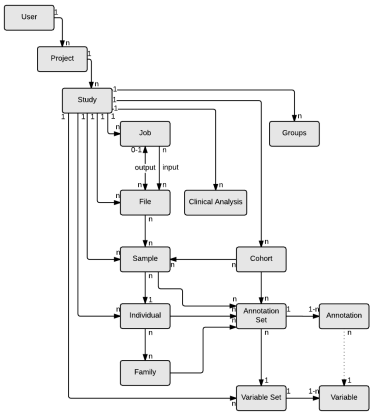
Releases and Versioning

Info

OpenCGA v1.3.0 allows to define internal data *releases* and *versions* of different Catalog data models

Table of Contents:

Remember Catalog Data Models:



Overview

OpenCGA v1.2.0 added support for creating internal data *releases*, a new *release* field was included in the data models (see GitHub issue #616) to indicate in which *release* the data was *created*, note that *rele ase* is implemented as a number and it is an immutable field. OpenCGA 1.3.0 adds a new feature that allows tracking different *versions* of the data models when they are updated (see GitHub issue #684), at the moment *versioning* has been added to *Sample, Individual* and *Family* data models.

These two features together constitute an important milestone for Catalog data management. *Release* an d *versioning* allow users to organise data such as *files* and *samples* in different releases and keep track of all changes. Also, users can now execute more powerful queries such as:

- query data from one or several releases
- query all the versions (whole history) of an entry*
- query a specific version of an entry*
- look for historic data from older releases*

* The only supported entries at the moment are Sample, Individual and Family.

Releases

Many research projects need to create deliverable or snapshot of the data from time to time that will contain everything that has been done so far up to a point. Since version 1.2.0, we added a new field *rele ase* to the data models (see GitHub issue #616) to track when the data such as *files* or *samples* were created. This will also allow OpenCGA to *export* specific releases and to *import* them in another OpenCGA installation, see below.

Release is defined in *project,* this means that different *projects* can have a different release number, notice that all *studies, files, samples, ...* from the same *project* share the same *release* numeric counter. Therefore, *projects* have a *release* counter field showing the current release of the data being ingested at the moment.

How it works

Every time a user creates a new *project,* the project will be created with the *release* set to 1. Only the owner of the project is authorised to increase the *release* and therefore to freeze current working release. This can be done either using a *Project* RESTful web service (*/{version}/projects/{project}/**increlease***) or the command-line

Entries stored in *Project* such as *Study, Sample, File, ...* will be assigned the current *release* number from the project in which they belong. This *release* number is **immutable**, even for the owner of the project, as it reflects when the new data was added.

Querying data by *release*

Now that a new *release* field is present in all entries, it is very easy to query data from different releases. All the ***search*** REST web services include now a new *release* query parameter. Some example queries can be found below:

- query samples created in release 2: .../samples/search?***release***=*2*
- query samples created before release 4: .../samples/search?***release***<*4*

Note: by default the examples above return the last version of each document in each release, as you will see in the next section, *Versioning,* each data model *version* is associated with the *release* in which it was updated.

Versioning

Keeping track of the different versions of the metadata as it is updated is very useful, specially when doing clinical analysis reports that are based on some specific *samples* or *families.* Because clinical data may change overtime, being able to easily fetch old data supporting an old report would be crucial. OpenCGA v1.3.0 (see GitHub issue #684) added version support for *Sample, Individual* and *Family* data models. At the moment, we are only supporting versioning in these entries because they will most probably contain all the clinical information. However, versioning might be extended to other entries in future releases.

How it works

A new numeric field called *version* has been added to *Sample, Individual* and *Family*. This new field is an *autonumeric* and it can be increased with every update only if the user decides to create a new version of the data. Therefore, the *update* REST web service of *Sample, Individual* and *Family* have a new query parameter called ***incVersion***, this parameter is a boolean that indicates whether the version of the entry being updated should be increased, if *true* a **new** document is created for storing new version containing the changes and the incremented version. If *false* then the update is applied in the **same** document and *version* remains equals, by default *incVersion* is false.

On the other hand, Individual and Family *update* web services have another query parameter called ***updateSampleVersion*** and ***updateIndividualVersion*** respectively. Bearing in mind the data models hierarchy, these new Boolean parameters are used to update the version of the references. For example, we can think of an Individual in the database containing two samples. However, the individual is pointing to old versions of those two samples. If we want the Individual to be updated and point to the latest Sample versions, we would set the *updateSampleVersion* boolean field to true.

These parameters could be used with any possible combination or all at the same time. For example, it would be allowed to call to the *individual/update* web service passing some fields to be updated and setting *updateSampleVersion* and *incVersion* to true. In that case, a new version of the individual will be stored in the database containing the changes the user demanded and with the sample references updated to point to their latest versions.

Querying data by *version*

Sample, Individual and Family ***info*** web services now contain two new query parameters, a numeric called ***version*** and a Boolean called ***allVersions***. If the user does not pass any of these parameters, the *info* web service will work as expected, returning the latest version of the data being fetched only. If *allVersions* is set to true, it will return the whole history of the entry, that is, a list containing all the different versions of the entry being requested. Furthermore, the user is also allowed to request a concrete version of the entry using the *version* parameter.

Sample, Individual and Family ***search*** web services have two query parameters called ***snapshot*** and ***release*** to control how to query data. Though they might seem pretty much the same, the connotation is quite different and the results obtained may be really different:

- If the *release* parameter is used, the query will return the latest version of the entries that were **created** in that release number (or range).
- If the *snapshot* parameter is used, the query will return the latest version of the entries in that release. This can be easily understood with an example. Let's imagine we are currently in the release 3 and our sample have 3 versions (1 different version per release), and we specify *snapshot*=2, the sample information we would be fetching would correspond to the version 2 of the data and not the very latest one.

If we don't specify the *snapshot* parameter we will always get the latest version available in the database that matches the criteria specified. Obviously, *release* and *snapshot* parameters can be used together. If we do this we could do queries such as *Give me the latest snapshot available in release 2 of the samples that were created in release 1* for instance.

Export and import

Export data to a different database

The release concept can be normally associated to the concept of deliverables. OpenCGA can be used by a bunch of users that will ingest new data and will be updating it over time to satisfy some kind of deadline. When the time is due, the owner of the project will need to increase the *release counter* so new ingested data is associated to a new *release* number to satisfy the requirements for the next deadline.

Once a *release* is finished, that data could be made available for other kind of users that will only needs access as is (read-only). One of the things OpenCGA offers is the option to export old releases of data to a **read-only** database so other researchers can access that data without interfering the work that might still be in progress in the source database with the next release.

Export

The export option will export **complete projects** up to a specified release number. This means, that if the *release counter* is 4 in the project and the user wants to export up to release 3, all the studies, samples, files... created during releases 1, 2 and 3 and the project itself, will be exported.

When exporting a project, it will never export permissions or groups associated to the studies. This information will be lost in the exported file(s). It will only export the data itself and the cross-references.

The command line would look like:

Export OpenCGA data

```
opencga-admin.sh catalog export --project {project} --output-dir /tmp
/catalog_export --release 5 -p
```

Import

Importing data from other OpenCGA installation is much more trickier than just exporting the data. For this reason, some restrictions need to be satisfied to guarantee that everything will work properly.

1. The very first time something is going to be imported to other database, the database should NOT contain any project, study... However, users are allowed.
2. Imports can be incremental

The command line to import data from a different installation looks like:

Import data to new OpenCGA installation

```
opencga-admin.sh catalog import --directory /tmp/catalog_export --owner
{owner} -p
```