GET service ostor-limits

3.4.1 Description

Lists information about limits on operations and bandwidth for the specified user or bucket.

3.4.2 Requests

Syntax

```
GET /?ostor-limits&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-limits&bucket=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service ostor-limits parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	Yes*
id	User ID. Type: string. Default value: none.	Yes*
bucket	Bucket name. Type: string. Default value: none.	Yes*

* Only one of the required parameters can be set in a single request.

Headers

This implementation uses only common request headers.

3.4.3 Responses

Headers

This implementation uses only common response headers.

Body

AJSON dictionary with information about limits for a user or bucket in the following format:

```
{
  "ops:default" : "<default_limit_value_in_ops/sec>",
  "ops:get" : "<get_ops_limit_value_in_ops/sec>",
  "ops:put" : "<put_ops_limit_value_in_ops/sec>",
  "ops:list" : "<list_ops_limit_value_in_ops/sec>",
  "ops:delete" : "<delete_ops_limit_value_in_ops/sec>",
  "bandwidth:out" : "<bandwidth_limit_value_in_kb/sec>",
}
```

Zero value means “unlimited”.

Errors

Returns Error Code 400 if multiple parameters are set at once.

Note

The limits are disabled by default. If limits for a user/bucket requested are disabled, an error will be returned. Use PUT `ostor-limits` to enable limits.

Examples

Sample request #1

Returns information about limits for the user with the email `user1@email.com`.

```
GET /?ostor-limits&emailAddress=user1@email.com HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json

{
  "ops:default" : "0.50",
  "ops:get" : "0.50",
  "ops:put" : "0.50",
  "ops:list" : "0.50",
}
```

```
"ops:delete" : "0.50",
"bandwidth:out" : "0"
}
```

Sample request #2

Returns information about limits for the bucket bucket-1.

```
GET /?ostor-limits&bucket=bucket-1 HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 30 Apr 2016 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : closed
x-amz-request-id : 800000000000000030003c6b538eedd95
Date: Wed, 30 Apr 2016 22:32:00 GMT
Content-type : application/json
{
  "ops:default" : "0",
  "ops:get" : "0",
  "ops:put" : "0",
  "ops:list" : "0",
  "ops:delete" : "0",
  "bandwidth:out" : "3.33"
}
```

3.5 PUT service ostor-limits

3.5.1 Description

Sets limit values for the specified user or bucket. Either operations count or bandwidth limits can be specified in a single request.

3.5.2 Requests

Syntax

```
PUT /?ostor-limits&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```



```

GET /?ostor-limits&bucket=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>

```

Parameters

PUT Service ostor-limits parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	Yes*
id	User ID. Type: string. Default value: none.	Yes*
bucket	Bucket name. Type: string. Default value: none.	Yes
bandwidth	Enables bandwidth limits. Bandwidth limits types: { out kb/s } Type: flag.	Yes**
ops	Enables operations limits. If set, all unspecified bandwidth limits are set to 0. Operations limits types: { default ops/min, put ops/min, get ops/min, list ops/min, delete ops/min } Type: flag.	Yes**
default	Sets the default value for operations limits. If set, all unspecified operations limits are set to default, otherwise they are set to 0. Requires the ops subresource to be set. Type: integer. Default: 0.	No
put	Sets the PUT operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.	No

Parameter	Description	Required
get	Sets the GET operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.	No
delete	Sets the DELETE operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.	No
list	Sets the LIST operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.	No
out	Sets an outgoing bandwidth limit. Requires the ops subresource to be set. Type: integer. Default: 0.	No

* Only one of the required parameters can be set in a single request.

** Either ops or bandwidth can be set in a single request.

Zero value means "unlimited".

Headers

This implementation uses only common request headers.

3.5.3 Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Errors

Returns Error Code 400 if a wrong set of parameters is specified.

Examples

Sample request #1

Sets all operations limits for the user with the email user1@email.com to zero.

```
PUT /?ostor-limits&emailAddress=user1@email.com&ops&default=0 HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #2

Sets all operations limits for the user with the email user1@email.com to 1 ops/sec.

```
PUT /?ostor-limits&emailAddress=user1@email.com&ops&default=60 HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #3

Sets all bandwidth.out limit for the bucket testbucket to 50 kb/s.

```
PUT /?ostor-limits&bucket=testbucket&bandwidth&out=50 HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #3

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #4

Sets operations limits for the bucket `testbucket`. The new PUT operations limit is 60 ops/s, LIST limit is 0.5 ops/s, GET and DELETE limits are 1 ops/s.

```
PUT /?ostor-limits&bucket=testbucket&ops&default=60&put=3600&list=30 HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #4

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

3.6 DELETE service ostor-limits

3.6.1 Description

Sets a limit of the selected type to 0.0 (unlimited) for the specified user or bucket.

3.6.2 Requests

Syntax

```
DELETE /?ostor-limits&emailAddress=<value>&ops HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-limits&id=<value>&ops HTTP/1.1
Host: s3.amazonaws.com
```

```
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-limits&bucket=<value>&bandwidth HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

DELETE service ostor-limits parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	Yes*
id	User ID. Type: string. Default value: none.	Yes*
bucket	Bucket name. Type: string. Default value: none.	Yes*
ops	Removes operations limits.	No
bandwidth	Removes bandwidth limits.	No

* Only one of the required parameters can be set in a single request.

Headers

This implementation uses only common request headers.

3.6.3 Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Note

If limits are successfully removed, Status204NoContent will be returned.

Examples

Sample request #1

The following request deletes all operations limits for a user with the email user1@email.com.

```
PUT /?ostor-limits&emailAddress=user1@email.com&ops HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 204 No Content
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #2

The following request removes bandwidth limits for the bucket testbucket.

```
PUT /?ostor-limits&bucket=testbucket&bandwidth HTTP/1.1
Host: s3.amazonaws.com
Date: Thu, 07 Apr 2016 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 204 No Content
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 800000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

3.7 GET service ostor-buckets

3.7.1 Description

Lists information on all buckets or the buckets of the user specified by either email or ID.

3.7.2 Requests

Syntax

```
GET /?ostor-buckets HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-buckets&emailAddress=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-buckets&id=<value> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service ostor-buckets parameters

Parameter	Description	Required
emailAddress	User email address. Type: string. Default value: none.	No*
id	User ID. Type: string. Default value: none.	No*

* Only one of the required parameters can be set in a single request.

If neither `emailAddress` nor `id` are set, the response is the list of all buckets, otherwise the response is the bucket list of the user with the specified email or ID.

Headers

This implementation uses only common request headers.

3.7.3 Responses

Headers

This implementation uses only common response headers.

Body

AJSON dictionary with a bucket list in the following format:

```
{
  "Buckets": [
    {
      "name": <name>,
      "epoch": <epoch>,
      "creation_date": <date>,
      "owner_id": <id>,
      "size":
      {
        "current" : <cur>,
        "hmax": <hmax>,
        "h_integral": <hint>,
        "last_ts": <last_ts>
      }
    },
    {
      ...
    }
  ]
}
```

Errors

Returns Error Code 400 if more than one parameter is set.

Examples

Sample request

Returns information on all buckets in S3.

```
GET /?ostor-buckets HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 30 Apr 2016 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response


```

{
  "Buckets": [
    {
      "size": {
        "current": 12288,
        "h_integral": 7360512,
        "hmax": 12288,
        "last_ts": 424241
      },
      "epoch": 0,
      "owner_id": "ba7eba06129464c5",
      "name": "bucket1",
      "creation_date": "2018-05-25T17:12:00.000Z"
    },
    {
      "size": {
        "current": 46700160,
        "h_integral": 28160196480,
        "hmax": 46700160,
        "last_ts": 424237
      },
      "epoch": 0,
      "owner_id": "ccbec013d9fd3918",
      "name": "bucket2",
      "creation_date": "2018-05-25T13:51:55.000Z"
    },
    {
      "size": {
        "current": 12288,
        "h_integral": 8036352,
        "hmax": 12288,
        "last_ts": 424186
      },
      "epoch": 0,
      "owner_id": "9d80d59edbe2862a",
      "name": "bucket3",
      "creation_date": "2018-05-23T10:30:49.000Z"
    }
  ]
}

```

4 Usage statistics

The S3 gateway can collect usage statistics for S3 users and S3 buckets. The collected data are saved as regular objects. One such object contains statistics for the set usage period.

To enable statistics collection, set `S3_GW_USAGE_BUCKET` to `True` in the gateway configuration file (`/var/lib/ostor/local/gw.conf` by default).

Other options you may need to set are: `S3_GW_USAGE_PERIOD` (usage period in a single statistics object, in seconds) and `S3_GW_USAGE_CACHE_TIMEOUT` (the frequency of dumping statistics from memory to storage, in seconds).

4.1 GET service ostor-usage

4.1.1 Description

Lists existing statistics objects or queries information contained in a specified object.

4.1.2 Requests

Syntax

```
GET /?ostor-users HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-users&obj=object name HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>
```

Parameters

The parameter is specified by the `obj` subresource. If the `obj` subresource is undefined, the response contains information about all existing statistics objects. Otherwise information from the specified object `obj` is returned.

GET service ostor-usage parameters

Parameter	Description	Required
<code>obj</code>	Statistics object name. Type: string. Default value: none.	No

Headers

This implementation uses only common request headers.

4.1.3 Responses

Headers

This implementation uses only common response headers.

Body

If `obj` is unspecified:

```
{ "nr_items": number of statistics objects,
  "truncated": true if a list is truncated,
  "items": [ //list of statistics objects
    "first object's name",
    "s3-usage-obj1",
    "s3-usage-obj2",
    "s3-usage-obj3",
    ...
  ]
}
```

If `obj` is specified:

```
{ "fmt_version": version of response format,
  "service_id": id of a service that collected statistics,
  "start_ts": timestamp of statistics upload,
  "period": statistics upload period in seconds,
  "nr_items": number of counters,
  "items": [//list of usage counters
    {
      "key": { "bucket": "bucket-name", "epoch": bucket's epoch, "user_id": "user
id", "tag": "statistics object tag" },
      "counters": {
        "ops": { "put": count of put ops, "get": count of get ops, "list":
count of list ops, "other": count of other ops },
        "net_io": { "uploaded": number of uploaded bytes during the period,
"downloaded": number of downloaded bytes during the period }
      }
    },
    ...
  ]
}
```

Examples

Sample request #1

The following request returns information about all statistics objects.

```
GET /?ostor-usage /HTTP1.1
Date : Mon, 11 Apr 2016 16:43:16 GMT+3:00
Host : ostor-test-1
Authorization : <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
x-amz-req-time-micros : 404
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : keep-alive
x-amz-request-id : 80000000000000030006b6be3b0ae378
Date : Mon, 11 Apr 2016 13:43:16 GMT
Content-type : application/json

{ "nr_items": 9,
  "truncated": false,
  "items": [
    "s3-usage-8000000000000003-2016-04-11T13:10:29.000Z-1800",
    "s3-usage-8000000000000003-2016-04-11T13:12:53.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:13:23.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:15:53.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:16:23.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:31:54.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:33:25.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:33:55.000Z-30",
    "s3-usage-8000000000000003-2016-04-11T13:34:25.000Z-30"
  ]
}
```

Sample request #2

The following request returns information from the object s3-usage-8000000000000003-2016-04-11T13:33:55.000Z-30.

```
GET /?ostor-usage&obj=s3-usage-8000000000000003-2016-04-11T13:12:53.000Z-30
/HTTP1.1
Date: Mon, 11 Apr 2016 17:48:21 GMT+3:00
Host: ostor-test-1
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
X-amz-req-time-micros : 576
Transfer-encoding : chunked
Server : nginx/1.8.1
```

```

Connection : keep-alive
X-amz-request-id : 800000000000000030006b6bf23c77f09
Date : Mon, 11 Apr 2016 14:48:21 GMT
Content-type : application/json

{ "fmt_version": 1, "service_id":8000000000000003,
  "start_ts":1460380373, "period": 30, "nr_items":2,
  "items": [
    {
      "key": { "bucket": "bucket", "epoch":16394, "user_id": "f82c23f7823589eb",
"tag": "" },
      "counters": {
        "ops": { "put":15, "get":0, "list":1, "other":0 },
        "net_io": { "uploaded":99785, "downloaded":0 }
      }
    },
    {
      "key": { "bucket": "", "epoch":0, "user_id": "f82c23f7823589eb", "tag": ""
},
      "counters": {
        "ops": { "put":0, "get":2, "list":0, "other":0 },
        "net_io": { "uploaded":0, "downloaded":0 }
      }
    }
  ]
}

```

4.2 DELETE service ostor-usage

4.2.1 Description

Deletes the statistics object specified by name.

4.2.2 Requests

Syntax

```

DELETE /?ostor-users&obj=<object_name> HTTP/1.1
Host: s3.amazonaws.com
Date: <date>
Authorization: <authorization_string>

```

Parameters

DELETE service ostor-usage parameters

Parameter	Description	Required
obj	Statistics object name. Type: string. Default value: none.	No

Headers

This implementation uses only common request headers.

4.2.3 Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Note

If the request is successful, `Status204NoContent` is returned.

Examples

Sample request

The following request deletes statistics object with name `s3-usage-800000000000003-2016-04-11T13:33:55.000Z-30`.

```
DELETE /?ostor-usage&obj=s3-usage-800000000000003-2016-04-11T13:12:53.000Z-30
/HTTP1.1
Date : Mon, 11 Apr 2016 17:52:05 GMT+3:00
Host : ostor-test-1
Authorization : authorization string
```

Sample response

```
HTTP/1.1 204 No Content
Date : Mon, 11 Apr 2016 14:52:05 GMT
x-amz-req-time-micros : 4717
Connection : keep-alive
```

```
x-amz-request-id : 8000000000000030006b6bf31262d2c  
Server : nginx/1.8.1
```