Acceptance Tests

Table of Contents:

Overview

OpenCGA uses *FitNesse* along with *RestFixture* to write and execute Acceptance Tests. Each Test page is an independent collection of tests and can be either executed independently or as a part of Suite run.

Start Fitnesse Server

User can run OpenCGA Fitnesse tests in two different ways, in both cases user must use shell script that we developed to easily execute Fitnesse Java application server, by default port *7070* is used. First mechanism is from OpenCGA source code, this allows developers to easily develop and run tests. To execute tests user must clone and install application with Maven (more detailed information at Building from Source Code):

Clone

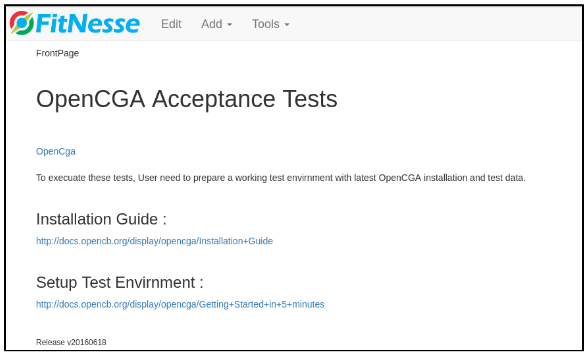
```
## Clone from Git and build with Maven
$ git clone https://github.com/opencb/opencga.git
$ mvn clean install

## Move to test folder and run shell script
$ cd opencga-tests
$ ./target/appassembler/bin/opencga-fitnesse.sh
```

The second mechanism of running the tests is from the installation folder – during the installation process tests are copied – , this allows any user to run the tests without the need of getting the source code and build the application. You must move to the installation folder and execute the following commands:

Start FitNesse Server

```
$ cd test
$ ./bin/opencga-fitnesse.sh                ## you must run this command
from 'test' folder
```

In both cases, after successful start of server, Fitnesse is launched at port *7070,* you can open in a web browser http://localhost:7070/ and click in *OpenCga* link, you should see see something like the following webpage:



OpenCGA Acceptance Tests

OpenCga

To execuate these tests, User need to prepare a working test envirnment with latest OpenCGA installation and test data.

Installation Guide :

http://docs.opencb.org/display/opencga/Installation+Guide

Setup Test Envirnment :

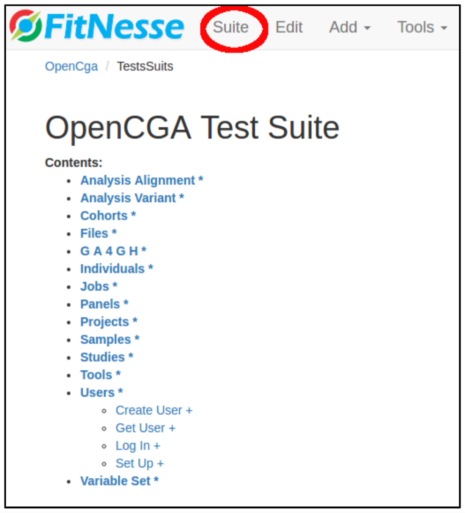
http://docs.opencb.org/display/opencga/Getting+Started+in+5+minutes

Release v20160618

How To Run Acceptance Tests

As a first step, change the **"OPENCGA_VERSION"** and **"TEST_HOST"** variable from SuiteSetUp page and point it to the desired OpenCGA installation.

FitNesse tests can either be executed whole as a Suite by pressing *"Suite"* button on top of page or individual test pages can be independently executed by pressing *"Test"* button on top of that page.

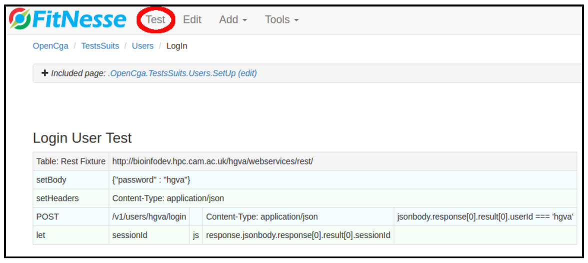


OpenCga / TestsSuits

OpenCGA Test Suite

Contents:

- **Analysis Alignment ***
- **Analysis Variant ***
- **Cohorts ***
- **Files ***
- **G A 4 G H ***
- **Individuals ***
- **Jobs ***
- **Panels ***
- **Projects ***
- **Samples ***
- **Studies ***
- **Tools ***
- **Users ***
  - Create User +
  - Get User +
  - Log In +
  - Set Up +
- **Variable Set ***



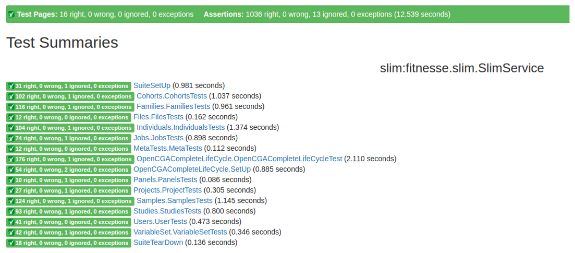
OpenCga / TestsSuits / Users / LogIn

+ *Included page: .OpenCga.TestsSuits.Users.SetUp (edit)*

Login User Test

Table: Rest Fixture	http://bioinfodev.hpc.cam.ac.uk/hgva/webservices/rest/			
setBody	{"password" : "hgva"}			
setHeaders	Content-Type: application/json			
POST	/v1/users/hgva/login		Content-Type: application/json	jsonbody.response[0].result[0].userId === 'hgva'
let	sessionId	js	response.jsonbody.response[0].result[0].sessionId	

After successful execution of tests, A summary of results displayed, one like below :



Test Pages: 16 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 1036 right, 0 wrong, 13 ignored, 0 exceptions (12.539 seconds)

Test Summaries

slim:fitnesse.slim.SlimService

```
31 right, 0 wrong, 1 ignored, 0 exceptions   SuiteSetUp (0.981 seconds)
102 right, 0 wrong, 1 ignored, 0 exceptions  Cohorts.CohortsTests (1.037 seconds)
116 right, 0 wrong, 1 ignored, 0 exceptions  Families.FamiliesTests (0.961 seconds)
12 right, 0 wrong, 0 ignored, 0 exceptions   Files.FilesTests (0.162 seconds)
104 right, 0 wrong, 1 ignored, 0 exceptions  Individuals.IndividualsTests (1.374 seconds)
74 right, 0 wrong, 1 ignored, 0 exceptions   Jobs.JobsTests (0.898 seconds)
12 right, 0 wrong, 0 ignored, 0 exceptions   MetaTests.MetaTests (0.112 seconds)
176 right, 0 wrong, 1 ignored, 0 exceptions  OpenCGACompleteLifeCycle.OpenCGACompleteLifeCycleTest (2.110 seconds)
54 right, 0 wrong, 2 ignored, 0 exceptions   OpenCGACompleteLifeCycle.SetUp (0.885 seconds)
10 right, 0 wrong, 1 ignored, 0 exceptions   Panels.PanelsTests (0.086 seconds)
27 right, 0 wrong, 1 ignored, 0 exceptions   Projects.ProjectTests (0.305 seconds)
124 right, 0 wrong, 1 ignored, 0 exceptions  Samples.SamplesTests (1.145 seconds)
93 right, 0 wrong, 1 ignored, 0 exceptions   Studies.StudiesTests (0.800 seconds)
41 right, 0 wrong, 0 ignored, 0 exceptions   Users.UserTests (0.473 seconds)
42 right, 0 wrong, 1 ignored, 0 exceptions   VariableSet.VariableSetTests (0.346 seconds)
18 right, 0 wrong, 0 ignored, 0 exceptions   SuiteTearDown (0.136 seconds)
```

Coverage

In this section you can find information about which is the current status of acceptance test coverage. The Following are the **general guidelines** which apply to each of the test scenarios described below in tables

- JSON is the expected response for each of tests with either some information or proper error message
  1. No JSON response is blocker
  2. No error message in case of failure is major error
  3. Incorrect error message in case of failure is major error
  4. Insufficient error message will be minor error
  5. Incomplete response message will be a major error
  6. Incorrect data response will be a blocker
- Parameters in **bold** are mandatory fields
- ***Italic*** names inside the **{...}** are variables in WS URL and must be replaced with value
- In most use cases id/alias can be used interchangeably
- id, name and alias are unique items EXCEPT for files and jobs
- **limit**, **skip** should be tested for each web services where applicable
- **include**, **exclude** are exclusive in usage and should be tested where applicable
- contents inside **{ }** are Post Body data
- Each of the test cases should cover **Positive** as well as **Negative** scenarios
- OpenCGA Version Column
  - ✅ Test has successfully passed
  - ❌ Test has been failed
  - **No Entry**: Functionality not yet implemented / Test case has **not** yet been developed.

Users

Path (/users)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	{ **userId, name, email, password,** organization}	Creates a new user		✓	✓
/{*user*}/login	POST	**user, password**	login user and return a JWT token			✓
/{*user*}/update	POST	**user,** name, email...	Updates user data			✓
/{*user*}/info	GET	**user**	Returns all information related to the specific user		✓	✓
/{*user*}/projects	GET	**user,** shared	Retrieves the list of projects and studies belonging or shared with the user		✓	✓
/{*user*}/change-password	POST	**user,** { **password, npassword**}	Changes the password of a user		✓	✓
/{*user*}/configs/create	POST	**user, name { }**	This should store the preferences of user in catalog			
/{*user*}/configs/{*name*}/info	GET	**user, name**	Fetch a user information			
/{*user*}/configs/{*name*}/delete	GET	**user, name**	Deletes a user configuration			
/{*user*}/configs/filters/create	POST	**user, {** name, description, bioformat,query, options }	Stores a predefined, frequent used set of filters			
/{*user*}/configs/filters/{*name*}/info	GET	**user, name**	Fetch a filter of given name			
/{*user*}/configs/filters/list	GET	**user**	Fetch all the filters for a user			
/{*user*}/configs/filters/{*name*}/update	POST	**user, name {...}**	Update fields provided in body of a specific filter			
/{*user*}/configs/filters/{*name*}/delete	GET	**user, name**	Deletes a custom filter provided as name			
/delete	GET {PENDING}	**user**	Deletes a given user			

Projects

Path (/projects)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	**name, alias, organism.scientificName,** **organism.assembly,** organization	Creates new project		✓	✓
/{*project*}/info	GET	**projects**	Fetches project information		✓	✓

/{*project*}/studies	GET	**projects**	Fetch all the studies inside a project		✓	✓
/{*project*}/update	POST	**project**	Update project, mandatory fields must not be allowed to update		✓	✓
/{*project*}/increlease	POST	**project**	Increment the current release number If new data is created			✓

Studies

Path (/studies)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb 2017)	**1.1 (May 2017)**	**1.2 (Jul 2017)**
/create	POST	**projectId, name, alias**	Creates new study attached with a project. Test all mandatory fields and uniqueness		✓	✓
/{*study*}/groups/create	POST	**study, {groupId,** list of users}	Creates a group with optional user list		✓	✓
/{*study*}/update	POST	**study**	Updates some of the data inside study and verify			✓
/{*study*}/groups/{*group*}/update	POST	**study**, **groupId**, {addUsers, setUsers, removeUsers}	Updates the members of the group addUsers – add new member removeUsers – remove existing member setUsers – delete existing members list and update with new one		✓	✓
/{*study*}/groups/{members}/update	POST	**study**, **memberIds**, { "permissions": "", "action": "", "study": "", "template": ""}	Updates the members of the group			✓
/search	GET	**projectId,** name, alias, type	Search studies based on different parameter combinations		✓	✓
/{*study*}/info	GET	**study**	Finds info for existing and non existing study using id and alias		✓	✓
/{study}/summary	GET	**study**	Fetch study information plus some basic stats		✓	✓
/{*study*}/scanFiles	GET	**study**	Scans the study folder to find untracked or missing files			
/{study}/files	GET	**study**, id, name , path, type, bioformat	Fetch flies in study, using different combinations of parameters and verify			
/{study}/groups	GET	**study**	Returns the groups present in the studies		✓	✓
/{study}/samples	GET	**study**, name, individualId, annotationSetName, variableSetId, annotation	Fetch samples in the study, use different combination of the available parameters			
/{*study*}/groups/{*groupId*}/info	GET	**study, groupId**	Returns the group		✓	✓
/{*study*}/groups/{*groupId*}/delete	GET	**study, groupId**	Deletes the group			

/{study}/resyncFiles	GET	**study**	This method is intended to keep the consistency between the database and the file system. It will check all the files and folders belonging to the study and will keep track of those new files and/or folders found in the file system as well as update the status of those files/folders that are no longer available in the file system setting their status to MISSING. (Tester have to create those files using file create or manually to test this service)			
~~/{study}/acl/create~~	POST	**study,** permissions, **members**, templateId	Defines a set of permissions for a list of users or groups		✓	
/{*study*}/acl	GET	**study**	Returns the ACL of the study		✓	✓
/{*study*}/acl/{*memberId*}/info	GET	**study, memberId**	Returns the set of permissions granted for the user or group		✓	✓
/{*study*}/acl/{memberId}/update	POST	**study, memberId,** {JSON containing one of the keys 'add', 'set' or 'remove'}	Updates the set of permissions granted for the user or group		✓	✓
/{*study*}/acl/{*memberId*}/delete	GET	**study, memberId**	Deletes all the permissions granted for the user or group		✓	✓

Files

Path (/files)	**HTTP Method**	**Parameters**	**Description**	**OpenCGA Version**		
				1.0 (Feb 2017)	**1.1 (May 2017)**	**1.2 (Jul 2017)**
/create	POST	**path,** content**,** description, directory	Creates a file		✓	✓
/upload	POST	**file, fileformat, bioformat, relativeFilePath**	Uploads a file to OpenCGA server			
/update	POST	**file, ...**	Modify file			
/info	GET	**files,** study	Get the File info		✓	✓
/search	GET	id, study, name, type, bioformat	Multi-study search that allows the user to look for files from from different studies of the same project applying filters.			
/bioformats	GET		Return the list of accepted file bioformats			
/formats	GET		Returns the list of accepted formats			
/groupBy	GET	**fields,** study, id, name, path , type, buoformat, format, smapleIds	Group files based on different combinations of filters			
/{*folder*}/scan	GET	**folder,** study	Scans a folder			
/{folder}/list	GET	**folder**, study	List all the files inside a folder			
/{*file*}/content	GET	**file,** study	Show contents of a file (limited)			
/{*folder*}/tree	GET	**folder,** study	Tree view of the files and folders inside		✓	✓

/{*file*} /download	GET	**file,** study	Download a file			
/{*file*} /update	POST	**file,** study {Parameters to modify}	Modify the file attributes			
/{*file*} /refresh	GET	**file**, study	Refresh meta data related to a file/folder and returns updated files			
/{*file*} /delete	GET	**file,** study	Deletes a file			
/{*files*} /acl /create	POST	**files,** study **{members}**	Defines a set of permissions for a list of users or groups			
/*files*/ {files}/acl	GET	**files**	Returns the ACL defined for file / folder		✓	✓
/*files*/ {file}/grep	GET	**file**	Filter lines of the file containing a match of the pattern			
/{*file*}/acl/ {*member Id*}/info	GET	**file, memberId**	Returns the permissions granted for the user or group			
/{*file*}/acl/ {*member Id*} /update	POST	**file, memberId** {add, set, remove}	Updates the permission granted for the user or group			
/{*files*}/acl / {*member Id*}/delete	GET	**files, memberId**	Removes all the permissions granted for the user or group			

Jobs

Path (/jobs)	**HTTP Method**	**Parameters**	**Description**	**OpenCGA Version**		
				1.0 (Feb 2017)	**1.1 (May 2017)**	**1.2 (Jul 2017)**
/create	POST	**{ name...}**	Registers a job that has been previously run outside catalog into catalog.		✓	✓
/{*jobId*}/info	GET	**JobId**	Get the job information		✓	✓
/search	GET	study, name [Pending]			✓	✓
/groupBy	GET	**fields**	Group jobs by several fields		✓	✓
/{*jobId*}/visit	GET	**jobId**	Increment job visits		✓	✓
/{*jobIds*} /delete	GET	**jobIds**	Deletes job(s)		✓	✓
/{*jobIds*}/acl /update	POST	**jobIds, {members}**	Defines a set of permissions for a list of members			✓
/{*jobIds*}/acl	GET	**jobIds**	Returns the ACL of the job		✓	✓

/{*jobId*}/acl/{
memberId}
/info
	GET	**jobId, memberId**	Returns the set of permissions granted for the member		✓	✓

/{*jobId*}/acl/{
memberId}
/update
	POST	**jobId, memberId, {add , set, remove}**	Updates the set permissions granted for the member		✓	✓

/{*jobId*}/acl/{
memberId}
/delete
	GET	**jobId, memberId**	Removes all the permissions granted for the member		✓	✓

Individuals

				OpenCGA Version

Path (/individuals)	HTTP Method	Parameters	Description	1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	study, **{ name...}**	Creates a new Individual with data provided in body		✓	✓
/{*individuals*} /info	GET	**individuals**	Gets individual information		✓	✓
/{*individuals*} /search	GET	name, fatherId, motherId, species.scientificName, population.name	Search the individual using different combination of available fields		✓	✓
/{*individuals*} /groupBy	GET	**fields,** name, fatherId, motherId, species. scientificName,population. name	Returns data into group based on fields provided			
/{individual} /update	POST	**individual, {..}**	Updates the given param inside body for a particular individual		✓	✓
/{*individuals*} /delete	GET	**individuals**	Deletes individuals information			
/{*individual*} /annotationsets /create	POST	**individual, variableSetId, {**JSON containing the annotation set name and the array of annotations. The name should be unique for the individual **}**	Creates an annotation set for a particular user		✓	✓
/{*individual*} /annotationsets/ annotationsetName/info	GET	**individual, individualSetName**	Fetches info related to Annotation Set			
/{*individual*} /annotationsets /annotationsetName/search	GET	**individual**				
/{*individual*} /annotationsets /annotationsetName/update	POST	**individual, memberId, {add, set, remove}**	Updates the set of permissions granted for the member			
/{*individual*} /annotationsets /annotationsetName/delete	GET	**individual, individualSetName**	Deletes an Annotation Set			
/{*individuals*}/acl /create	POST	**individuals, {members}**	Creates ACL for a list of individuals		✓	✓
/{*individuals*}/acl	GET	**individuals**	Returns ACL related to individuals		✓	✓
/{*individuals*} /acl/{memberId} /info	GET	**individualId,memberId**	Get the ACL info related to the specified member		✓	✓
/{*individual*}/acl/ {*memberId*} /update	POST	**individual, memberId, {add, set, remove}**	Updates the ACL info related to a specified member		✓	✓
/{*individuals*} /acl/{*memberId*} /delete	GET	**individuals, memberId**	Removes the ACL related to aindividual(s) of a specified member			

Cohorts

Path (/cohorts)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	study, **{ name...}**	Creates a new Cohort with data provided in body		✓	✓
/{*cohortId*}/info	GET	**cohortId**	Gets Cohort information		✓	✓
/search	GET	study, name, source, sample, status, type	Search for the cohorts using different combination of available fields		✓	✓
/groupBy	GET	**fields,** name, id, type, status....	Returns data into group based on fields provided			
/{cohort}/update	POST	**cohort, {..}**	Updates the given param inside body for a particular cohort		✓	✓
/{*cohort*}/delete	GET	**cohort**	Deletes given cohort		✓	✓
/{*cohort*}/annotationsets/create	POST	**cohort, variableSetId, {**JSON containing the annotation set name and the array of annotations. The name should be unique for the sample **}**	Creates an annotation set for a particular user			
/{*cohort*}/annotationsets/annotationsetName/info	GET	**cohort, samplesetName**	Fetches info related to Annotation Set			
/{*cohort*}/annotationsets/annotationsetName/search	GET	**cohort**				
/{*cohort*}/annotationsets/annotationsetName/update	POST	**cohort**, **anootationSetName** {}				
/{*cohort*}/annotationsets/annotationsetName/delete	GET	**cohort, samplesetName**	Deletes an Annotation Set			
/{*cohorts*}/acl/create	POST	**cohorts, {members}**	Creates ACL for a list of samples		✓	✓
/{*cohorts*}/acl	GET	**cohorts**	Returns ACL related to samples		✓	✓
/{*cohort*}/acl/{memberId}/info	GET	**cohortId,memberId**	Get the ACL info related to the specified member		✓	✓
/{*cohort*}/acl/{*memberId*}/update	POST	**cohort, memberId, {add,set, remove}**	Updates the ACL info related to a specified member		✓	✓
/{*cohort*}/acl/{*memberId*}/delete	GET	**cohorts, memberId**	Removes the ACL related to asample(s) of a specified member		✓	✓

Samples

				OpenCGA Version

Path (/samples)	HTTP Method	Parameters	Description	1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	study, **{ name...}**	Creates a new sample with data provided in body		✓	✓
/load	GET	**file**	Load samples from a file			
/{*samples*}/info	GET	**samples**	Gets sample information		✓	✓
/{*samples*} /search	GET	name, source, individual.id, annotationSsetName, variableSetId, Annotation	Search the sample using different combination of available fields		✓	✓
/{*samples*} /groupBy	GET	**fields,** name, individual.id, annotationSetName, variableSetId, Annotation	Returns data into group based on fields provided		✓	✓
/{sample} /update	POST	**sample, {..}**	Updates the given param inside body for a particular sample		✓	✓
/{*samples*} /delete	GET	**samples**	Deletes samples information		✓	✓
/{*sample*} /annotationset s/create	POST	**sample, variableSetId, {**JSON containing the annotation set name and the array of annotations. The name should be unique for the sample **}**	Creates an annotation set for a particular user		✓	✓
/{*sample*} /annotationset s /annotationset Name/info	GET	**sample, samplesetName**	Fetches info related to Annotation Set			
/{*sample*} /annotationset s /annotationset Name/search	GET	**sample**				
/{*sample*} /annotationset s /annotationset Name/update	GET	**sample**, **anootationSetName** {}				
/{*sample*} /annotationset s /annotationset Name/delete	GET	**sample, samplesetName**	Deletes an Annotation Set			
/{*samples*}/acl /create	POST	**samples, {members}**	Creates ACL for a list of samples		✓	✓
/{*samples*}/acl	GET	**samples**	Returns ACL related to samples			
/{*samples*}/acl/ {memberId} /info	GET	**sampleId,memberId**	Get the ACL info related to the specified member		✓	✓
/{*sample*}/acl/{ *memberId*} /update	POST	**sample, memberId, {add,set, remove}**	Updates the ACL info related to a specified member		✓	✓
/{*samples*}/acl/ {*memberId*} /delete	GET	**samples, memberId**	Removes the ACL related to asample(s) of a specified member		✓	✓

VariableSet

Path (/variableset)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	study, **{ name...}**	Creates a new variable set with data provided in body		✓	✓
/{*variableset*} /info	GET	**variableset**	Gets variable Set information		✓	✓
/{*variableset*} /search	GET	name, id	Search for the variable set using different combination of available fields		✓	✓
/{*variableset*} /update	POST	**variableset, {..}**	Updates the given param inside body for a particular variableSet			
/{*variableset*} /field/add	POST	**variableset**	add a new field in variableSet		✓	✓
/{*variableset*} /field/rename	GET	**variableset, oldName, newName**	renames a given field in variable Set with new name		✓	✓
/{*variableset*} /field/delete	GET	**variableset, name**	deletes a given field from vairableSet		✓	✓

Meta

Path (/meta)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/about	GET		Gets information related to OpenCGA version, branch, commit		✓	✓
/ping	GET		returns pong, simplest way to see if REST is up and running		✓	✓
/status	GET		returns OK in case of system working properly		✓	✓

Families

Path (/families)	HTTP Method	Parameters	Description	OpenCGA Version		
				1.0 (Feb. 2017)	1.1 (May 2017)	1.2 (Jul 2017)
/create	POST	study, **{ name...}**	Creates a new family with data provided in body		✓	✓
/{*families*}/info	GET	**familyId(s)**	Gets Families information		✓	✓
/search	GET	study, name, mother, father, children, ...	Search for the families using different combination of available fields		✓	✓
/{family} /update	POST	**family, {..}**	Updates the given param inside body for a particular family		✓	✓

/{*family*}/annotationsets/create	POST	**family, variableSetId, {**JSON containing the annotation set name and the array of annotations. The name should be unique for the sample **}**	Creates an annotation set for a particular user		✓	✓
/{*family*}/annotationsets/annotationsetName/info	GET	**family, samplesetName**	Fetches info related to Annotation Set		✓	✓
/{*family*}/annotationsets/annotationsetName/search	GET	**family**				
/{*family*}/annotationsets/info	GET	**family**	return the annotation sets for the family		✓	✓
/{*family*}/annotationsets/annotationsetName/update	POST	**family**, **anootationSetName** {}				
/{*family*}/acl/{*memberId*}/update	POST	**families, {members}**	Creates/update ACL for a list of samples		✓	✓
/{*families*}/acl	GET	**families**	Returns ACL related to samples		✓	✓