CloudBackend

CLI user guide

version 1.4

2021-06-24

Document objective

This user guide will give an overview on how to use the client CLI (Command Line Interface) for CloudBackend (CBE) distributed database Platform as a Service. This is done by listing the commands and added practical examples. The CLI is provided as a tool to give a quick way to access the CBE service without the need for first writing own client code and to verify your own code during development and testing. It can also be used to verify the data sent to the backend service.

Conventions

Describing text is printed in font Calibri.

Examples from running on a computer are shown in a box.

```
Terminal output that can be expected in a computer teletype window is printed in font
Courier New.

Terminal output that is kept secret is printed here as ***

Terminal input is printed in font Lucida Console.
```

Preparation

For this you need the following installed:

- Ubuntu 18.04.5 LTS Linux operating system
  or
- Ubuntu 20.04.2 LTS Linux operating system

To install the CloudBackend, see the Tutorial installation section.

Installation

From a terminal window; set execute permission on the file CLI.

```
cd cbe              # replace with your project directory
cd 1.4*             # the name previously given to the directory of the latest release
cd CLI              # CLI directory
chmod u+x cbe_CLI   # set execute permission
```

Start

Login with your user credentials.

```
./cbe_CLI -?   # see program options

./cbe_CLI  # start the program
Login to your CloudBackend tenant user account: Type in username and password.
Username:  username1
Password:  ***
Tenant:  ABC
User: username1 - login complete - RootContainer Id: ***
Input the command first and then any arguments; help will list them.

CBE>
```

This indicates that the login is complete and the user's root container is available.

In the CLI context the root container is referred to as /

Command overview: help

In the CLI environment; list the commands available.

```
CBE>  help

help container          # help on container commands
help object             # help on object commands
help query              # help on query commands
help all                # full help
?                       # same as help

createUser
getUserInfo
cd [container]          # set default container
-recent                 # types most recent CLI command
-version                # version of CLI + compile time.
!!                      # repeats most recent CLI command
! ls                    # execute the command following the ! in the o/s shell
exit
```

```
CBE> help all

createContainer [container] [name]
renameContainer [container] [new name]
moveContainer [container] [destination container]
removeContainer [container]
getContainerACL [container]
setContainerACL [container] [number of user permissions to set]
shareContainer [container] [toUserId] [Description/Name of the share]
unshareContainer [container] [yes/no] [shareId / userId to unShare]
set [Container]

createObject [Container] [name of new object] [number of Key/Value pairs]
updateKeyValues [Container] [Object] [number of Key/Value pairs]
upload [container] [local path] [filename]
download [Object path] [local path]
type [Object path of text]
renameObject [object] [new name]
moveObject [object] [destination container]
removeObject [Container] [Object to remove]
shareObject [Object] [toUserId] [Description/Name of the share]
unshareObject [Object] [yes/no] [shareId / userId to unShare]

query [/container-path]
loadGroups
queryGroup [group:/path]
queryCID [container ID]
queryKey [Key:Value, Key:*]
queryFilter [filter statement]
loadMeta [Object path]
waypoints [Object1 path] [Object2 path]
```

Command: createContainer

Create a new container in your backend.

```
CBE>  createContainer
Path to container to create new container in:  /
Name for new container:  test1
Container added, Container name: test1 Container created: 0
```

Alternatively, give the command and arguments in the same line.

```
CBE>  createContainer / test2
Container added, Container name: test2 Container created: 0
```

Command: getUserInfo

List the userid and name

```
CBE>  getUserInfo
User id: ***
First name: Demo, Last name: Person.
root container id: ***
```

Command: createObject

Create a new object in your backend.

```
CBE> createObject
Create object in container: /test1
Name for the Object: fooA
Set the number of Key/Value pairs you want: 2
Input Key nr 1 : ida
Input Value for tag nr 1 : 11
Input if KeyValue pair 1 is indexed or not (1 indexed, 0 not indexed): 1
Input Key nr 2 : name
Input Value for tag nr 2 : Adam
Input if KeyValue pair 2 is indexed or not (1 indexed, 0 not indexed): 1

Object created: fooA
Object KeyValues:
ida (indexed) = '11'
name (indexed) = 'Adam'
```

Alternatively, give the command and arguments in the same line.

```
CBE> createObject /test2 fooB 2
Input Key nr 1 : ida
Input Value for tag nr 1 : 21
Input if KeyValue pair 1 is indexed or not (1 indexed, 0 not indexed): 1
Input Key nr 2 : name
Input Value for tag nr 2 : Bob
Input if KeyValue pair 2 is indexed or not (1 indexed, 0 not indexed): 1

Object created: fooB
Object KeyValues:
ida (indexed) = '21'
name (indexed) = 'Bob'
```

Command: query

Ask for the content of a container.

```
CBE>  query
Path to container to query:  /
Query done For RootContainer:
Query items:
test1                Container id: ***
test2                Container id: ***
=====
Count: 2 containers + 0 object = 2 items
```

Another example.

```
CBE>  query
Path to container to query:  /test1
Query done For Container: test1
Query items:
fooA                 Object   id: ***
=====
Count: 1 container + 0 object = 1 item
```

Alternatively, give the command and arguments in the same line.

```
CBE>  query /
Query done For RootContainer:
Query items:
test1                Container id: ***
test2                Container id: ***
=====
Count: 2 containers + 0 object = 2 items
```

Another example.

```
CBE>  query /test2
Query done For Container:
test2
Query items:
fooB                 Object   id: ***
=====
Count: 0 container + 1 object = 1 item
```

queryGroup does the same thing and can also work on group containers. loadGroups is a pre-requisite to it.

queryCID takes a container id and does query on that.

Command: queryKey

Ask for object with specific key tag name or key tag value.

```
CBE>  queryKey
string of tags to search for:  ida
Query done for Keys: ida
Query Objects:
fooA                 Object   id: ***
fooB                 Object   id: ***
=====
Count: 0 container + 2 object = 2 item
```

Another example.

```
CBE>  queryKey
string of tags to search for:  name:Bob
Query done for Keys: name:Bob
Query Objects:
fooB                 Object   id: ***
=====
Count: 0 container + 1 object = 1 item
```

Alternatively, give the command and arguments in the same line.

```
CBE>  queryKey name:Adam
Query done for Keys: name:Adam
Query Objects:
fooA                 Object   id: ***
=====
Count: 0 container + 1 object = 1 item
```

Another example.

```
CBE>  queryKey ida:21
Query done for Keys: ida:21
Query Objects:
fooB                 Object   id: ***
=====
Count: 0 container + 1 object = 1 item
```

Command: loadMeta

List the key tag meta data of one object.

```
CBE>  loadMeta
Path to Object:  /test1/fooA

Object Name: fooA
Container: /test1
Created: ***
Updated: ***
Total size: 0 kB

Key/Value data:
ida (indexed) = '11'
name (indexed) = 'Adam'
```

Alternatively, give the command and arguments in the same line.

```
CBE>  loadMeta /test2/fooB

Object Name: fooB
Container: /test2
Created: ***
Updated: ***
Total size: 0 kB

Key/Value data:
ida (indexed) = '21'
name (indexed) = 'Bob'
```

Command: updateKeyValues

Change the Key Values of an object.

```
CBE> updateKeyValues
Path of Container: /test1
Name of the Object: fooA
Set the number of Key/Value pairs you want to add/update: 3
Input key nr 1 : ida
Input value for key nr 1 'ida' : 111
Input if KeyValue pair 1 is indexed or not (1 indexed, 0 not indexed): 1
Input key nr 2 : name
Input value for key nr 2 'name' : Anna
Input if KeyValue pair 2 is indexed or not (1 indexed, 0 not indexed): 1
Input key nr 3 : age
Input value for key nr 3 'age' : 33
Input if KeyValue pair 3 is indexed or not (1 indexed, 0 not indexed): 0
Key/Value pairs updated/added for object: fooA
New KeyValue pairs:
ida = '111' (indexed)
age = '33'
name = 'Anna' (indexed)
```

Alternatively, give the command and arguments in the same line.

```
CBE> updateKeyValues /test2 fooB 3
Input key nr 1 : ida
Input value for key nr 1 'ida' : 211
Input if KeyValue pair 1 is indexed or not (1 indexed, 0 not indexed): 1
Input key nr 2 : name
Input value for key nr 2 'name' : Bambi
Input if KeyValue pair 2 is indexed or not (1 indexed, 0 not indexed): 1
Input key nr 3 : age
Input value for key nr 3 'age' : 22
Input if KeyValue pair 3 is indexed or not (1 indexed, 0 not indexed): 0
Key/Value pairs updated/added for object: fooB
New KeyValue pairs:
ida = '211' (indexed)
age = '22'
name = 'Bambi' (indexed)
```

Command: upload

Upload an object from the local computer to the backend.

```
CBE>  upload
Path to container to upload to:  /test1
Path to file on local filesystem:  /home/demo/
Filename on local filesystem:  car.jpg
Object uploaded!
Object name: car.jpg Object id: *** object parent id (uploaded to container): ***
```

Alternatively, give the command and arguments in the same line.

```
CBE>  upload /test2 /home/demo/ plat.mp3
Object uploaded!
Object name: mine.txt Object id: *** object parent id (uploaded to container): ***
```

Verify the uploads.

```
CBE>  query /test1
Query Loaded!
Query done For Container: test1
Query items:
Name: car.jpg              (Object)    Id: ***           Parent Id: ***
Name: fooA                 (Object)    Id: ***           Parent Id: ***
Total Result(s):2

CBE>  query /test2
Query Loaded!
Query done For Container: test2
Query items:
Name: plat.mp3             (Object)    Id: ***           Parent Id: ***
Name: fooB                 (Object)    Id: ***           Parent Id: ***
Total Result(s):2
```

Command: renameContainer

Change the name of a container.

```
CBE>  renameContainer
Container to rename:  /test1
New name:  x1
Container id: *** has been renamed to: x1
```

Alternatively, give the command and arguments in the same line.

```
CBE>  renameContainer /test2 x2
Container id: *** has been renamed to: x2
```

Command: moveContainer

In this example first a new container and object are created, and then the name of the object is changed.

```
CBE>  query /
CBE>  createContainer / test3
CBE>  createContainer / test4
```

Verify the content. Move a container into another container.

```
CBE>  query /
CBE>  moveContainer
Move Container:  /x1
to Container:  /test3
Container x1 has been moved to container with id: ***
```

Alternatively, give the command and arguments in the same line.

```
CBE>  query /
CBE>  moveContainer /x2 /test4
Container x2 has been moved to container with id: ***
```

Command: removeContainer

Remove a container.

Please Note! All content of that container will be removed, including objects and subcontainers *!*

```
CBE>  query /
CBE>  removeContainer
Path to container to remove:  /test3
Container x1 with ID: *** has been removed
```

Alternatively, give the command and arguments in the same line.

```
CBE>  query /
CBE>  removeContainer /test4
Container x2 with ID: *** has been removed

CBE>  query /
```

Please Note! All content of that container will be removed, including objects and subcontainers *!*

Command: setContainerACL

List the ACL (Access Control List) of a container. The following permission values can be specified:

```
 0 = no permissions
 1 = read
 2 = write
 3 = read/write
 4 = delete
 5 = delete/read
 6 = delete/write
 7 = read/write/delete
 8 = setACL
 9 = setACL/read
10 = setACL/write
11 = setACL/read/write
12 = setACL/delete
13 = setACL/delete/read
14 = setACL/delete/write
15 = setACL/read/write/delete
```

```
CBE> createContainer / test5
CBE> setContainerACL
Container: /test5
Container to setACL for: /test5
Number of user ids and permissions to set: 1
Input user id: 1 : ***
Input permission for user id 1 '***' : 3
Container ACL:s have been added:
UserId: *** Permission/ACL: 3
```

Alternatively, give the command and arguments in the same line.

```
CBE> setContainerACL /test5 1
Input user id: 1 : ***
Input permission for user id 1 '***' : 7
Container ACL:s have been added:
UserId: *** Permission/ACL: 7
```

Command: getContainerACL

List the Access Control List of a container.

```
CBE>  getContainerACL
Container:  /test5
Container ACL:s have been Loaded:
UserId: *** Permission/ACL: 7
UserId: *** Permission/ACL: 15
```

Alternatively, give the command and argument in the same line.

```
CBE>  getContainerACL /test5
Container ACL:s have been Loaded:
UserId: *** Permission/ACL: 7
UserId: *** Permission/ACL: 15
```

Command: shareContainer

Share a container with another user. The ACL of the container should have been set using **setContainerACL** with access rights for that particular user.

```
CBE>  shareContainer
Share Container:  /test5
To user Id: ***
Description/name of the share:  Test
Container has been shared! ShareId: ***
```

Alternatively, give the command and arguments in the same line.

```
CBE>  shareContainer /test5
Container has been shared! ShareId: ***
```

Command: unshareContainer

Undo a sharing of a container with another user.

```
CBE>  unshareContainer
Container to unShare: /test5
Do you know the shareId? (yes/no)  no
Which user (id) do you want to remove the share for: ***
Container has been un shared, message from edge platform: OK
```

Alternatively, give the command and arguments in the same line.

```
CBE>   unshareContainer /test5 no
Which user (id) do you want to remove the share for: ***
Container has been un shared, message from edge platform: OK
```

Command: renameObject

Change the name of an Object.

In this example first a new container and object are created, and then the name of the object is changed.

```
CBE>  query /
CBE>  createObject /test5 fooX 0
CBE>  query /test5

CBE>  renameObject
Full path to Object:  /test5/fooX
New name: fooE
Object with id: *** has been renamed to: fooE

CBE>  query /test5
```

Command: moveObject

Move an Object between containers.

In this example first an object is created, and then the object is moved to another container.

```
CBE>  query /test5
CBE>  createObject /test5 A1 0
CBE>  createObject /test5 A2 0
CBE>  createContainer / test6
CBE>  query /test6
CBE>  query /test5

CBE>  moveObject
Move Object:  /test5/A1
to Container:  /test6
Object A1 has been moved to container with id: ***
```

Alternatively, give the command and arguments in the same line.

```
CBE>  moveObject /test5/A2 /test6
Object A2 has been moved to container with id: ***

CBE>  query /test5
CBE>  query /test6
```

Command: removeObject

Delete an Object.

```
CBE>  removeObject
Container Path:  /test6
Object to remove, name:  A1
Object fooL with id: *** has been removed
```

Alternatively, give the command and arguments in the same line.

```
CBE>  removeObject /test6 A2
Object fooM with id: *** has been removed

CBE>  query /test6
```

Command: shareObject

Share an object with another user. The ACL of the container holding the object should have been set using **setContainerACL** with access rights for that particular user. When seting the ACL of a container, all objects in it will inherit the same ACL.

```
CBE>  createObject /test5 fooE 0
CBE>  setContainerACL /test5 1

CBE>  shareObject
Share Object, full path:  /test5/fooE
To user Id: ***
Description/name of the share:  Testobject
Object has been shared! ShareId: ***
```

Alternatively, give the command and arguments in the same line.

```
CBE>  shareObject /test5/fooE

Object has been shared! ShareId: ***
```

Command: unshareObject

Undo a sharing of an object with another user.

```
CBE>  unShareObject
Object to unShare: /test5
Do you know the shareId? (yes/no)  yes
ShareId:  ***
Object has been un shared, message from edge platform: OK
```

Alternatively, give the command and arguments in the same line.

```
CBE>  unShareObject /test5/fooE

Object has been un shared, message from edge platform: OK
```

Command: loadGroups

Load the Groups the user belongs to. This is a pre-requisite for the queryGroups command.

```
CBE>  loadGroups
  #             group           database        container-id
----    ---------------         --------        ---------------
  1:    demotesters:/               home        ***

CBE>  queryGroup demotesters:/
Items
-----
demotester2                 Object    id:***
=====
Count: 0 container + 1 object = 1 item
```

queryGroup does the same thing as query but can also work on group containers.

queryCID takes any container id you have access to and does query on that.

Command: cd

Set the container context.

This saves specifying the container when many commands will be performed in the same container.

```
CBE>  cd /
CBE(/)> query
Items
-----
test5                           Container id: ***
test6                           Container id: ***
=====
Count: 2 containers + 0 object = 2 item
```

Another example.

```
CBE>(/)>  cd /test5
CBE(/test5)>  query
Items
-----
fooE                            Object    id: ***
=====
Count: 0 container + 1 object = 1 item
```

Set context to a group container.

```
CBE(/)>  cd githubtesters:/Profiles
CBE(githubtesters:/Profiles)>
 queryGroup
Items
-----
githubtester2                  Object    id:***
=====
Count: 0 container + 1 object = 1 item
```

Command: -version

Print the version information for this CLI program.

```
CBE>  -version
CloudBackend CLI Version *
```

Command: exit

This is the exit door to leave the CLI.

```
CBE(/test5)>  exit
Exiting CLI
Goodbye.
```