Allele Registry

versions 1.01.xx

Administrator's manual

document version 2

Table of Contents

Introduction

This document contains description of additional functionality of Allele Registry that is available only for users with administrator privileges, as well as, some features that are not described in official documentation available for API users. The document does not repeat information included already in the API specification.

Users privileges

API requests can be send with or without authentication tokens. The authentication procedure is described in details in API specification. It is based on Genboree authentication and uses connection to MySQL Genboree database to verify users credentials. In general, each requests is assigned to one of the four authorization levels described in the table below.

Authorization level	**Requirements**	**Access rights**
Anonymous	• request without authentication parameters (gbLogin, gbTime, gbToken)	• all GET and POST requests
Authenticated user	• request with correct authentication parameters (gbLogin, gbTime, gbToken) • user assigned to Genboree group “Registry”	• all GET and POST requests • new variants registration
Authenticated user assigned to external source	• request with correct authentication parameters (gbLogin, gbTime, gbToken) • user assigned to Genboree group “Registry” • user assigned to one of external sources	• all GET and POST requests • new variants registration • adding and removing links from assigned external sources
Administrator	• request with correct authentication parameters (gbLogin, gbTime, gbToken) • user assigned to Genboree group “Registry” • user login set in an array “superusers” in the configuration file	• all requests

When Allele Registry is run with the parameter “--no_auth”, authentication parameters are ignored and all request are authorized on “Administrator” level. In this mode, Allele Registry server can be run without installed Genboree (wowever, MySQL with “externalSources” database is still required).

Responses in simple text format

All requests returning as a response a single Allele object or an array of Alleles objects support additional protocol named “txt”. When this protocol is specified in the request, a response contains simple text file instead of JSON content. The text content consists of two TAB-separated columns, the first column contains single HGVS expressions (alleles definitions) while the second column contains CA/PA identifiers. This format is useful for dumping main part of Allele Registry database (allele definitions and CA/PA identifiers).

Example

```
Request: reg.genome.network/allele/CA000559.txt
```

Response:

```
NC_000010.11:g.87957955C>T    CA559
```

Alleles Database

Alleles database consists of several files, each of them corresponds to single “table”. The most important are the three following files:

- genomic
- protein
- sequence

They contain all data stored by Allele Registry. If the Registry cannot find some of these files, it creates new empty table (it allows to start Registry without database).

The database contains also additional “tables”, which work as indexes:

- idCa
- idPa
- idClinVarAllele
- idClinVarRCV
- idClinVarVariant
- idDbSnp

If at least one of these files is missing when Registry process is starting, the Registry tries to rebuild missing index (or indexes) by performing full scan of “genomic” and “protein” tables. During this procedure, identifiers of proper types are collected and missing indexes are built from scratch. The full scan procedure may take long time for large databases.

Administrator’s API for managing alleles and external identifiers

As “identifiers” we understand here all identifiers presented in the section “externalRecords” in Allele’s document. It includes identifiers imported from large datasets like dbSNP, MyVariant.Info or gnomAD.

Import new identifiers

This is an extended version of the request used for bulk registration of new alleles that was described in API specification (see “Bulk query of alleles with HGVS expressions or identifiers” and “Register new alleles”). When the request is sent by an administrator, a payload may contain additional columns with identifiers to register.

The content sent as payload is processed in the following way:

1. the system identify the variant by “Key” column (variant definition or CA/PA ID)
2. if the variant is not in the database, it is automatically registered (CA/PA ID is assigned)
3. all defined identifiers are added to the variant; if given identifier is already registered (with exactly the same type and value), it is ignored
4. the request returns vector of Allele documents, each document corresponds to single line in the payload

The payload must contain exactly one “Key” column. The only exception is allowed, when CA/PA ID column is defined along with the other “Key” column – in this special case, variants from payloads are recognized by the other “Key” column and registered with given CA/PA ID values. **This special case is very dangerous and should be used only in exceptional situation. The uniqueness of CA/PA ID is not verified by the system, the administrator has to make sure that submitted variants and CA/PA identifiers are unique.**

Request

```
PUT /alleles
```

Parameters

- *file* – contains non-empty list of columns names separated by plus characters (‘+’). This expression defines the format of a payload. The expression must meet the following conditions:
  - the same column name cannot appear twice in the expression, the only exception is *ignored* column represented by empty string in the *file* parameter

- the expression must contain exactly one “Key” column, the only exception is CA/PA ID column represented by “id”; it can be used together with other “Key” column (see special case in the description above)

The table below summarize all supported columns names:

Column name	Key?	Required format in payload
hgvs	Yes	well-defined HGVS expression
id	Yes	CA ID or PA ID
MyVariantInfo_hg19.id	Yes	e.g. chr9:g.107620835G>A
MyVariantInfo_hg38.id	Yes	the same as above
ExAC.id	Yes	e.g. 5-112043382-A-G
gnomAD.id	Yes	the same as above
(empty string)	No	this column is ignored, useful for files with extra columns
dbSNP.rs	No	a number or an empty string if none
ClinVar.alleleId	No	a number or an empty string if none
ClinVar.variationId	No	a number or an empty string if none
ClinVar.preferredName	No	a string, must have already a “ClinVar.alleleId” column defined if non-empty
ClinVar.RCV	No	list of numbers separated by comma characters (‘,’), must have already a “ClinVar.variationId” column defined if non-empty
COSMIC.id	No	COSMIC id with suffix “/0” (obsolete) or “/1” (active), e.g.: COSM2240643/1 COSM5424721/0 COSN19651925/1 or an empty string if none
externalSource.{sourceName}	No	see “External sources” in API specification

Payload

It must contain list of records separated by end-of-line character. Each record must consist of the same number of fields (columns) separated by a TAB character. The number and types of columns must correspond to definition given in the “file” parameter described above.

Response

A vector of corresponding Allele documents in JSON. You probably should use “fields” parameter to normalize processing time, like “fields=none+@id” (see “Adjusting response format” in API specification).

Delete all identifiers of given type

These request deletes all identifiers (whole entries in “externalRecords” section) originating from given external system.

Request

`DELETE /externalRecords/{name}`

`{name}` must be one of the following: dbSNP, ClinVar

Response

Empty JSON object.

Delete alleles by CA/PA IDs

This requests expects payload with list of CA/PA IDs (one per line). Existing alleles with given CA/PA identifiers are deleted, along with assigned to them external identifiers and links from external sources.

This request is not suppose to be used under normal circumstances! It is non-standard in two ways:

- **according to main assumptions of Allele Registry, registered alleles along with their CA/PA identifiers are not supposed to be deleted**
- **HTTP DELETE requests are not supposed to have a payload (many proxy servers do not support that), so this request should be sent only to localhost**

Request

`DELETE /alleles?file=id`

Payload

Single column with CA/PA identifiers

Response

A vector of corresponding JSON Allele documents, but only with some basic data.

Administrator’s API for External Sources

See “Allele Registry API specification” for description of External Sources. This sections contains only requests available from administrator’s level.

Final links and their labels are generated from patterns set in an External Source object. Patterns can use references to external source parameters, that are substituted with proper values during generation of final textual content. Currently, the following parameters references can be used: {id}, {p1}, {p2}, {p3}. The first one corresponds to raw integer value of CAR allele id, while values of other three parameters are set by the user along with link definitions. Every parameter holding integer value can be zero-padded by append to the parameter’s name a character “_” (underscore) followed by a number. Eg.: to get “standard” CA identifier the pattern “CA{id_6}” may be used.

Create new external source

Request

`PUT /externalSource/{sourceName}`

`{sourceName}` – a name of the external source, it is used in API. The name can contain only alphanumeric characters and underscores (0-9, a-z, A-Z, _).

Parameters

- *guiName* - user-friendly name for external source
- *params* - defines number and types of external source parameters. It must contain from 0 to 3 characters, each character must be S (string) or I (integers). E.g. .: SI means that an external source has 2 parameters (p1 and p2), where p1 is a string and p2 is an unsigned integer.
- *url* - pattern for API url, it is used for generation of the value of “@id” field
- *guiUrl* - pattern for GUI url, it is used for generation of the value of “guiUrl” field
- *guiLabel* - pattern for user-friendly link label, it is used for generation of the value of “guiLabel” field

Modify existing external source

Name of the external source, as well as, number and types of its parameters cannot be altered.

Request

`PUT /externalSource/{sourceName}`

`{sourceName}` – a name of the external source

Parameters:

Exactly one of the following parameter must be set:

- *guiName* - new "guiName" parameter value, see the request above
- *url* - new "url" parameter value, see the request above
- *guiUrl* - new "guiUrl" parameter value, see the request above
- *guiLabel* - new "guiLabel" parameter value, see the request above

Delete external source

It deletes an external source with whole content, including all links.

Request

`DELETE /externalSource/{sourceName}`

`{sourceName}` – a name of the external source

Assign user to an external source

Request

`PUT /externalSource/{sourceName}/user/{login}`

`{sourceName}` - name of an external source
`{login}` - name of Genboree user, it is not verified by the system

Remove user from an external source

Request

`DELETE /externalSource/{sourceName}/user/{login}`

`{sourceName}` - name of an external source
`{login}` - name of Genboree user, it is not verified by the system

Unpublished standard API requests

This section list all API requests available for all users but not published in API specification. Since they are not published, we do not have to support them in future releases.

Query all genomic alleles

In connection with “txt” protocol described in the Introduction it can used to dump to a simple text file list of all genomic alleles.

Request

```
GET /genomicAlleles
```

Parameters

- (optional) *skip* – standard pagination parameter
- (optional) *limit* – standard pagination parameter

Response

An array of Allele objects.

Query all protein alleles

In connection with “txt” protocol described in the Introduction it can used to dump to a simple text file list of all protein alleles.

Request

```
GET /proteinAlleles
```

Parameters

- (optional) *skip* – standard pagination parameter
- (optional) *limit* – standard pagination parameter

Response

An array of Allele objects.

Return basic properties of given reference sequence

Request

```
GET /refseq/{RSid}/sequence
```

Response

Returned object is similar to standard Reference Sequence document, but contains also some additional fields like:

- CDS – region of transcript's coding sequences
- sequence – sequence, for transcript it contains all introns
- spliced sequence – transcript's sequence after splicing (without introns)

Match protein and transcript variants for gene mutation

This request returns all matching variants for notation gene+mutation. Both protein and transcript variants are calculated. If transcript mutation is given, protein mutations are calculated directly from matching transcript mutations. If protein mutation is given, corresponding transcript mutations are searched across registered alleles.

Request

```
GET /matchAlleles
```

Parameters

- gene – symbol of gene (e.g. TP53, GALE)
- variant – mutation in HGVS notation without reference (e.g.: c.930C>G, p.Asn310Lys)

Response

Not trivial.