Acronis Cyber Infrastructure

4.6

Object Storage Orchestration API Reference

REVISION: 7/28/2021
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 ..................................................................................... 10
       3.2.1 Description ............................................................................................ 10
       3.2.2 Requests ............................................................................................... 10
       3.2.3 Responses .............................................................................................. 11
   3.3 POST service ostor-users ................................................................................... 12
       3.3.1 Description ............................................................................................ 12
       3.3.2 Requests ............................................................................................... 12
       3.3.3 Responses ............................................................................................. 13
   3.4 DELETE service ostor-users ............................................................................. 15
       3.4.1 Description ............................................................................................ 15
       3.4.2 Requests ............................................................................................... 16
       3.4.3 Responses ............................................................................................. 16
   3.5 POST service ostor-accounts .......................................................................... 17
       3.5.1 Description ............................................................................................ 17
       3.5.2 Requests ............................................................................................... 17
       3.5.3 Responses ............................................................................................. 18
   3.6 DELETE service ostor-accounts .................................................................... 19
       3.6.1 Description ............................................................................................ 19
       3.6.2 Requests ............................................................................................... 19
       3.6.3 Responses ............................................................................................. 20
   3.7 GET service ostor-limits .................................................................................. 21
       3.7.1 Description ............................................................................................ 21
       3.7.2 Requests ............................................................................................... 21
       3.7.3 Responses ............................................................................................. 22
   3.8 PUT service ostor-limits .................................................................................. 24
       3.8.1 Description ............................................................................................ 24
       3.8.2 Requests ............................................................................................... 24
       3.8.3 Responses ............................................................................................. 26
1 About the guide

The guide explains how to use the REST API to manage S3 clusters based on Acronis Cyber 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` 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: <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>

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>User ID.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>

* 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.
Header
This implementation uses only common request headers.

3.1.3 Responses

Headers
This implementation uses only common response headers.

Body
A JSON dictionary with user information in the following format:

```json
{
    "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: 800000000000016000060d778c73410
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.

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": "CBUpFmnpUglXskTivgDQu4qjYksWpceGZeH6yqct"
        }
      ]
    },
    {
      "Name": "account2",
      "AWSAccessKeys": [
        {
          "AWSAccessKeyId": "b09693b73b3c7686LCZ5",
          "AWSSecretAccessKey": "xLpUDFJMFMO5rR9acAbUDp1rpQIO6flnKNFjEB5c"
        },
        {
          "AWSAccessKeyId": "b09693b73b3c7686NI2",
          "AWSSecretAccessKey": "ajowU8pWSG5ZJHa7AR90jTnt1HmHhCJscMd247W"
        }
      ]
    }
  ]
}
3.2 PUT service ostor-users

3.2.1 Description

Creates a new user.

3.2.2 Requests

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>

Headers

This implementation uses only common request headers.
3.2.3 Responses

Headers
This implementation uses only common response headers.

Body
A JSON dictionary with user information in the following format:

```json
{
    "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 : 800000000000000030003746059efad68
Date : Thu, 07 Apr 2016 13:01:08 GMT
Content-type : application/json
{
    "UserEmail": "test@test.test",
    "UserId": "a721fca64f13a05",
```
"AWSAccessKeys": [
{
"AWSAccessKeyId": "a721fc1a64f13a05QF4",
"AWSSecretAccessKey": "VtzYY4ZHvzWLuRMSzVhB07UvD6Z5nPAPtESV"
}
]

3.3 POST service ostor-users

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

3.3.2 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>Yes</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>accountName</td>
<td>Account name.</td>
<td>No</td>
</tr>
<tr>
<td>Type: string.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value: none.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>genKey</td>
<td>Generates a new access key pair for the user or account. A user or an account can only have two key pairs.</td>
<td>No*</td>
</tr>
<tr>
<td>Type: flag.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value: none.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>revokeKey</td>
<td>Removes the access key pair that corresponds to the specified access key.</td>
<td>No*</td>
</tr>
<tr>
<td>Type: string.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value: none.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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**

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

```json
{
  "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 : 800000000000003000374603639905b
Date : Thu, 07 Apr 2016 12:51:09 GMT
Content-type : application/json
{
    "UserEmail" : "user1@email.com",
    "UserId" : "8eaa6ab4749a29b4",
    "AWSAccessKeys" : [
        {
            "AWSAccessKeyId" : "8eaa6ab4749a29b4034G",
            "AWSSecretAccessKey" : "7spuMfShCI12tX6dFtS17TEP7ZQbIG1GgE0Emdy"
        },
        {
            "AWSAccessKeyId" : "8eaa6ab4749a29b4EJUY",
            "AWSSecretAccessKey" : "ELzQ8CTMFCyQCGSP5nGvmJxFC9xXrEJ4CjBAA2k"
        }
    ]
}
```

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
```
Sample request #3

Reverses the key pair with the ID 8ea6ab4749a29b4034G for the user with the email user1@email.com.

POST /?ostor-users&emailAddress=user1@email.com&revokeKey=8ea6ab4749a29b4034G
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: 800000000000000016000600d7e970100a

3.4 DELETE service ostor-users

3.4.1 Description

Deletes the user specified by email or ID.
3.4.2 Requests

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address. Type: string. Default value: none.</td>
<td>Yes*</td>
</tr>
<tr>
<td>id</td>
<td>User ID. Type: string. Default value: none.</td>
<td>Yes*</td>
</tr>
</tbody>
</table>

* 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

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.

```plaintext
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

```plaintext
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
```

3.5 POST service ostor-accounts

3.5.1 Description

Creates a new account.

3.5.2 Requests

Syntax

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string. Default value: none.</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>User ID.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string. Default value: none.</td>
<td></td>
</tr>
<tr>
<td>accountName</td>
<td>Account name.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Type: string. Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>

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

Headers
This implementation uses only common request headers.

3.5.3 Responses

Headers
This implementation uses only common response headers.

Body
A JSON dictionary with account information in the following format:

```json
{
  "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.

```plaintext
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: 800000000000000001600060d722e695e2

{  
  "Name": "account1",
  "AWSAccessKeys": [  
    {  
      "AWSAccessKeyId": "bc6265392b818465FQYC",
      "AWSSecretAccessKey": "iWs4rkwHMUYn8K0fPhjjAENC4QYUBIGyJhNEx41"
    }
  ]
}
```

3.6 DELETE service ostor-accounts

3.6.1 Description

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

3.6.2 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>User ID.</td>
<td>No*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>accountName</td>
<td>Account name.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>

* 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.

Errors

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

Note

If an account is successfully deleted, Status 204 NoContent is returned.

Examples

Sample request

Deletes the account with the name account1 for the user with the email user1@email.com.
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: 800000000000001600060d75ec8e4dd

3.7 GET service ostor-limits

3.7.1 Description

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

3.7.2 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>id</td>
<td>User ID.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>bucket</td>
<td>Bucket name.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>

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

Headers

This implementation uses only common request headers.

3.7.3 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:

```json
{
    "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 : 80000000000000030003c6b538eecd95
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"
}
3.8 PUT service ostor-limits

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

3.8.2 Requests

Syntax

PUT /?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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
</table>
| 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** |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>ops</td>
<td>Enables operations limits. If set, all unspecified bandwidth limits are set to 0. Operations limits types: ( default</td>
<td>Yes**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ops/min, put</td>
</tr>
<tr>
<td></td>
<td>Type: flag.</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>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.</td>
<td>No</td>
</tr>
<tr>
<td>put</td>
<td>Sets the PUT operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.</td>
<td>No</td>
</tr>
<tr>
<td>get</td>
<td>Sets the GET operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.</td>
<td>No</td>
</tr>
<tr>
<td>delete</td>
<td>Sets the DELETE operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.</td>
<td>No</td>
</tr>
<tr>
<td>list</td>
<td>Sets the LIST operations limit value. Requires the ops subresource to be set. Type: integer. Default: default.</td>
<td>No</td>
</tr>
<tr>
<td>out</td>
<td>Sets an outgoing bandwidth limit. Requires the ops subresource to be set. Type: integer. Default: 0.</td>
<td>No</td>
</tr>
</tbody>
</table>

* 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.

3.8.3 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.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: 8000000000000030005c8caec96d65b
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
```
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 bandwidth.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
3.9 DELETE service ostor-limits

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

3.9.2 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>User ID.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
<tr>
<td>bucket</td>
<td>Bucket name.</td>
<td>Yes*</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>ops</td>
<td>Removes operations limits.</td>
<td>No</td>
</tr>
<tr>
<td>bandwidth</td>
<td>Removes bandwidth limits.</td>
<td>No</td>
</tr>
</tbody>
</table>

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

### Headers

This implementation uses only common request headers.

#### 3.9.3 Responses

### Headers

This implementation uses only common response headers.

### Body

Empty.

---

**Note**

If limits are successfully removed, Status 204 NoContent 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.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**

The following request removes bandwidth limits for the bucket testbucket.
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

3.10 GET service ostor-buckets

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

3.10.2 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>
Parameters

**GET service ostor-buckets parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>User email address. Type: string. Default value: none.</td>
<td>No*</td>
</tr>
<tr>
<td>id</td>
<td>User ID. Type: string. Default value: none.</td>
<td>No*</td>
</tr>
</tbody>
</table>

* 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.

3.10.3 Responses

Headers

This implementation uses only common response headers.

Body

A JSON dictionary with a bucket list in the following format:

```json
{
  "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.example.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/oster/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 /?oster-users HTTP/1.1
Host: <host>
Date: <date>
Authorization: <authorization_string>
```

```
GET /?oster-users&obj=object name HTTP/1.1
Host: <host>
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Statistics object name.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Type: string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default value: none.</td>
<td></td>
</tr>
</tbody>
</table>
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:

```json
{ "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:

```json
{ "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.

```plaintext
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-8000000000000003-2016-04-11T13:15:53.000Z-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.

```plaintext
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
```
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: <host>
Date: <date>
Authorization: <authorization_string>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Statistics object name. Type: string. Default value: none.</td>
<td>No</td>
</tr>
</tbody>
</table>

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

```
DELETE /?ostor-usage&obj=s3-usage-8000000000000003-2016-04-04T13:12:53.000Z-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
```