Virtuozzo

Virtuozzo Infrastructure 7.3

Object Storage Orchestration API Reference

3/31/2026

Table of contents

About the guide

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

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.

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 on the gateway activity, do the following:

1. On each S3 node, create the gateway configuration file `/var/lib/ostor/local/gw.conf` with the following contents:

   ```
   # Enable usage statistics collection.
   S3_GW_COLLECT_STAT=1
   ```

   Other options you may need to set:

   `S3_GW_USAGE_PERIOD`

   Usage period in a single statistics object, in seconds.

   `S3_GW_USAGE_CACHE_TIMEOUT`

   Frequency of dumping statistics from memory to storage, in seconds.

2. On each S3 node, restart the gateway to apply the changes:

   ```
   # systemctl restart ostor-agentd
   ```

   Important
   Restarting the gateway is disruptive for the S3 service.

GET service ostor-usage

Description

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

Requests

Syntax

```
GET /?ostor-usage HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-usage&obj=<object_name> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET ?ostor-usage&after=<object_name>&limit=<number> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service ostor-usage parameters

Parameter	Description	Required
`obj`	Statistics object name. Type: string. Default value: none.	No
`after`	Object name after which to show the list of statistics objects. The specified name will not be shown in the response. Type: string. Default value: none.	No
`limit`	Number of statistics objects to show in the response. Type: integer. Default value: none.	No

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

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

If `obj` is unspecified:

```
{ "nr_items": <number_of_statistics_objects>,
  "truncated": <true_if_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_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_epoch_time>, "user_id":
"<user_id>", "tag": "<statistics_object_tag>" },
       "counters": {
               "ops": { "put":<count_of_put_operations>, "get": <count_of_get_
operations>, "list": <count_of_list_operations>, "other": <count_of_other_operations> },
               "net_io": { "uploaded": <number_of_uploaded_bytes_during_period>,
               "downloaded": <number_of_downloaded_bytes_during_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 : s3.example.com
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: s3.example.com
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 : 80000000000000030006b6bf23c77f09
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 }
       }
  }
 ]
}
```

Sample request #3

The following request returns information about 5 statistics objects after the object `s3-usage-8000000000000015-2024-02-15T11:56:49.000Z-1800`.

```
GET /?ostor-usage&after=s3-usage-8000000000000015-2024-02-15T11:56:49.000Z-1800&limit=5
/HTTP1.1
Date: Fri, 19 Apr 2024 15:07:53 GMT+3:00
Host: s3.example.com
Authorization: <authorization_string>
```

Sample response #3

```
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 : Fri, 19 Apr 2024 15:07:53 GMT
Content-type : application/json

{ "nr_items": 5,
  "truncated": false,
  "items": [
   "s3-usage-8000000000000015-2024-02-15T12:12:03.000Z-30",
   "s3-usage-8000000000000015-2024-02-15T12:15:21.000Z-30",
   "s3-usage-8000000000000015-2024-02-15T12:20:36.000Z-30",
   "s3-usage-8000000000000015-2024-02-15T12:25:40.000Z-30",
   "s3-usage-8000000000000015-2024-02-15T12:31:54.000Z-30"
  ]
}
```

DELETE service ostor-usage

Description

Deletes the statistics object specified by name.

Requests

Syntax

```
DELETE /?ostor-usage&obj=<object_name> HTTP/1.1
Host: <host>
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.

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-8000000000000003-2016-04-11T13:33:55.000Z-30`.

```
DELETE /?ostor-usage&obj=s3-usage-8000000000000003-2016-04-11T13:12:53.000Z-30 /HTTP1.1
Date : Mon, 11 Apr 2016 17:52:05 GMT+3:00
```

```
Host : s3.example.com
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 : 80000000000000030006b6bf31262d2c
Server : nginx/1.8.1
```

User management

This section describes how to manage S3 users and list S3 buckets.

GET service ostor-users

Description

Lists information about all users and their space usage, or the user specified by either email or ID.

Requests

Syntax

```
GET /?ostor-users HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

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

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

```
GET /?ostor-users&space HTTP/1.1
Host: <host>
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.	No*

Parameter	Description	Required
	Type: string. Default value: none.	
space	Show space usage. Type: flag. 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.

Responses

Headers

This implementation uses only common response headers.

Body

A JSON dictionary with user information in the following format:

```
{
"UserEmail" : "<email>"
"UserId" : "<id>",
"AWSAccessKeys : [
{
"AWSAccessKeyId" : "<access_key>",
"AWSSecretAccessKey" : "<secret_key>"
}]
}
{
"UserEmail": "<email>",
"UserId": "<id>",
"State": "<state>",
"OwnerId": "<id>",
"Flags": ["<flag>"],
"AWSAccessKeys": [
{
"AWSAccessKeyId": "<access_key>",
"AWSSecretAccessKey": "<secret_key>"
}],
"AccountCount": "<count>",
"Accounts": [
{
"Name": "<name>",
"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.example.com
Date: Wed, 24 Mar 2021 17:01:11 +0200
Authorization: <authorization_string>
```

Sample response #1

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection : keep-alive
x-amz-req-time-micros: 921
x-amz-request-id: 80000000000000016000060d778c73410
Date: Wed, 24 Mar 2021 15:01:11 GMT
Connection:keep-alive
Content-type : application/json{
  "Users": [
    {
      "UserEmail": "user1@email.com",
      "UserId": "b09693b73b3c7686",
      "State": "disabled",
      "OwnerId": "0000000000000000",
      "Flags": [
        "disabled"
      ]
    },
    {
      "UserEmail": "user2@email.com",
      "UserId": "bc6265392b818465",
      "State": "enabled",
      "OwnerId": "0000000000000000",
      "Flags": []
    },
    {
      "UserEmail": "user@example.com",
      "UserId": "f373d5175d1f3b63",
      "State": "enabled",
      "OwnerId": "0000000000000000",
      "Flags": [
        "system"
      ]
    }
  ]
}
```

Sample request #2

Returns information about the user with the ID b09693b73b3c7686.

```
GET /?ostor-users&id=b09693b73b3c7686 HTTP/1.1
Host: s3.example.com
Date: Wed, 24 Mar 2021 17:02:25 +0200
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Date: Wed, 24 Mar 2021 15:01:11 GMT
x-amz-req-time-micros: 983
x-amz-request-id: 8000000000000016000060d77d2db664
{
  "UserEmail": "user@email.com",
  "UserId": "b09693b73b3c7686",
  "State": "disabled",
  "OwnerId": "0000000000000000",
  "Flags": [
    "disabled"
  ],
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "b09693b73b3c7686FIGH",
      "AWSSecretAccessKey": "jO2p4JBN1tWc4FEGxwZ8qW2jPCJBYp8RJ4KgBcZP"
    }
  ],
  "AccountCount": "3",
  "Accounts": [
    {
      "Name": "account1",
      "AWSAccessKeys": [
        {
          "AWSAccessKeyId": "b09693b73b3c768613NV",
          "AWSSecretAccessKey": "CBUpFmnpUGlXskTivgDQu4qjYksWpceGZeH6Qyct"
        }
      ]
    },
    {
      "Name": "account2",
      "AWSAccessKeys": [
        {
          "AWSAccessKeyId": "b09693b73b3c7686LCZ5",
          "AWSSecretAccessKey": "xLpUDFJMFMO5rR9acAbUDplrPqIO6fneKNFjEB5c"
        },
        {
          "AWSAccessKeyId": "b09693b73b3c76866NI2",
          "AWSSecretAccessKey": "ajowU8pWSGW5ZJhA7AR9OjTrt11HmHPCJsMd247W"
        }
      ]
    },
    {
      "Name": "account3",
      "AWSAccessKeys": [
        {
          "AWSAccessKeyId": "b09693b73b3c7686OVV1",
          "AWSSecretAccessKey": "EOT652BDvByLwy2qPt0VsQ6s3I0pTrfPXKDw9i75"
        },
        {
          "AWSAccessKeyId": "b09693b73b3c7686Z8BU",
          "AWSSecretAccessKey": "m8PgWFLXPeJVSWojCE3DxWDoRk80g7CMyB7xK3Hd"
        }
      ]
    }
  ]
}
```

Sample request #3

Returns information about space usage.

```
GET /?ostor-users&space HTTP/1.1
Host: s3.example.com
Date: Wed, 23 Oct 2023 13:11:45 +0200
Authorization: <authorization_string>
```

Sample response #3

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
```

```
Connection : keep-alive
x-amz-req-time-micros: 921
x-amz-request-id: 8000000000000016000060d778c73410
Date: Wed, 23 Oct 2023 13:15:04 GMT
Connection:keep-alive
Content-type : application/json{
  "Users": [
    {
      "UserEmail": "user2@email.com",
      "UserId": "bc6265392b818465",
      "State": "enabled",
      "OwnerId": "0000000000000000",
      "SpaceStat":
        {
          "LastTs": 471425,
          "SizeCurr": 9323413504,
          "SizeHMax": 9323413504,
          "SizeHInt": 2414764097536
        },
      "Flags": []
    },
    {
      "UserEmail": "user@example.com",
      "UserId": "f373d5175d1f3b63",
      "State": "enabled",
      "OwnerId": "0000000000000000",
      "SpaceStat":
        {
          "LastTs": 0,
          "SizeCurr": 0,
          "SizeHMax": 0,
          "SizeHInt": 0
        },
      "Flags": [
        "system"
      ]
    }
  ]
}
```

PUT service ostor-users

Description

Creates a new user.

Requests

Syntax

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

Parameters

PUT service ostor-users parameters

Parameter	Description	Required
`emailAddress`	User email address. Type: string. Default value: none.	Yes

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

A JSON 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 multiple parameters are set at once.

Examples

Sample request

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

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

Sample response

```
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 : 80000000000000030003746059efad68
Date : Thu, 07 Apr 2016 13:01:08 GMT
Content-type : application/json
{
"UserEmail": "test@test.test",
"UserId": "a721fc1a64f13a05",
"AWSAccessKeys": [
{
"AWSAccessKeyId": "a721fc1a64f13a05OQF4",
"AWSSecretAccessKey": "VtzYY4ZHWYzbWLUrRMSzVhB07UvD6Z5nGsAPtESV"
}]
}
```

POST service ostor-users

Description

Generates or revokes access key pairs of existing users or accounts.

Requests

Syntax

```
POST /?ostor-users&emailAddress=<value>&genKey HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
POST /?ostor-users&emailAddress=<value>&revokeKey=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
POST /?ostor-users&emailAddress=<value>&accountName=<value>&genKey HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
POST /?ostor-users&emailAddress=<value>&accountName=<value>&revokeKey=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

POST service ostor-users parameters

Parameter	Description	Required
`emailAddress`	User email address. Type: string. Default value: none.	Yes
`accountName`	Account name. Type: string. Default value: none.	No
`genKey`	Generates a new access key pair for the user or account. A user or an account 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*

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

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

If 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

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

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

Sample response #1

```
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": "7spuMfShCIl2tX6dFtSl7TEP7ZQbIGl1GgE0Emdy"
    },
    {
      "AWSAccessKeyId": "8eaa6ab4749a29b4EJUY",
      "AWSSecretAccessKey": "ELzQ8CTMFcYQCGSP5lnGvmJxFC9xXrEJ4CjBAA2k"
    }
  ]
}
```

Sample request #2

Generates a new key pair for the account `account1` of the user with the email `user1@email.com`.

```
POST /?ostor-users&emailAddress=user1@email.com&accountName=account1&genKey HTTP/1.1
Host: s3.example.com
Date: Wed, 24 Mar 2021 17:32:41 +0200
Authorization: <authorization_string>
```

Sample response #2

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Date: Wed, 24 Mar 2021 15:32:42 GMT
x-amz-req-time-micros: 51835
x-amz-request-id: 80000000000000016000060d7e970100a
{
  "UserEmail": "user2@email.com",
  "UserId": "bc6265392b818465",
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "bc6265392b818465YQ0R",
      "AWSSecretAccessKey": "D6dSND8MZFSsKxp4bJFRXsCFEz3bC32nhpEzFpvP"
    }
  ]
}
```

Sample request #3

Revokes the key pair with the ID 8eaa6ab4749a29b4034G for the user with the email `user1@email.com`.

```
POST /?ostor-users&emailAddress=user1@email.com&revokeKey=8eaa6ab4749a29b4034G HTTP/1.1
Host: s3.example.com
Date: Wed, 24 Mar 2021 17:36:57 +0200
Authorization: <authorization_string>
```

Sample response #3

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Date: Wed, 24 Mar 2021 15:36:58 GMT
x-amz-req-time-micros: 43652
x-amz-request-id: 8000000000000016000060d7f8b178be
```

DELETE service ostor-users

Description

Deletes the user specified by email or ID.

Note
Objects and buckets owned by a deleted user are unaffected and continue to be stored in the system.

Requests

Syntax

```
DELETE /?ostor-users&emailAddress=<value> HTTP/1.1
Host: <host>
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.

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.example.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 : 80000000000000030005c8ca5862476a
Date : Wed, 30 Apr 2016 22:32:03 GMT
Content-type : application/xml
```

GET service ostor-buckets

Description

Lists information on all buckets, buckets of the user specified by either email or ID, as well as multiple or a single bucket specified by name.

Requests

Syntax

```
GET /?ostor-buckets HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

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

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

```
GET /?ostor-buckets&bucket=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-buckets&multi HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service ostor-buckets parameters

Parameter	Description	Required
`emailAddress`	User email address. Type: string.	No*

Parameter	Description	Required
	Default value: none.	
`id`	User ID. Type: string. Default value: none.	No*
`bucket`	Bucket name. Type: string. Default value: none.	No*
`multi`	Lists multiple buckets specified by their names in the request body. Type: flag. Default value: none.	No*

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

If none of the parameters is set, the response is the list of all buckets.

Headers

This implementation uses only common request headers.

Body

If `multi` is specified, the request body should contain a list of bucket names in the following format:

```
["<bucket_name1>", "<bucket_name2>", "<bucket_name3>"]
```

Responses

Headers

This implementation uses only common response headers.

Body

A JSON 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 #1

Returns information on all buckets in S3.

```
GET /?ostor-buckets HTTP/1.1
Host: s3.example.com
Date: Wed, 30 May 2018 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #1

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

Sample request #2

Returns information on all buckets owned by the S3 user with email `user1@example.com`.

```
GET /?ostor-buckets&emailAddress=user1@example.com HTTP/1.1
Host: s3.example.com
Date: Wed, 30 May 2018 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #2

```
{
"Buckets": [
    {
        "name": "bucket1",
        "epoch": 0,
        "creation_date": "2018-05-25T17:12:00.000Z",
        "owner_id": "ba7eba06129464c5",
        "size": {
            "current": 12288,
            "h_integral": 7360512,
            "hmax": 12288,
            "last_ts": 424241
            }
    },
    {
        "name": "bucket2",
        "epoch": 0,
        "creation_date": "2018-05-25T13:51:55.000Z",
        "owner_id": "ba7eba06129464c5",
        "size": {
            "current": 46700160,
            "h_integral": 28160196480,
            "hmax": 46700160,
            "last_ts": 424237
            }
    }
]}
```

Sample request #3

Returns information on the bucket with the name `bucket1`.

```
GET /?ostor-buckets&bucket=bucket1 HTTP/1.1
Host: s3.example.com
Date: Wed, 30 May 2018 22:32:00 GMT
Authorization: <authorization_string>
```

Sample response #3

```
{
"Buckets": [
    {
        "name": "bucket1",
        "epoch": 0,
        "creation_date": "2018-05-25T17:12:00.000Z",
        "owner_id": "ba7eba06129464c5",
        "size": {
            "current": 12288,
            "h_integral": 7360512,
            "hmax": 12288,
            "last_ts": 424241
            }
    }
]}
```

Sample request #4

Returns information on buckets specified in the request body.

```
GET /?ostor-buckets&multi HTTP/1.1
Host: s3.example.com
Date: Wed, 30 May 2018 22:32:00 GMT
Authorization: <authorization_string>
["bucket2", "bucket3"]
```

Sample response #4

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

Account management

This section describes how to manage S3 accounts, which are containers for S3 users with additional credentials.

POST service ostor-accounts

Description

Creates a new account.

Requests

Syntax

```
POST /?ostor-accounts&emailAddress=<value>&accountName=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
POST /?ostor-accounts&id=<value>&accountName=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

POST service ostor-accounts parameters

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

Responses

Headers

This implementation uses only common response headers.

Body

A JSON dictionary with account information in the following format:

```
{
"Name" : "<name>",
"AWSAccessKeys : [
{
"AWSAccessKeyId" : "<access_key>",
"AWSSecretAccessKey" : "<secret_key>"
}]
}
```

Examples

Sample request

Creates an account with the name `account1` for the user with the email `user1@email.com`.

```
POST /?ostor-accounts&emailAddress=user1@email.com&accountName=account1 HTTP/1.1
Host: s3.example.com
Date: Wed, 24 Mar 2021 14:37:10 GMT
Authorization: <authorization_string>
```

Sample response

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Date: Wed, 24 Mar 2021 14:37:11 GMT
x-amz-req-time-micros: 32753
x-amz-request-id: 800000000000016000060d722e695e2
{
  "Name": "account1",
  "AWSAccessKeys": [
    {
      "AWSAccessKeyId": "bc6265392b818465FQYC",
      "AWSSecretAccessKey": "iWs4rkwHMUYn8K0fPhjjAENC4QYUBIgIyJhNEx4l"
    }
  ]
}
```

DELETE service ostor-accounts

Description

Deletes an account of a user specified by email or ID.

Requests

Syntax

```
DELETE /?ostor-accounts&emailAddress=<value>&accountName=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-accounts&id=<value>&accountName=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

DELETE service ostor-accounts parameters

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

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 an account is successfully deleted, `Status204NoContent` is returned.

Examples

Sample request

Deletes the account with the name `account1` for the user with the email `user1@email.com`.

```
DELETE /?ostor-accounts&emailAddress=user1@email.com&accountName=account1 HTTP/1.1
Host: s3.example.com
Date: Wed, 24 Mar 2021 14:53:53 GMT
Authorization: <authorization_string>
```

Sample response

```
HTTP/1.1 204 No Content
Server: nginx
Content-Type: application/xml
Connection: keep-alive
Date: Wed, 24 Mar 2021 14:53:55 GMT
x-amz-req-time-micros: 47411
x-amz-request-id: 8000000000000016000060d75ec8e4dd
```

Limit management

This section describes operation rate limits and outgoing bandwidth limits that can be defined for S3 users and buckets.

GET service ostor-limits

Description

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

Requests

Syntax

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

```
GET /?ostor-limits&bucket=<value> HTTP/1.1
Host: <host>
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.

Responses

Headers

This implementation uses only common response headers.

Body

A JSON 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.example.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 : 80000000000000030005c8caec96d65b
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.example.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 : 80000000000000030003c6b538eedd95
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"
}
```

PUT service ostor-limits

Description

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

Requests

Syntax

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

```
PUT /?ostor-limits&bucket=<value> HTTP/1.1
Host: <host>
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	No

Parameter	Description	Required
	operations limits are set to `default`, otherwise they are set to 0. Requires the `ops` subresource to be set. Type: integer. Default: 0.	
`put`	Sets the PUT operations limit value. Requires the `ops` subresource to be set. Type: integer. Default: `default`.	No
`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.

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.example.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 : 80000000000000030005c8caec96d65b
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.example.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 : 80000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #3

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

```
PUT /?ostor-limits&bucket=testbucket&bandwidth&out=50 HTTP/1.1
Host: s3.example.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 : 80000000000000030005c8caec96d65b
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.example.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 : 80000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

DELETE service ostor-limits

Description

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

Requests

Syntax

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

```
DELETE /?ostor-limits&id=<value>&ops HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-limits&bucket=<value>&bandwidth HTTP/1.1
Host: <host>
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*

Parameter	Description	Required
`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.

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

Deletes all operations limits for a user with the email `user1@email.com`.

```
DELETE /?ostor-limits&emailAddress=user1@email.com&ops HTTP/1.1
Host: s3.example.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 : 80000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Sample request #2

Removes bandwidth limits for the bucket `testbucket`.

```
DELETE /?ostor-limits&bucket=testbucket&bandwidth HTTP/1.1
Host: s3.example.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 : 80000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/json
```

Quota management

This section describes storage usage quotas that can be defined for S3 users and buckets.

GET service ostor-quotas

Description

Lists information about quotas on storage usage for the specified user/bucket or for all users/buckets.

Requests

Syntax

```
GET /?ostor-quotas&emailAddress=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-quotas&bucket=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET /?ostor-quotas&default=<value> HTTP/1.1
Host: <host>
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.	Yes*

Parameter	Description	Required
	Type: string. Default value: none.	
`default`	Shows the default value for quotas. If set to `user`, shows the default quotas for all users. If set to `bucket`, shows the default quotas for all buckets. Type: string. Default value: none.	No

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

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

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

```
{
"version" : "<quota_version>",
"type" : "{0|1}",
"size" : "<usage_limit_value_in_bytes>"
}
```

For the `type` parameter, 0 means "user" and 1 means "bucket".

For the `size` parameter, zero value means “unlimited”.

Errors

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

Examples

Sample request #1

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

```
GET /?ostor-quotas&emailAddress=user1@email.com HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:53:41 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 20:53:46 GMT
Content-type : application/json
{
"version" : "1",
"type" : "0",
"size" : "1024"
}
```

Sample request #2

Returns information about quotas for the bucket `bucket1`.

```
GET /?ostor-quotas&bucket=bucket1 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:54:34 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 : 80000000000000030003c6b538eedd95
Date: Thu, 09 Sep 2021 20:54:37 GMT
Content-type : application/json
{
"version" : "1",
"type" : "1",
"size" : "256"
}
```

Sample request #3

Returns information about the default user quotas.

```
GET /?ostor-quotas&default=user HTTP/1.1
Host: s3.example.com
```

```
Date: Thu, 09 Sep 2021 20:57:48 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 20:57:51 GMT
Content-type : application/json
{
"version" : "1",
"type" : "0",
"size" : "1024"
}
```

Sample request #4

Returns information about the default bucket quotas.

```
GET /?ostor-quotas&default=bucket HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:58:05 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 : 80000000000000030003c6b538eedd95
Date: Thu, 09 Sep 2021 20:58:09 GMT
Content-type : application/json
{
"version" : "1",
"type" : "1",
"size" : "256"
}
```

PUT service ostor-quotas

Description

Sets a quota value for the specified user/bucket or for all users/buckets.

Requests

Syntax

```
PUT /?ostor-quotas&emailAddress=<value>&quota-size=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
PUT /?ostor-quotas&bucket=<value>&quota-size=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
PUT /?ostor-quotas&default=<value>&quota-size=<value> HTTP/1.1
Host: <host>
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*
`quota-size`	Sets the storage usage limit per user or bucket, as well as for all users or buckets, in gigabytes. Type: integer. Default: 0.	Yes
`default`	Sets the default value for quotas. If set to `user`, defines the default quotas for all users. If set to `bucket`, defines the default quotas for all buckets.	No

Parameter	Description	Required
	Type: string. Default: none.	

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

Zero value means “unlimited”.

Headers

This implementation uses only common request headers.

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 a quota for the user with the email `user1@email.com` to 1024 GB.

```
PUT /?ostor-quotas&emailAddress=user1@email.com&ops&quota-size=1024 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:21:35 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 : 80000000000000030005c8caec96d65b
Date: Thu, 09 Sep 2021 20:21:40 GMT
Content-type : application/json
```

Sample request #2

Sets a quota for the bucket `bucket1` to 256 GB.

```
PUT /?ostor-quotas&bucket=bucket1&quota-size=256 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:22:57 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 20:23:02 GMT
Content-type : application/json
```

Sample request #3

Sets the default user quotas to 1024 GB.

```
PUT /?ostor-quotas&default=user&quota-size=1024 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:24:15 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 20:24:19 GMT
Content-type : application/json
```

Sample request #4

Sets the default bucket quotas to 256 GB.

```
PUT /?ostor-quotas&default=bucket&quota-size=256 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 20:25:31
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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 20:25:36 GMT
Content-type : application/json
```

DELETE service ostor-quotas

Description

Sets a quota value to 0 (unlimited) for the specified user/bucket or for all users/buckets.

Requests

Syntax

```
DELETE /?ostor-quotas&emailAddress=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-quotas&bucket=<value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
DELETE /?ostor-quotas&default=<value> HTTP/1.1
Host: <host>
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.	Yes*

Parameter	Description	Required
	Type: string. Default value: none.	
`default`	Removes the default value for quotas. If set to `user`, deletes the default quotas for all users. If set to `bucket`, deletes the default quotas for all buckets. Type: string. Default: none.	No

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

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Note
If quotas are successfully deleted, `Status204NoContent` is returned.

Examples

Sample request #1

Deletes a quota for the user with the email `user1@email.com`.

```
DELETE /?ostor-quotas&emailAddress=user1@email.com HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 21:13:49 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 21:14:03 GMT
Content-type : application/json
```

Sample request #2

Deletes a quota for the bucket `bucket1`.

```
DELETE /?ostor-quotas&bucket=bucket1 HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 21:14:35 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 : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 21:14:39 GMT
Content-type : application/json
```

Sample request #3

Removes the default user quotas.

```
DELETE /?ostor-quotas&default=user HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 21:16:18 GMT
Authorization: <authorization_string>
```

Sample response #3

```
HTTP/1.1 204 No Content
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 21:16:22 GMT
Content-type : application/json
```

Sample request #4

Removes the default bucket quotas.

```
DELETE /?ostor-quotas&default=bucket HTTP/1.1
Host: s3.example.com
Date: Thu, 09 Sep 2021 21:17:01 GMT
Authorization: <authorization_string>
```

Sample response #4

```
HTTP/1.1 204 No Content
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 80000000000000030005c8caec96d65b
Date : Thu, 09 Sep 2021 21:17:05 GMT
Content-type : application/json
```

Replication management

This sections describes how to manage S3 cross-region replication (CRR) that enables copying of objects across S3 buckets in different regions.

GET service replication

Description

Lists information about replication configuration for the specified bucket.

Requests

Syntax

```
GET /?replication HTTP/1.1
Host: <bucket>.<host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

GET service replication parameters

Parameter	Description	Required
`bucket`	Bucket name. Type: string. Default value: none.	Yes

Headers

This implementation uses only common request headers.

Responses

Headers

Header	Description
`x-amz-geo-endpoint`	Endpoint of the remote region where to replicate objects to.
`x-amz-geo-access-key`	Access key of a user of the remote region used to replicate objects.
`x-amz-geo-access-secret`	Access secret of a user of the remote region used to replicate objects.

Body

An XML replication configuration in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::<user_id>:role/s3-replication-role</Role>
   <Rule>
      <Status>Enabled|Disabled</Status>
      <Priority>1</Priority>
      <DeleteMarkerReplication>
         <Status>Enabled|Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix />
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::<destination_bucket></Bucket>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

Examples

Sample request

Returns replication configuration of the bucket `test`.

```
GET /?replication HTTP/1.1
Host: test.s3.example.com
Date: Tu, 18 Jan 2021 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 80000000000000030005c8caec96d65b
Date : Thu, 07 Apr 2016 14:08:56 GMT
Content-type : application/xml
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::850b4943d62191a5:role/s3-replication-role</Role>
   <Rule>
      <Status>Enabled</Status>
      <Priority>1</Priority>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix />
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2</Bucket>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

PUT service replication

Description

Sets replication configuration for the specified bucket.

Requests

Syntax

```
PUT /?replication HTTP/1.1
Host: <bucket>.<host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

PUT service replication parameters

Parameter	Description	Required
`bucket`	Bucket name. Type: string. Default value: none.	Yes
`user_id`	ID of the user that is used to replicate objects on your behalf. Type: string. Default: none.	Yes
`destination_bucket`	The name of the bucket where you want to store the results. Type: string. Default: none.	Yes

Body

An XML replication configuration in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::<user_id>:role/s3-replication-role</Role>
   <Rule>
      <Status>Enabled|Disabled</Status>
      <Priority>1</Priority>
      <DeleteMarkerReplication>
         <Status>Enabled|Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix />
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::<destination_bucket></Bucket>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

Headers

Header	Description
`x-amz-geo-endpoint`	Endpoint of the remote region where to replicate objects to.
`x-amz-geo-access-key`	Access key of a user of the remote region used to replicate objects.
`x-amz-geo-access-secret`	Access secret of a user of the remote region used to replicate objects.

Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Examples

Sample request

Sets replication configuration for the bucket. `test`.

```
PUT/?replication HTTP/1.1
Host: test.s3.example.com
Date: Tu, 18 Jan 2021 14:08:55 GMT
Authorization: <authorization_string>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
   <Role>arn:aws:iam::850b4943d62191a5:role/s3-replication-role</Role>
   <Rule>
      <Status>Enabled</Status>
      <Priority>1</Priority>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix />
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2</Bucket>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

Sample response

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 80000000000000030005c8caec96d65b
Date : Tu, 21 Jan 2021 14:08:56 GMT
```

DELETE service replication

Description

Deletes replication configuration for the specified bucket.

Requests

Syntax

```
DELETE /?replication HTTP/1.1
Host: <bucket>.<host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

DELETE service replication parameters

Parameter	Description	Required
`bucket`	Bucket name.	Yes

Parameter	Description	Required
	Type: string. Default value: none.	

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

Empty.

Examples

Sample request

Deletes replication configuration of the bucket `test`.

```
DELETE/?replication HTTP/1.1
Host: test.s3.example.com
Date: Tu, 18 Jan 2021 14:08:55 GMT
Authorization: <authorization_string>
```

Sample response

```
HTTP/1.1 200 OK
Transfer-encoding : chunked
Server : nginx/1.8.1
Connection: closed
x-amz-request-id : 80000000000000030005c8caec96d65b
Date : Tu, 21 Jan 2021 14:08:56 GMT
```

Object storage configuration

This section describes how to configure the object storage parameters, such as:

- Maximum number of object services
- Maximum size of the object service
- Automatic split behavior
- Maximum number of concurrent splits
- Default Cross-origin resource sharing (CORS) behavior

GET service ostor-settings

Description

Lists existing object storage settings.

Requests

Syntax

```
GET /?ostor-settings HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

Empty.

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

A JSON dictionary with a list of settings in the following format:

```
{
  "OS.max_count": <value>,
  "OS.max_size": <value>,
  "cfg.autosplit.enabled": <value>,
  "cfg.autosplit.max_active": <value>,
  "ostor.default_cors.enabled": <value>
}
```

Examples

Sample request

The following request returns information about all object storage settings.

```
GET /?ostor-settings /HTTP1.1
Date : Mon, 14 Nov 2023 14:39:21 GMT+3:00
Host : s3.example.com
Authorization : <authorization_string>
```

Sample response

```
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 : 80000000000000030008e3ac6b0bc436
Date : Mon, 14 Nov 2023 14:39:22 GMT
Content-type : application/json

{
  "OS.max_count": 100,
  "OS.max_size": 1000,
  "cfg.autosplit.enabled": 1,
  "cfg.autosplit.max_active": 1,
  "ostor.default_cors.enabled": 1
}
```

PUT service ostor-settings

Description

Changes existing object storage settings.

Requests

Syntax

```
PUT /?ostor-settings&name=<parameter_name>&value=<parameter_value> HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

Parameters

PUT service ostor-settings parameters

Parameter name	Description	Possible values	Default value
`OS.max_count`	Maximum number of object services. When this value is reached, the automatic split stops.	1–65536	100
`OS.max_size`	Maximum size of the object service, in gigabytes. When this value is reached, the automatic split is performed.	1–16000	1000
`cfg.autosplit.enabled`	Enables or disables the automatic split.	0/1	1
`cfg.autosplit.max_active`	Maximum number of splits ongoing at the same time.	0–2^31	1
`ostor.default_cors.enabled`	Default CORS behavior if no policy is specified. To allow all CORS, specify 1. Otherwise, specify 0.	0/1	1

Headers

This implementation uses only common request headers.

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

The following request updates the default CORS behavior.

```
PUT /?ostor-settings&name=ostor.default_cors.enabled&value=0 /HTTP1.1
Date : Mon, 14 Nov 2023 14:41:05 GMT+3:00
Host : s3.example.com
Authorization : <authorization_string>
```

Sample response

```
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 : 80000000000000030008e3ac6b0bc436
Date : Mon, 14 Nov 2023 14:41:07 GMT
Content-type : application/json
```

Object storage health check

This section describes how to check object storage health.

GET service ostor-health

Description

Lists information about object storage health.

Requests

Syntax

Note
The health check API does not require authorization.

```
GET /?ostor-health HTTP/1.1
Host: <host>
```

```
GET /?ostor-health&maintenance=<true|false> HTTP/1.1
Host: <host>
```

Parameters

GET service ostor-users parameters

Parameter	Description	Required
`maintenance`	Checks if the host running the S3 GW services is in the requested maintenance state. Type: boolean. Default value: none.	No

Headers

This implementation uses only common request headers.

Responses

Headers

This implementation uses only common response headers.

Body

If the service is healthy and reachable, returns `200 OK` status code.

With the `maintenance` parameter, returns `200 OK` if the host is in the requested maintenance state and `412 Precondition Failed` if the host state differs.

Examples

Sample request #1

Returns information about object storage health.

```
GET /?ostor-health HTTP/1.1
Host: s3.example.com
```

Sample response #1

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 15 Oct 2025 12:15:23 GMT
Transfer-Encoding: chunked
Connection: close
```

Sample request #2

Checks if the host is in the maintenance state.

```
GET /?ostor-health&maintenance=true HTTP/1.1
Host: s3.example.com
```

Sample response #2

```
HTTP/1.1 412 Precondition Failed
Server: nginx
Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Date: Wed, 05 Nov 2025 12:19:19 GMT
x-amz-req-time-micros: 83
x-amz-request-id: 800000000000002e0002ba950159f905
```