Working with Clinical Data

Table of Contents:

Prerequisites

A working setup of openCGA is required to setup a Testing environment. If user hasn't yet set it up, please follow the steps on installation guide and set it up.

Initialisation

All of the following steps assume, user is under openCGA installation directory (*/opt/opencga/*).

The following CLI command will create the database, the collections and all the indexes, it also creates the admin user with the specified password. The MongoDB database *host* and *name* are read from the */conf/catalog-configuration.yml* file by default.

Install Catalog v1.4.x

```
./opencga-admin.sh catalog install --secret-key any_string_you_want <<<
admin_P@ssword
```

Install Catalog v1.3.x

```
./opencga-admin.sh catalog install --algorithm HS256 --secret-key
any_string_you_want -p <<< admin_P@ssword
```

This following command will create a user name "John Doe" and ID "test". Note that as by default OpenCGA is configured as ***private,*** only the OpenCGA admin user can create other users. We are using opencga-admin CLI.

Create User

```
./opencga-admin.sh users create -p -u test --email test@gel.ac.uk --name
"John Doe" --user-password user_P@ssword <<< admin_P@ssword
```

Now we will use this newly created user "test" for further actions. First, we will need to authenticate as that user:

Login

```
./opencga.sh users login -u test -p <<< user_P@ssword
```

This will create a hidden directory in your home called .opencga. This directory will contain a file named *~/.opencga/session.json* with the user id and the valid token. This will be used automatically by opencga.sh and will only be valid for some minutes, by doing this users do not have to write the password too many times. The contents of session.json file will look like :

session.json

```
{
  "userId" : "test",
  "token" : "eyJhbGciOiJIUzI1NiJ9.
eyJzdWIiOiJ0ZXN0IiwiYXVkIjoiT3BlbkNHQSB1c2VycyIsImlhdCI6MTUxMzAwOTQ2MiwiZXh
wIjoxNTEzMDEzMDYyfQ.PAgz4hus2oL4wRCxt2JJ54_jR-efjiERRrFgg49Pkrs",
  "login" : "2017-12-11T15:31:59.616",
  "logout" : null,
  "projectsAndStudies" : { }
```

Now with new user, we create a project name "*Reference studies GRCh37*" and alias "*reference_grch37*" with the following command :

Create Project

```
./opencga.sh projects create -a reference_grch37 -n "Reference studies
GRCh37" --organism-scientific-name "Homo sapiens" --organism-assembly
"GRCh37"
```

Note: organism-scientific-name and organism-assembly should be available in CellBase. User can get this information using the following public WS: http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v4/meta/species

Next step, create a study name "" inside project "reference_grch37"

Create Study

```
./opencga.sh studies create -a 1kG_phase3 -n "1000 Genomes Project - Phase
3" --project reference_grch37
```

Now, we will create a dummy sample that we'll use to show how to add annotations to.

Create sample

```
./opencga.sh samples create -n sample1 -s 1kG_phase3
```

Now, we will create a dummy individual that will also contain some annotations:

Create individual

```
./opencga.sh individuals create -n individual1 -s 1kG_phase3
```

Ingesting Clinical Data (creating Variable Sets and Annotation Sets)

We are going to use the *Variable Sets* and *Annotation Sets* used in the examples of the Annotation and Clinical Data section. Here are the files needed to load those *Variable Sets* and *Annotation Sets* using the command line: demo.tar.gz

First, we will need to load both *Variable Sets.* To do so, we will run the following command lines:

```
./opencga.sh variables create --json demo/individual_vs.json -n
individual_private_details --confidential --description "Private details
of the individual" -s 1kG_phase3 --of yaml
./opencga.sh variables create --json demo/sample_vs.json -n
sample_metadata --description "Sample origin" -s 1kG_phase3 --of yaml
```

From that moment on, we can *annotate* using any of the *Variable Sets* any of the *Annotable* entries. For example, to annotate both the sample and the individual we created we will run the following commands:

```
# Annotate the sample sample1 using the variable set 'sample_metadata'
./opencga.sh samples annotation-sets-create --annotation-set-name
sampleAnnotName --annotations demo/sample_as.json --id sample1 --variable-
set-id sample_metadata


# Annotate the individual individual1 using the variable set
'individual_private_details'
./opencga.sh individuals annotation-sets-create --annotation-set-name
individualAnnotName --annotations demo/individual_as.json --id individual1
--variable-set-id individual_private_details
```

Querying Clinical Data

Querying individuals

```
# Querying all individuals annotated with gender = MALE. Result: The only
individual we have created
./opencga.sh individuals search --annotation gender=MALE --variable-set
individual_private_details

# Querying all individuals annotated with age < 60. Result: None because
the individual we annotated has age = 60
./opencga.sh individuals search --annotation "age<60" --variable-set
individual_private_details

# But we can obtain it if we change the query to age <= 60 as follows
./opencga.sh individuals search --annotation "age<=60" --variable-set
individual_private_details

# Querying all individuals with age <= 60 and gender = FEMALE. No results
because our individual is a MALE.
./opencga.sh individuals search --annotation "age<=60;gender=FEMALE" --
variable-set individual_private_details

# Now we change the query to age <=60 and gender = MALE. We get again the
individual we expected.
./opencga.sh individuals search --annotation "age<=60;gender=MALE" --
variable-set individual_private_details
```

Querying samples

```
# Querying all samples annotated with tissue = "umbilical cord blood".
Result: The only sample we have created
./opencga.sh samples search --annotation tissue="umbilical cord blood" --
variable-set sample_metadata


# Querying all samples annotated with tissue = "umbilical cord blood" and
cell type = "multipotent progenitor"
./opencga.sh samples search --annotation "tissue=umbilical cord blood;
cell_type=multipotent progenitor" --variable-set sample_metadata
```