

Using the Java REST client

The Java client provides an API to the whole [OpenCGA RESTful layer](#). We will here only focus on **those methods which are of most interest for HGVA users**. In order to understand how to create queries using these methods, it would be interesting to have a look at the [Datasets and Studies](#) section before.

Getting the Java client code

As previously said, the Java client code is distributed together with the rest of the [OpenCGA code](#).

The OpenCGA code can be cloned in your machine by executing in your terminal:

```
$ git clone https://github.com/opencb/opencga.git
$ cd opencga
$ git checkout release-1.1.0
```

Alternatively, you can download *tar.gz* files with the code for the latest tags/releases of OpenCGA from:

<https://github.com/opencb/opencga/releases>

Once you have downloaded the code, follow the instructions at the *How to Build* section of the OpenCGA repository:

<https://github.com/opencb/opencga>

That will generate the *.jar* containing the Java client library. If you are using *Maven* as a build and dependency manager you shall find the client *.jar* file at:

```
$ ll opencga/build/libs/opencga-client-1.1.0.jar
-rw-r--r-- 1 user user 40K Jan  4 17:34 opencga/build/libs/opencga-client-1.1.0.jar
$ ll ~/.m2/repository/org/opencb/opencga/opencga-client/1.1.0/opencga-client-1.1.0.jar
-rw-r--r-- 1 user user 40K Jan  4 17:34 /home/user/.m2/repository/org/opencb/opencga/opencga-client/1.1.0/opencga-client-1.1.0.jar
```

Initializing the Java client

The [OpenCGAClient constructor](#) requires a [ClientConfiguration](#) object to be passed as a parameter. This *ClientConfiguration* object will contain basic connection details, namely the URL that points to HGVA web services. The best way to obtain a *ClientConfiguration* object is to create a *.yml* configuration file that will later be passed to the *load* static method of the *ClientConfiguration* class to generate a new *ClientConfiguration* object. A *client-configuration.yml* template is provided within the OpenCGA code. If you have cloned the OpenCGA code, you will find the *client-configuration.yml* file at:

```
$ ll opencga/opencga-client/src/main/resources/client-configuration.yml
-rw-r--r-- 1 user user 272 Sep  8 14:10 opencga/opencga-client/src/main/resources/client-configuration.yml
```

This file can easily be edited to set the *resthost* attribute to the HGVA web services URL (<http://bioinfo.hpc.cam.ac.uk/hgva/webservices/>):

Table of Contents:

- [Getting the Java client code](#)
- [Initializing the Java client](#)
 - [Getting information about genomic variants](#)
 - [Getting information about projects](#)
 - [Getting information about studies](#)
 - [Getting information about samples](#)
 - [Getting information about cohorts](#)

client-configuration.yml

```
---
## number of seconds that session remain open
sessionDuration: 12000

## REST client configuration options
rest:
  host: "http://bioinfo.hpc.cam.ac.uk/hgva"
  batchQuerySize: 200
  timeout: 10000
  defaultLimit: 2000

## gRPC configuration options
grpc:
  host: "localhost:9091"
```

Once the *client-configuration.yml* file is ready, you can just create an *OpenCGAClient* object by running:

```
import org.opencb.opencga.client.rest.OpenCGAClient;
import org.opencb.opencga.client.config.ClientConfiguration;
...
...
...
OpenCGAClient openCGAClient;
ClientConfiguration clientConfiguration;

// Load client configuration from client-configuration.yml file
clientConfiguration = ClientConfiguration.load(new FileInputStream(Paths.
get("/path/to/client-configuration.yml")));
// Create OpenCGA client
openCGAClient = new OpenCGAClient(clientConfiguration);
```

The *OpenCGAClient* will, in turn, be able to generate different data client types that will provide methods for accessing the different data types. The most relevant data client types for HGVA users will be the *VariantClient*, *ProjectClient*, *StudyClient*, *CohortClient* and *SampleClient*, that you can create by simply doing:

```
VariantClient variantClient = openCGAClient.getVariantClient();
ProjectClient projectClient = openCGAClient.getProjectClient();
StudyClient studyClient = openCGAClient.getStudyClient();
SampleClient sampleClient = openCGAClient.getSampleClient();
```

Through these clients you will be able to access information about variants, projects, studies and samples. Please, have a look at the examples provided below.

Getting information about genomic variants

Getting variant data from a given study. You can use the *query* method within the *VariantClient* class:

```
public class VariantClient extends AbstractParentClient {
...
    public QueryResponse<Variant> query(ObjectMap params) throws
    CatalogException, IOException
...
}
```

The *params* parameter can be provided as a *QueryOptions* object, which works as a *Map* by providing a *put* method that enables to add pairs (filter, value) that form the actual query. Available filters and possible values for them are those described at the [Swagger specification](#) for the corresponding web service. For example, get TTN variants from the Genome of the Netherlands study, which is framed within the *reference_grch37* project. We will also limit the number of returned results to 3:

```
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
QueryOptions queryOptions = new QueryOptions();
queryOptions.put("gene", "TTN");
queryOptions.put("studies", "GONL");
queryOptions.put("limit", 3);
openCGAClient.getVariantClient().query(queryOptions);
```

Getting information about projects

Getting all metadata from a particular project. You can use the *get* method that the *ProjectClient* class inherits from the *CatalogClient* :

```
public abstract class CatalogClient<T, A> extends AbstractParentClient {
...
    public QueryResponse<T> get(String id, QueryOptions options) throws
IOException
...
}
```

Inputs:

- id: String containing the project alias or project name. You can get available project alias at the [Datasets and Studies](#) section.
- options: must be set to null in this case, since no filtering options are available for this purpose.

For example, getting all metadata for the *reference_grch37* project:

```
openCGAClient.getProjectClient().get("reference_grch37", null);
```

Getting all metadata from all studies associated to a particular project. You can use the *getStudies* method of the *ProjectClient*:

```
public class ProjectClient extends CatalogClient<Project, Project> {
...
    public QueryResponse<Study> getStudies(String projectId, QueryOptions
options) throws CatalogException, IOException
...
}
```

Inputs:

- projectId: String containing the project alias or project name. You can get available project alias at the [Datasets and Studies](#) section.
- options: QueryOptions object which will contain the pairs (filter, value) that form the query. QueryOptions objects work as a Map object, by providing a *put* method that enables to add the (filter, value) pairs. Available filters and possible values for them are those described at the [Swagger specification](#) for the corresponding web service.

For example, getting all studies and their metadata for the *cancer_grch37* project:

```
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
QueryOptions queryOptions = new QueryOptions();
openCGAClient.getProjectClient().getStudies("cancer_grch37", queryOptions);
```

Getting information about studies

Get all available studies and their metadata. Please note, of special interest will be here the field *alias* which contains the study identifier to be used as an input whenever a study must be passed as a parameter. You can use the *search* method of the *StudyClient* class:

```
public class StudyClient extends CatalogClient<Study, StudyAclEntry> {
    ...
    public QueryResponse<Study> search(Query query, QueryOptions options)
    throws IOException
    ...
}
```

Inputs:

- query: Query object which will contain the pairs (filter, value) that form the query. Query objects work as a Map object, by providing a *put* method that enables to add the (filter, value) pairs. Available filters and possible values for them are those described at the [Swagger specification](#) for the corresponding web service.
- options: not necessary, can be set to an empty *QueryOptions*.

For example, getting all metadata for all available studies:

```
import org.opencb.commons.datastore.core.Query;
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
Query query = new Query();
QueryOptions queryOptions = new QueryOptions();
openCGAClient.getStudyClient().search(query, queryOptions);
```

Getting summary data from a particular study. You can use the *getSummary* method of the *StudyClient* class:

```
public class StudyClient extends CatalogClient<Study, StudyAclEntry> {
    ...
    public QueryResponse<StudySummary> getSummary(String studyId,
    QueryOptions options) throws CatalogException, IOException
    ...
}
```

Inputs:

- studyId: String containing the study alias or study name. You can get available study aliases /names by using the method above.
- options: *QueryOptions* object which will contain the pairs (filter, value) that form the query. *QueryOptions* objects work as a Map object, by providing a *put* method that enables to add the (filter, value) pairs. Available filters and possible values for them are those described at the [Swagger specification](#) for the corresponding web service.

For example, getting summary data for study *1kG_phase3* which is framed within project *reference_grch37*:

```
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
QueryOptions queryOptions = new QueryOptions();
openCGAClient.getStudyClient().getSummary("reference_grch37:1kG_phase3",
queryOptions);
```

Getting all available metadata for a particular study. You can use the *get* method that the class *StudyClient* inherits from the *CatalogClient* class:

```
public abstract class CatalogClient<T, A> extends AbstractParentClient {
    ...
    public QueryResponse<T> get(String id, QueryOptions options) throws
    IOException
    ...
}
```

Inputs:

- id: String containing the study alias or study name. You can get available study aliases calling the *search* method of the *StudyClient*.
- options: must be set to null in this case, since no filtering options are available for this purpose.

For example, getting all metadata for study *GONL* which is framed within the project *reference_grch37*:

```
openCGAClient.getStudyClient().get("GONL", null);
```

Getting all samples metadata for a given study. You can use the method *getSamples* of the *StudyClient* class:

```
public class StudyClient extends CatalogClient<Study, StudyAclEntry> {
    ...
    public QueryResponse<Sample> getSamples(String studyId, QueryOptions
    options) throws CatalogException, IOException
    ...
}
```

Inputs:

- studyId: String containing the study alias or study name. You can get available study aliases /names by using the method above.
- options: QueryOptions object which will contain the pairs (filter, value) that form the query. QueryOptions objects work as a Map object, by providing a *put* method that enables to add the (filter, value) pairs. Available filters and possible values for them are those described at the [Swa gger specification](#) for the corresponding web service.

For example, getting all samples metadata for study *1kG_phase3* which is framed within project *reference_grch37*. Please, note that not all studies contain samples data, e.g. *GONL*, *ExAC*, among others, only provide variant lists and aggregated frequencies, i.e. no sample genotypes.

```
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
QueryOptions queryOptions = new QueryOptions();
openCGAClient.getStudyClient().getSummary("reference_grch37:1kG_phase3",
queryOptions);
```

Getting information about samples

Get all metadata for a particular sample. You can use the *get* method that the class *SampleClient* inherits from *CatalogClient* class:

```
public abstract class CatalogClient<T, A> extends AbstractParentClient {
    ...
    public QueryResponse<T> get(String id, QueryOptions options) throws
    IOException
    ...
}
```

Inputs:

- id: String containing the sample name. You can get available samples by using some of the methods described above.
- options: must be set to null in this case, since no filtering options are available for this purpose.

For example, get all metadata for sample *HG00096* of the *1kG_phase3* study which is framed within the *reference_grch37* project:

```
openCGAClient.getSampleClient().get("HG00096", null);
```

Getting information about cohorts

Getting all samples metadata in a given cohort. You can use the *getSamples* method of the *CohortClient*:

```
public class CohortClient extends AnnotationClient<Cohort, CohortAcEntry>
{
    ...
    public QueryResponse<Sample> getSamples(String cohortId, QueryOptions
options) throws CatalogException, IOException
    ...
}
```

Inputs:

- *cohortId*: String containing the cohort id.
- *options*: *QueryOptions* object which will contain the pairs (filter, value) that form the query. *QueryOptions* objects work as a Map object, by providing a *put* method that enables to add the (filter, value) pairs. Available filters and possible values for them are those described at the [Swagger specification](#) for the corresponding web service.

For example, get all samples metadata for cohort *GBR* from study *1kG_phase3* which is framed within project *reference_grch37*:

```
import org.opencb.commons.datastore.core.QueryOptions;
...
...
...
QueryOptions queryOptions = new QueryOptions();
queryOptions.put("study", "reference_grch37:1kG_phase3");
openCGAClient.getCohortClient().getSamples("GBR", queryOptions);
```