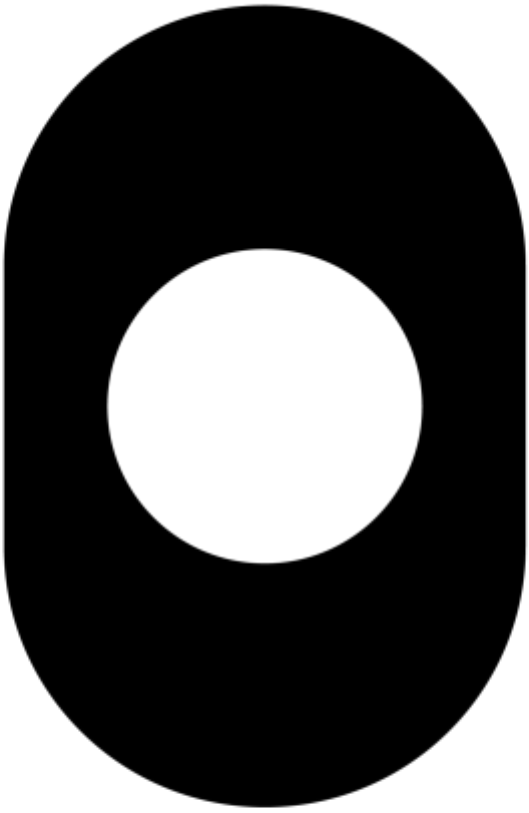

ObjectMatrix

MatrixStore Management REST API - cURL Usage Guide Supplemental

cURL is a free and open-source command-line interface tool that can transfer data via HTTP and other protocols.

As a command-line tool it is useful for creating simple scripts that interact with the REST API. For example, you could create a script that reads a CSV file with usernames and passwords that then creates new user accounts based on those values.

Overview

Be sure to have a copy of the *MatrixStore Management REST API* document to hand and read its Overview chapter before continuing.

cURL - Crash Course

Simple curl commands can look like this:

```
curl -k -b .cookie https://10.0.20.101:8443/mapi/v1/spaces/4b72df98-67de-4f0d-86cc-53840c9b4e7e/vaults
```

A more complex curl command can look like this:

```
curl -k -b .cookie -H "Content-Type: application/json" -i -X POST -d '{"username": "jsmith", "login": {"password": "p455W0rD", "enabled": true}}' https://10.0.20.101:8443/spaces/ab8e92f7-2d9c-40c2-89f6-14a80606f01f/users
```

The last argument is a positional argument, the URL to perform the request on. In between are "optional arguments" that are required to make the request work. e.g. `-H "Content-Type: application/json"` tells the server the type of content in the body so that it can decode it.

Common optional arguments

Option	Description	Default
-k	Allow connections to SSL sites without certificates or with self-signed certs	Only connections with valid certs are allowed
-c	File to write session cookie to	No cookie written to file
-b	Cookie string or file to read from	No cookie sent
-X	HTTP verb	If not provide: GET, unless body is present then POST
-H	Add HTTP header	
-d	HTTP body data. Concatenates if there is more than one occurrence.	No body
-i	Include HTTP headers in the output. Useful if response is `204 No Content`.	Headers not included in output
-o	Output file to write to	Writes to STDOUT
-f	Fail silently on HTTP status 400 or greater and exit with non-zero status (usually 22). Read this for more info.	curl always exits with 0 status code

There are over 100 options in cURL. If you need to get familiar with them then just run:

```
curl --help
```

Authentication

MatrixStore Management REST API currently does not support "stateless" calls but that feature may arrive in the future.

Login - Username and Password

Before you can perform any requests you need to authenticate as your user. If you do not do this, or if your session has expired, you will receive a `401 Unauthorized` HTTP Status for every call.

Example

```
curl -k -c .cookie -i -d username=jsmith -d password=p455W0rD
https://10.0.20.101:8443/mapi/v1/login
```

- `-k` tells curl to skip SSL certificate validation. This is because the REST API uses a self-signed certificate.
- `-c .cookie` tells curl to write the session cookie received in the response to a file called `.cookie` in the current working directory.
- `-i` tells curl to print out the headers in the response. This lets you know what HTTP status the response has. Also since there is no body, omitting this option would yield you no output.
- `-d <content>` concatenates the content provided to the request body. For this request the server expects a url-encoded form (application/x-www-form-urlencoded) which looks will look like this: `username=jsmith&password=p455W0rD`

Output

```
HTTP/1.1 302
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Set-Cookie: JSESSIONID=10127C202E51A9807B191CEF5E87DBAE; Path=/; HttpOnly
Location: https://10.0.20.101:8443/mapi/v1/users/current
Content-Length: 0
Date: Wed, 05 Dec 2018 14:11:41 GMT
```

You can now perform operations via the REST API provided you supply your session cookie in each request.

Login - Access Key

For a script/service, the recommended authentication method is to use an Access Key. Authenticating using an access key in curl is similar to when using a username and password except the field names are different. `accesskeyid` instead of `login` and `accesskeysecret` instead of `password`.

Example

```
curl -k -c .cookie -i -d accesskeyid=bdaa7623-dd7a-4553-8358-fc6bafef3524 -d
accesskeysecret=zLL-7C1IVmNEIecPDQW3OMC-5yWIhOOgk0aFLuiy9lGdIwXZ
https://10.0.20.101:8443/mapi/v1/login
```

Output should be the same as a login when signing-in with username and password.

Logout

After you have finished performing all operations in your script you must logout as the final step.

Example

```
curl -k -i -b .cookie -X POST http://10.0.20.101:8443/mapi/v1/logout
```

- `-b` tells curl to read the session cookie from a file called `.cookie` in the current working directory and provide it in the request. This file must be created during login.
- `-X POST` sets the HTTP verb for this request to POST.

Output

```
HTTP/1.1 204
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Date: Wed, 05 Dec 2018 14:12:51 GMT
```

The session is now invalidated; you should also delete the cookie file.

Performing Operations

Get Current User Info

The request URL for this is actually returned in a login response header with a `302 Found` status, normally indicating that you should fetch this information after login. This is not necessary however.

```
HTTP/1.1 302
...
Location: https://10.0.20.101:8443/mapi/v1/users/current
...
```

With this method you can discover the space id, which is required for requests such as creating or listing users and vaults.

Example

```
curl -k -b .cookie https://10.0.20.101:8443/mapi/v1/users/current
```

Output


```
{
  "id" : "1708ebca-03fd-4834-b3d8-4363f89b4336",
  "spaceId" : "4b72df98-67de-4f0d-86cc-53840c9b4e7e",
  "name" : "John Smith",
  "emailAddress" : "j.smith1970@example.com",
  "description" : null,
  "external" : false
}
```


From this response we can see that we are signed-in as the correct user. We discover the user id (`id`) so we can perform operations on your account if we need to. We also have the space id (`spaceId`) which will enables us to perform operations that need the space id.

Listing Vaults

Using this method you can see what vaults exist in the space, discover their configuration and vault ids. With the vault ids you can perform operations on them such as reconfiguration, downloading audits or purging the trash can.

Example

```
curl -k -b .cookie https://10.0.20.101:8443/mapi/v1/spaces/4b72df98-67de-4f0d-
86cc-53840c9b4e7e/vaults
```

Output


```
[ {
  "id" : "812be8f1-ef0a-11e8-91e6-8d82e58a79a9",
  "name" : "News",
  "usedCapacity" : 0,
  "freeCapacity" : 0,
  "totalCapacity" : 0,
  "numObjects" : 0,
  "config" : {
    "audits" : {
      "read" : false,
      "write" : false,
      "delete" : true
    },
    "capabilities" : {
      "write" : true,
      "read" : true,
      "search" : false,
      "delete" : true,
      "update" : true
    },
    "provisionedCapacity" : 0,
    "protectionScheme" : "Dual",
    "dataUpdatable" : false,
    "contentSearchEnabled" : true,
    "keepTombstones" : false,
    "integrityLevel" : "Medium",
    "compliance" : {
```

```
      "type" : "None",
      "thresholdMins" : 0
    },
    "pip" : {
      "amwa" : false,
      "image" : true,
      "mediaInfo" : false,
      "xmp" : false
    },
    "replication" : {
      "enabled" : false,
      "targetClusterId" : null,
      "targetClusterIPs" : null,
      "targetUserId" : null,
      "targetUserPass" : null,
      "targetVaultId" : null,
      "deleteOnTarget" : false,
      "encryptionEnabled" : false,
      "stubbing" : {
        "enabled" : false,
        "timeout" : null
      }
    },
    "trashCan" : {
      "enabled" : true,
      "thresholdDays" : 10
    }
  }
}, {
  "id" : "87e91bcd-f3c8-11e8-baea-cb4db506dbd9",
  "name" : "Sports",
  ... //More results. Truncated in this doc for readability.
} ]
```

Creating Vaults

Creating vaults through cURL can get complex if you want a detailed configuration that deviates from the default settings. Conversely it can be relatively easy if you have a simple configuration.

Creating anything else, such as Users, is not at all different from what you see in this example. Just use follow the REST API documentation for url paths and json schema.

Example

For this simple example we are just going to enable content search. *As we do not specify a vault admin as a parameter then by default the user making the request becomes the vault admin.*

```
curl -k -b .cookie -X POST -H "Content-Type: application/json" -d '{ "name":
"Productions 2019", "config": { "contentSearchEnabled": true } }'
https://10.0.20.101:8443/mapi/v1/spaces/fd6145ee-ef07-11e8-b54d-
fcc1452a9f53/vaults
```

- `-H "Content-Type: application/json"` tells curl to add this value as a HTTP header in the request. The server will now know how to decode the body data based on this.

- `-d` specifies the request body data which in this case is a partial json configuration for the vault. Optional fields not specified are given default values.

Output


```
{
  "id" : "8c24ea71-f941-11e8-96f7-c62e42d03d08",
  "name" : "Productions 2019",
  "usedCapacity" : 0,
  "freeCapacity" : 7571699500,
  "totalCapacity" : 16966459000,
  "numObjects" : 0,
  "config" : {
    "audits" : {
      "read" : false,
      "write" : false,
      "delete" : true
    },
    "capabilities" : {
      "write" : false,
      "read" : false,
      "search" : false,
      "delete" : false,
      "update" : false
    },
    "provisionedCapacity" : -1,
    "protectionScheme" : "Dual",
    "dataUpdatable" : true,
    "contentSearchEnabled" : true,
    "keepTombstones" : false,
    "integrityLevel" : "Medium",
    "compliance" : {
      "type" : "None",
      "thresholdMins" : 0
    },
    "pip" : {
      "amwa" : false,
      "image" : false,
      "mediaInfo" : false,
      "xmp" : false
    },
    "replication" : {
      "enabled" : false,
      "targetClusterId" : null,
      "targetClusterIPs" : null,
      "targetUserId" : null,
      "targetUserPass" : null,
      "targetVaultId" : null,
      "deleteOnTarget" : false,
      "encryptionEnabled" : false,
      "stubbing" : {
        "enabled" : false,
        "timeout" : null
      }
    },
    "trashCan" : {
      "enabled" : true,
```

```
      "thresholdDays" : 7
    }
  }
}
```

Downloading Vault Audits

With the REST API you can download audits as a zipped CSV file.

If you want to perform some operations on the audit entries inside you only need to first extract the CSV file from the zip file.

Example

We want to get the last 30 days worth of audits for my 'News' vault and store them somewhere for historical reasons. Thankfully there is a pre-defined date range called `last30Days` we can use as a parameter for this.

```
curl -k -b .cookie -o "News audits.zip"
https://10.0.20.101:8443/mapi/v1/vaults/812be8f1-ef0a-11e8-91e6-
8d82e58a79a9/audits/download?range=last30Days
```

- `-o "News audits.zip"` tells curl to write the response body to a file called `News audits.zip` in the current working directory.

Output

There is no response output in the terminal because it is directed to the output file. However you do get download progress feedback.

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  56  1028K  56  576K    0     0  9636k      0  0:00:09  0:00:05  0:00:4 1.01M
```

Partial Response

As detailed in the REST API documentation you can request a partial response which limits the amount of data returned to only the fields you requested.

For instance, say you are only interested in the id, name, and trash can configuration of vaults in a space, this is the URL path you would use:

```
/mapi/v1/spaces/145ee-ef07-11e8-b54d-fcc1452a9f53/vaults?
fields=id,name,config[trashCan]
```

Note the `fields` query parameter and its values.

Using cURL however you need to url escape the square brackets `[` `]` as `%5B` and `%5D` respectively. This is for 3 reasons:

1. Square brackets in cURL are used for URL globbing.

2. In bash, square brackets are reserved characters used for conditional statements.
3. As stated in RFC 1738, square brackets are considered unsafe characters and must be url encoded.

> These characters are "{", "}", "|", "\", "^", "~", "[", "]", and "`".
> All unsafe characters must always be encoded within a URL.

Example

```
curl -k -b .cookie https://10.0.20.101:8443/mapi/v1/spaces/145ee-ef07-11e8-b54d-fcc1452a9f53/vaults?fields=id,name,config%5BtrashCan%5D
```

Output

```
[ {
  "id" : "812be8f1-ef0a-11e8-91e6-8d82e58a79a9",
  "name" : "News",
  "config" : {
    "trashCan" : {
      "enabled" : true,
      "thresholdDays" : 10
    }
  }
}, {
  "id" : "87e91bcd-f3c8-11e8-baea-cb4db506dbd9",
  "name" : "Sports",
  "config" : {
    "trashCan" : {
      "enabled" : true,
      "thresholdDays" : 7
    }
  }
}, {
  "id" : "f3b2d4d4-f6ef-11e8-b7e0-ec15753f2f22",
  "name" : "Drama",
  "config" : {
    "trashCan" : {
      "enabled" : false,
      "thresholdDays" : 7
    }
  }
} ]
```

Bash Script Example

Provided here is an example bash script you could use as a template and expand upon.

If you are using this as a template you need to modify the variables at the start of the script so that they are valid for your own setup and then write your own instructions after `## REAL SCRIPT STARTS HERE ##`.

```
#!/bin/bash
```

```
################
## An example bash script using curl for with MatrixStore Management REST API.
## In this script we simply silence the raid alarm on nodes we specified.
################

## INIT VARIABLES (modify these to your needs) ##
MAPI_URL="https://10.0.20.101:8443/mapi/v1"
ACCKEY_ID="bdaa7623-dd7a-4553-8358-fc6bafef3524"
ACCKEY_SCRT="zLL-7C1IVmNEIecPDQW3OMC-5yWIhOOgk0aFLuiy9lGdIwXZ"
COOKIE=".cookie"

## SETUP ##

# Make script exit if a subsequent command ends with non-zero status or when an
undeclared variable is referenced, and make sure a exit status is propagated if
script is called as part of a pipeline.
set -euo pipefail

# Function to perform login. Always call this as the first operation.
fLogin() {
  echo "Logging into MAPI with access key: $ACCKEY_ID"
  curl -f -k -c "$COOKIE" -d "accesskeyid=$ACCKEY_ID" -d
"accesskeysecret=$ACCKEY_SCRT" "$MAPI_URL/login"
}

# Function to perform logout and delete the cookie file. Always call this before
exiting.
fLogout() {
  echo "Logging out of MAPI"
  curl -k -b "$COOKIE" -X POST "$MAPI_URL/logout"
  rm "$COOKIE"
}

# Login always necessary before you can perform any operations.
fLogin

# Setup a bash trap so that logout is always called before the script exits,
especially on failures.
trap fLogout EXIT

## REAL SCRIPT STARTS HERE ##

# Function that silences the RIAD alarm on a node with the given node id.
# $1 is the node id
fSilenceAlarm() {
  echo "Silencing RAID alarm of node: $1"
  curl -f -k -b "$COOKIE" -X POST "$MAPI_URL/nodes/$1/silence-alarm"
}

# Silence all three nodes in the cluster
fSilenceAlarm "74038e51-df58-2cd2-8d10-3dc6700a2306"
fSilenceAlarm "b4606cf8-69c8-4da5-aa39-41d65254ba4d"
fSilenceAlarm "edfb3e4d-5984-4315-aca9-49079a85bdcf"
```