ODM Advanced Training

Maria Borodaenko
Senior Solution Consultant

Genestack

Program

Welcome to ODM Advanced training

- Learning goals
- Training
- Q&A Session

Program

- **Learning goals**
- Training
- Q&A Session

Learning Goals

Get understanding of ODM APIs

- How to get access
- How to load and link data
- How to query available data

Making Data FAIR and Action-Ready for Both Consumers & Curators

DATA CURATORS

RESEARCHERS

DATA SCIENTISTS

Metadata Curation

Harmonise thousands of samples
Enforce minimum metadata model

Integrated Data Catalog

Find data across sources
Explore study-sample-data links

Search Services

Slice-and-dice analysis-ready data
Write R Shiny apps rapidly

Making Data FAIR and Action-Ready for Both Consumers & Curators

DATA CURATORS

Metadata Curation

Harmonise thousands of samples
Enforce minimum metadata model

RESEARCHERS

Integrated Data Catalog

Find data across sources
Explore study-sample-data links

DATA SCIENTISTS

Search Services

Slice-and-dice analysis-ready data
Write R Shiny apps rapidly

Program

- Learning goals
- **Training**
- Q&A Session

Training

Genestack

Training

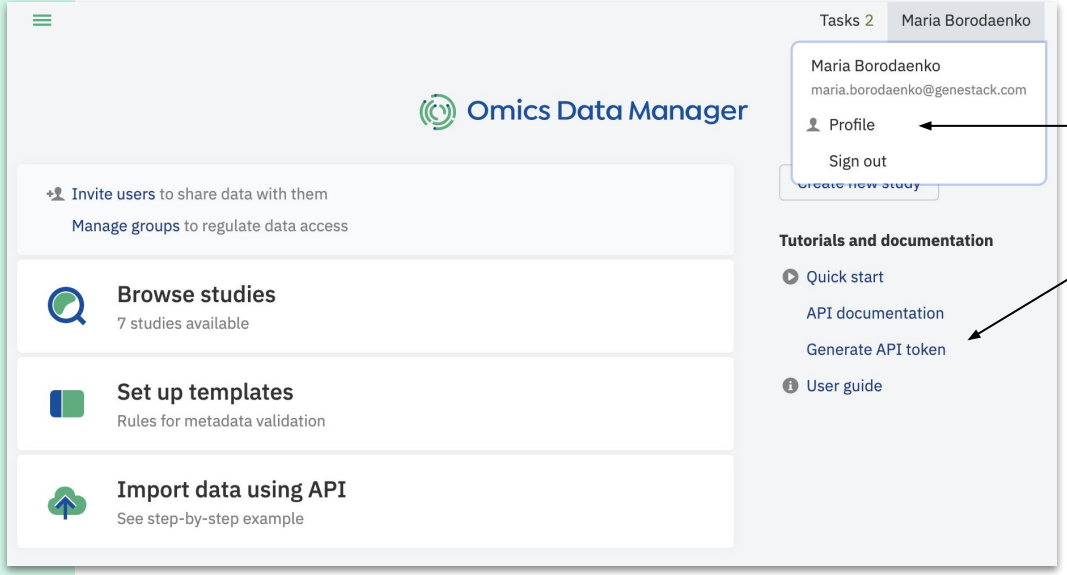
- API access - Getting access token
- Introducing data model and API organisation
- Data loading - Loading and linking test dataset
- Data versioning - Adding a new version of expression results
- Data querying - Applying filters to slice and dice your data

Training

- **API access - Getting access token**
- Introducing data model and API organisation
- Data loading - Loading and linking test dataset
- Data versioning - Adding a new version of expression results
- Data querying - Applying filters to slice and dice your data

How to Get an API Token

Personal access tokens allow you to access your data via the REST API. The tokens are permanent. You can have multiple tokens and revoke them at any time.



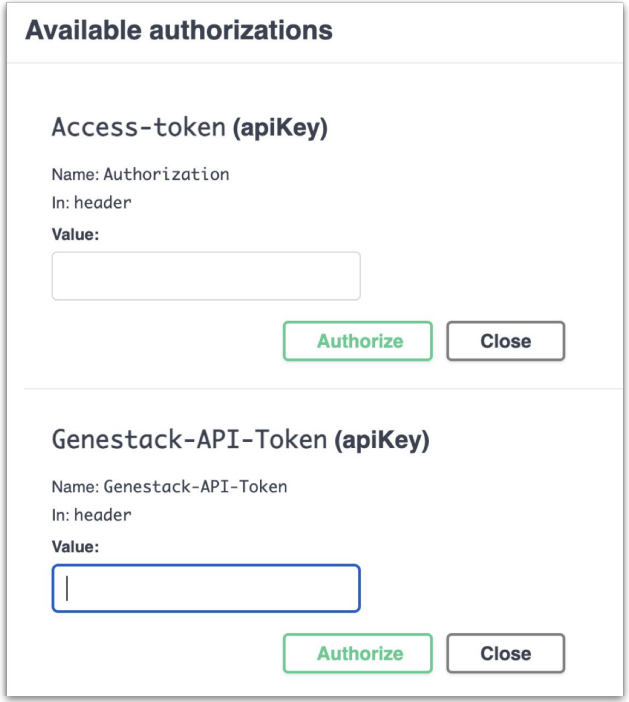
To request a token navigate to your profile or click “Generate API token” on the starting page.

Authorisation via an Access Token

Alternatively authorisation via Access token from Identity provider, e.g. Azure AD can be used. To specify the Access token use the “Authorisation” header, to specify the Genestack API token use the “Genestack-Api-Token” header.

<u>Note:</u> Access token takes precedence, meaning that if both tokens are supplied, the access token will be used for processing the request.

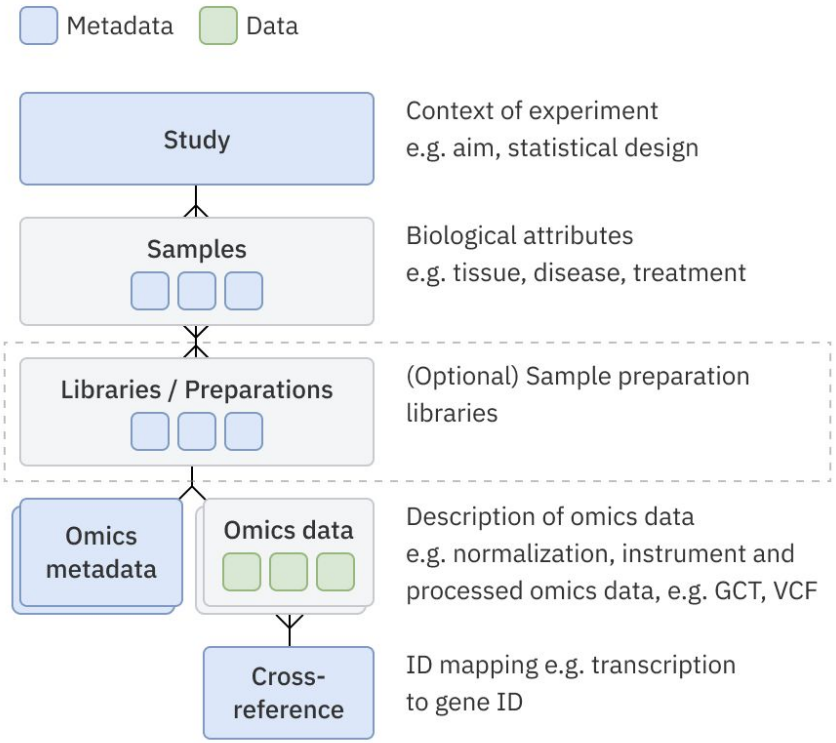
<u>Note 2:</u> The solution has been tested with the Azure AD access tokens only. For other providers pretesting is recommended.



Training

- API access - Getting access token
- **Introducing data model and API organisation**
- Data loading - Loading and linking test dataset
- Data versioning - Adding a new version of expression results
- Data querying - Applying filters to slice and dice your data

Data Model



Genestack APIs:

- Create, edit, link and query data
- Cross-study, cross-omics, data-metadata search
- Call via HTTP requests, Python, or R

Import	GUI	Python Script
Study	Manual	TSV
Samples	Manual or TSV import	TSV
Libraries / Preparations (optional)	X	TSV
Omics metadata	X	TSV
Omics data	X	GCT, VCF, FACS

Data Requirements

1. **Format.**
   All metadata should be in .tsv format.
   Expression data in .gct, variant data in .vcf.

2. **Mandatory attributes.**
   Study "Study Title" - used for displaying the name.
   Samples "Sample Source" with values and "Sample Source ID" with unique values
   Libraries "Library ID" with unique values
   Preparations "Preparation ID" with unique values

3. **No duplicated attributes**

Study Title	…
The name to be displayed	…

Sample Source	Sample Source ID	…
Source_1	Sample_1_ID	…
Source_1	Sample_2_ID	…

Library ID	Sample Source ID	…
Library_1_ID	Sample_1_ID	…
Library_2_ID	Sample_2_ID	…

Linking Attributes

Headers in the expression file should correspond to sample metadata so the system recognise what column relates to what sample.

During import user can specify which attribute from the template should be used for linking during import.

If nothing is specified the “Sample Source ID” attribute is used.

Can be any template attribute

Sample file

Name	Sample Source	Sample Source ID
Sampe_1	Lab_1	Sample_1_ID
Sample_2	Lab_1	Sample_2_ID
Sample_3	Lab_1	Sample_3_ID

Expression file

#1.2			
24571	60		
Name	Description	Sample_1_ID	Sample_2_ID
Gene_1		2.23451	0

Linkage for Model with Libraries/Preps

For automated linking Libraries/Preparations to should have the “Sample Source ID” attribute with corresponding samples’ IDs.

Headers in the expression file should correspond to values in the “Library ID”.

Custom linking attributes are not supported.

Sample Source	Sample Source ID	…
Source_1	Sample_1_ID	…
Source_1	Sample_2_ID	…

Library ID	Sample Source ID	…
Library_1_ID	Sample_1_ID	…
Library_2_ID	Sample_2_ID	…

#1.2			
24571	60		
Name	Description	Library_1_ID	Library_2_ID
Gene_1		2.23451	0

How Are the APIs Organised?

Query/retrieve data

- integrationUser - API endpoints for conducting multi-omics que (signal) data and metadata for a given query
- studyUser - API endpoints for retrieving only study metadata
- sampleUser - API endpoints for retrieving only sample metadat
- libraryUser - API endpoints for retrieving only library metadata
- preparationUser - API endpoints for retrieving only preparation
- expressionUser - API endpoints for retrieving only expression d
- variantUser - API endpoints for retrieving only variant data or m
- flowCytometryUser - API endpoints for retrieving only flow cyto

Import/curate data

- integrationCurator - API endpoints to link experimental data/me
- studyCurator - API endpoints to add, delete and update studies
- sampleCurator - API endpoints to add, delete and update samp
- libraryCurator - API endpoints to add, delete and update librarie
- preparationCurator - API endpoints to add, delete and update p
- expressionCurator - API endpoints to add, delete and update ex
- variantCurator - API endpoints to add, delete and update varian
- flowCytometryCurator - API endpoints to add, delete and updat
- job - Experimental API endpoints to run asynchronous jobs (imp
- tasks - API endpoints to work with asynchronous tasks

Data sources

- reference-data - Experimental API endpoints to create, delete,

APIs are documented in Swagger

Each data type has its own set of endpoints for CRUD operations - **SPoT** (Single Point of True)

Integration layer allows to link entities and integratively query different data types. It “knows” about relationships between objects from different SPoTs

User types user (researcher) and curator

- User endpoints: only retrieve and query data
- Curator endpoints: import, curate, retrieve and query data

Training

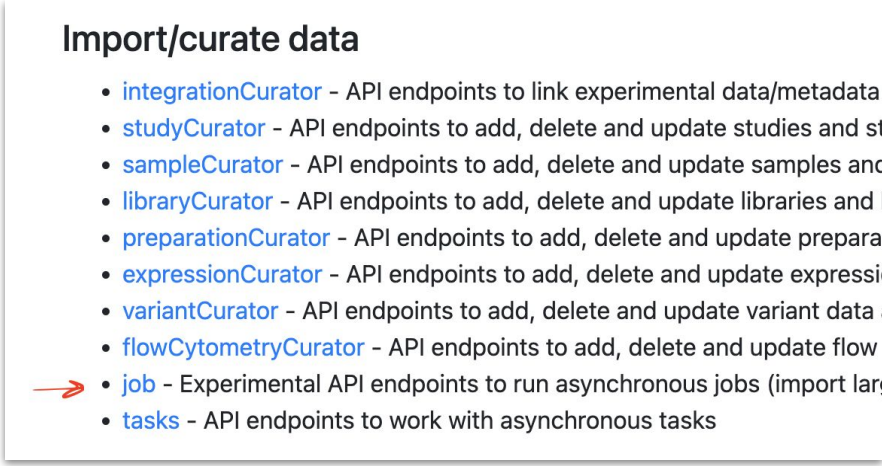
- API access - Getting access token
- Introducing data model and API organisation
- **Data loading - Loading and linking test dataset**
- Data versioning - Adding a new version of expression results
- Data querying - Applying filters to slice and dice your data

Data Loading via APIs

To load the data via APIs each entity is created via a separate endpoint specific for this data type. Then they are sequentially linked in the Integration layer. Loading data via asynchronous jobs is recommended.

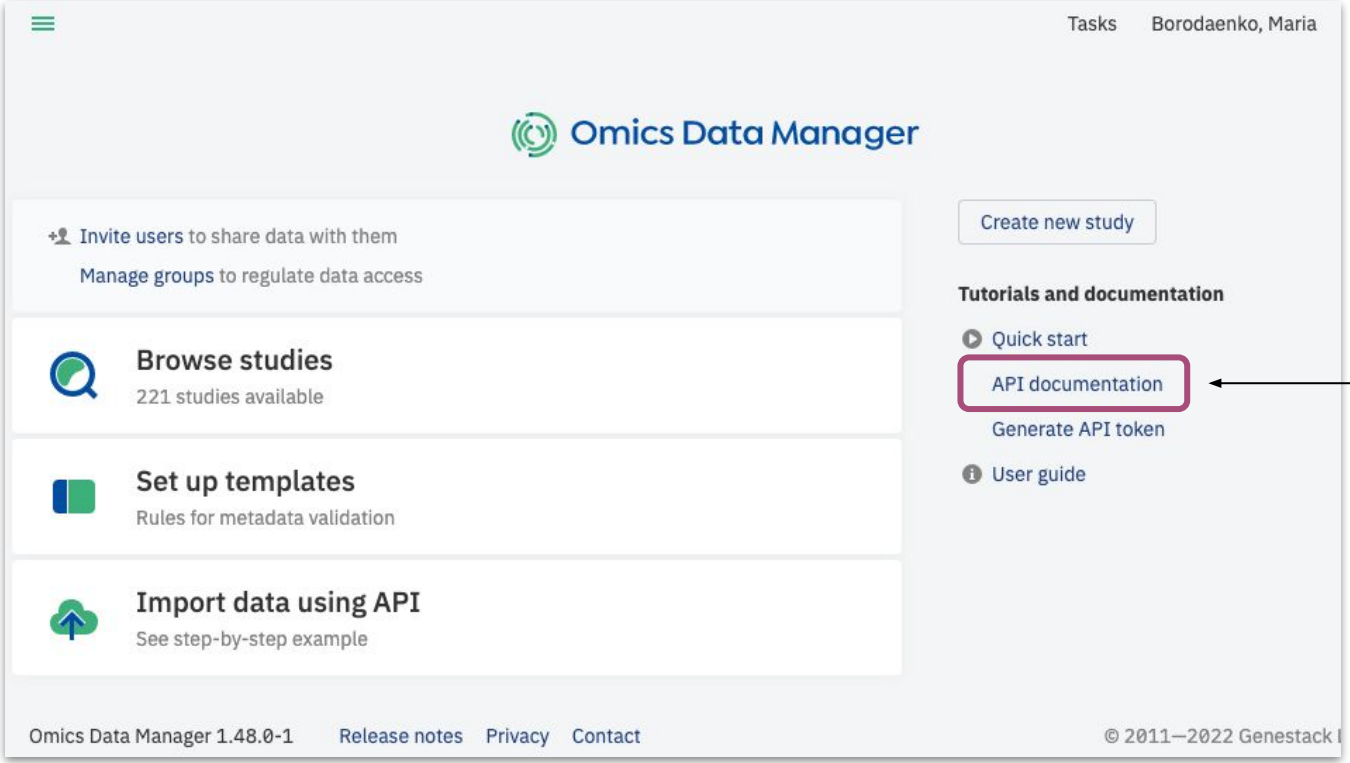
Steps to perform:

- Create a study,
- Create sample group*,
- Link the study and the samples,
- Create an expression file,
- Link the expression to the samples.



*a group of objects from 1 file

Navigating to Swagger



Open Swagger

Navigate to Jobs

Import/curate data

- integrationCurator - API endpoints to link experimental data/metadata to samples, and samples to studies
- studyCurator - API endpoints to add, delete and update studies and study metadata
- sampleCurator - API endpoints to add, delete and update sample
- libraryCurator - API endpoints to add, delete and update libraries
- preparationCurator - API endpoints to add, delete and update pr
- expressionCurator - API endpoints to add, delete and update exp
- variantCurator - API endpoints to add, delete and update variant
- flowCytometryCurator - API endpoints to add, delete and update
- job - Experimental API endpoints to run asynchronous jobs (impo
- tasks - API endpoints to work with asynchronous tasks

Data import jobs

POST	/import/expression	Import expression data and metadata from GCT and TSV files
POST	/import/flow-cytometry	Import flow-cytometry data and metadata from FACS and TSV files
POST	/import/libraries	Import a group of library metadata objects from a TSV file
POST	/import/preparations	Import a group of preparation metadata objects from a TSV file
POST	/import/samples	Import a group of sample metadata objects from a TSV file
POST	/import/study	Import study metadata from a TSV file
POST	/import/variant	Import variation data and metadata from VCF and TSV files

Creating an Object

Authorize

Authorize with your token

POST /import/study Import study metadata from a TSV file

When job finishes successfully the following **result** object can be obtained using `GET /job/{id}/output` request:

```
{
    "groupAccession": "GSF1234567"
}
```

Try it out

Click to make it actionable

Parameters

Name	Description
body object *(body)*	Edit Value | Model

```
{
  "source": "HTTP",
  "metadataLink": "https://mybucket.s3.amazonaws.com/my-experiment/my-object-met
  "templateId": "GSF334953"
}
```

200

Response body

```
{
  "jobExecId": 316,
  "startedBy": "Maria Borodaenko",
  "jobName": "IMPORT_STUDY_TSV",
  "status": "STARTING",
  "createTime": "28-10-2022 11:17:08"
}
```

Troubleshooting

In case you got errors 401, that means that there is an issue with your token.

1. Are you authorized? - Check the green lock above
2. Did you fully copied the token? - All symbols including the very first/last one
3. Is the token from an environment you are connecting to? - qa and prod
4. Is your token valid? - Check if it is listed in your profile

In case you got error 403 please check if you are a member of the Curators group. Only curators are able to use POST endpoints

401

Error: response status is 401

Response body

```
{
  "error": {
    "message": "No API token supplied. Check that the request header coincides with Genestack-API-Token"
  }
}
```

Download

Getting Results

The job progress can be track via **GET /{jobExecId}/info.**

The accession of the created object is available via

GET /{jobExecId}/output.

Job operations

GET	/{jobExecId}/info	get information about one particular job execution
GET	/{jobExecId}/output	retrieve job output (result)
PUT	/{jobExecId}/restart	restart stopped (failed) job
PUT	/{jobExecId}/stop	stop running job

Code	Details
200	Response body


```
{
  "jobExecId": 316,
  "startedBy": "Maria Borodaenko",
  "jobName": "IMPORT_STUDY_TSV",
  "status": "COMPLETED",
  "createTime": "28-10-2022 11:17:09",
  "endTime": "28-10-2022 11:17:09"
}
```


Code	
200	Response body


```
{
  "status": "COMPLETED",
  "result": {
    "accession": "GSF601399"
  },
  "errors": []
}
```

Linking Objects

Swagger
Supported by SMARTBEAR

Select a definition: sampleCurator

ODM API default-released

[Base URL: /frontend/rs/genestack/sampleCurator/default-released]
https://qa.magnum.genestack.com/swagger/yaml/sampleCurator.yaml

Navigate to the IntegrationCurator

- expressionUser
- flowCytometryCurator
- flowCytometryUser
- integrationCurator
- integrationUser
- job
- libraryCurator
- libraryUser
- preparationCurator
- preparationUser
- reference-data
- ✓ sampleCurator
- sampleUser
- studyCurator

Linking Study and Samples

Schemes
HTTPS

Authorize

Authorize with your token

Expression integration

Flow Cytometry integration

Library integration

Linkage

Metadata versioning

Omics queries

Expand

POST /integration/link/sample/group/{sou

Preparation integration

Sample integration

POST /integration/link/sample/group/{sourceId}/to/study/{targetId}

Create a link between a group of sample objects and a study

Linking Study and Samples

Sample integration

POST `/integration/link/sample/group/{sourceId}/to/study/{targetId}`

Create a link between a group of sampl objects and a study

Try it out

Click to make it actionable

Create a link between a group of sample objects and a study.

Sample objects of the same group can only be linked to the same study.

Parameters Cancel

Name	Description
sourceId * required string *(path)*	The ID (accession) of the sample group object GSF058583
targetId * required string *(path)*	The ID (accession) of the study object GSF058582

Sample group accession

Study accession

Execute

Checking Results

Curl

```
curl -X 'POST' \
  'https://qa.magnum.genestack.com/frontend/rs/genestack/integrationCurator/default-released/integration/link/sample/group/GSF058583/
  -H 'accept: application/json' \
  -H 'Genestack-API-Token: 80f83d81ee735eccb347bdcdb31f240f1ac4abf3' \
  -d ''
```

Request URL

```
https://qa.magnum.genestack.com/frontend/rs/genestack/integrationCurator/default-released/integration/link/sample/group/GSF058583/to/study/GSF058582
```

Server response

Code	Details
204	Response headers

```
access-control-allow-origin: *
access-control-expose-headers: User-Agent,If-Modified-Since,Cache-Control,Content-Type,Content-Range,Content-Disposition,Range,DNT,X-CustomHeader,Keep-Alive,X-Requested-With
cache-control: no-cache
date: Thu,29 Sep 2022 02:48:29 GMT
server: nginx/1.21.6
```

Responses

Code	Description
204	Link created.
400	Link cannot be created.

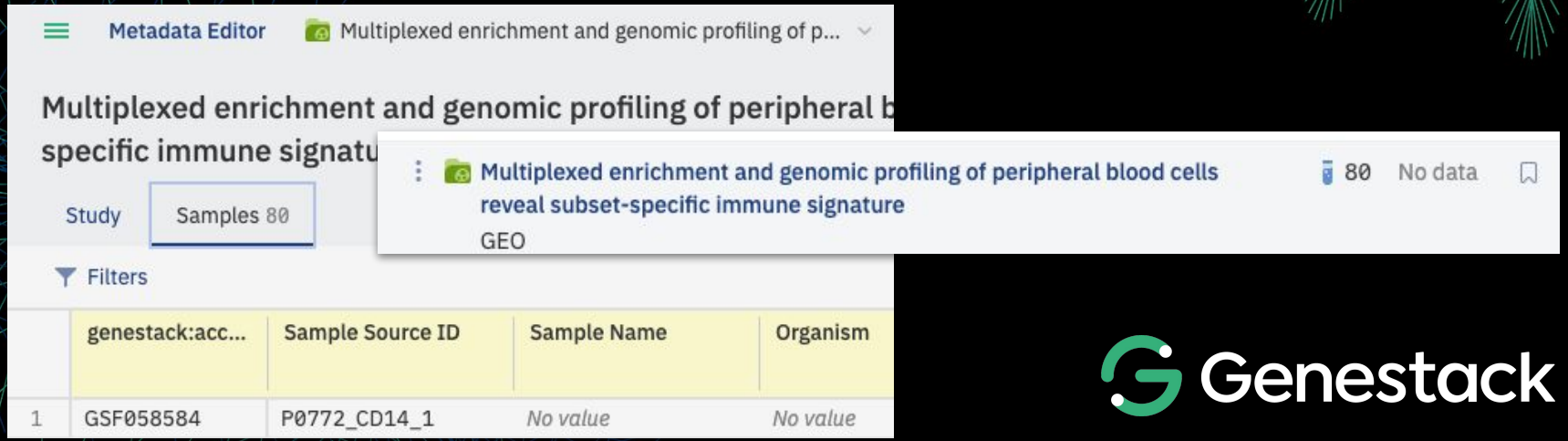
API request

Query results

Codes explanation

Samples are linked to the study:

- Samples displayed in Study Browser,
- Sample tab available in Metadata Editor.



Genestack

Linking Samples and Expression

POST /integration/link/expression/group/{sourceId}/to/sample/group/{targetId}

Create a link between a group of expression objects and a group of sample objects

Create a link between a group of expression objects and a group of sample objects.

An expression object can be linked to one object only. If an expression object is already linked to another object, this link will be deleted and a new link with the specified object will be created.
Expression objects of the same group can only be linked to objects of the same study.

Parameters

Try it out

Click to make it actionable

Name	Description
sourceId * required string *(path)*	The ID (accession) of the group of run-level objects (corresponding to a GCT file) GSF058666
targetId * required string *(path)*	The ID (accession) of the sample group object GSF058583

Expression group accession

Sample group accession

Execute

Curation

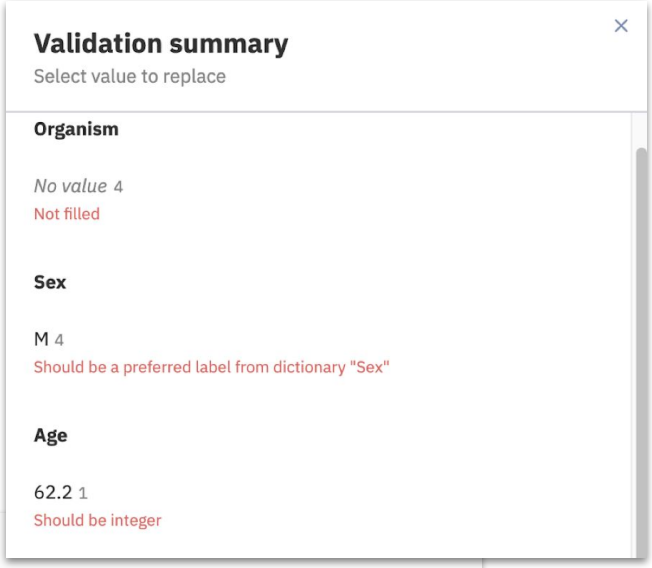
Genestack

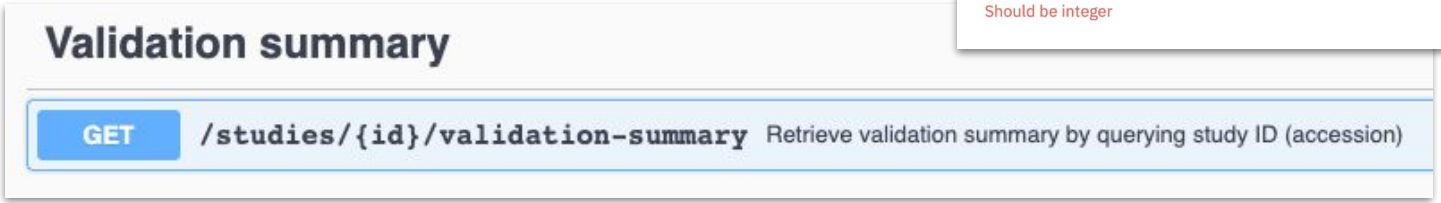
Validation Summary

Validation summary endpoint works similar to the GUI “Metadata Validation Summary”.

It returns all invalid metadata based on the applied template. Currently it is implemented for samples only.

Input for the endpoint is a Study ID (Study accession).





Validation Summary

Output contains:

- “groupAccession” - a group of samples loaded from one file;
- “attributeName” - attributes which have invalid values;
- “attributeInvalidValues” - the invalid values for each attribute;
- “count” - the count of samples with the invalid value for each invalide value;
- “errorType” and “errorMessage” - violated validation rule.

In the example values for an attribute “Organism” are missing for 4 samples in the study.


```
{
  "samples": [
    {
      "groupAccession": "GSF087341",
      "attributes": [
        {
          "attributeName": "Organism",
          "attributeInvalidValues": [
            {
              "value": null,
              "count": 4,
              "errors": [
                {
                  "errorType": "VALUE_NOT_SET",
                  "errorMessage": "The value for required attribute \"Organism\" is not present"
                }
              ]
            }
          ]
        },
```

Patching Metadata

Metadata can be updated with PATCH method. The value can be changed in an existing attribute, or a new attribute can be added, or an existing one can be deleted.

The method should be run for each object separately. Each run creates a new version in Version history.

Name	Description
id * required string (path)	Unique identifier (accession) of the object. GSF052948
body object (body)	Metadata in the form of `{key: value, key2: value2, ...}` Edit Value | Model {"Sex": "Female", "Organism": "Homo sapiens", "New Attribute": "Test", "Species ID": "null"}

PATCH /samples/{id} Update a sample object

Curation Script

Curation script is a python script written for automating frequent operations in the curation process.

The main purpose of the script is to facilitate the curation process, reduce manual operations and reduce errors in metadata curation.

The script could be run on a particular study(ies). Changes done by the script are regulated by JSON file with predefined rules.

```
    "object_type": "sample",
    "raw_keys": ["Cell Type", "Cells"],
    "genestack_key": "Cell Type",
    "rules": {
        "CD19+": "B cell",
        "CD14+": "CD14-positive monocyte",
        "CD8+": "CD8-positive, alpha-beta T cell",
        "CD4+": "CD4-positive, alpha-beta T cell"
    }
},
{
    "object_type": "sample",
    "raw_keys": ["Sex", "Gender"],
    "genestack_key": "Sex",
    "dictionary": "Sex"
```

Task logs

Error Log | Output Log

```
Found 82 files
2022-11-23 03:10:14 UTC - Starting to process 82 files in target folder
Creating new version for file GSF052045 with message Edited using curation script
Processing done in 13.10 seconds
Average editing speed: 7.09 files/s
============================
2022-11-23 03:10:27 UTC - Processed 80 files (average editing speed: 6.92 files/s)
COUNTERS:
 - Cell Type
   - Successful mapping: 80
 - Sex
   - Successful mapping: 80
```

Training

- API access - Getting access token
- Introducing data model and API organisation
- Data loading - Loading and linking test dataset
- **Data versioning - Adding a new version of expression results**
- Data querying - Applying filters to slice and dice your data

Metadata Versioning

It is always the latest version the one see by default in APIs. The list of all available versions of a metadata object can be retrieved by an object ID on the corresponding SPoT.

Response body

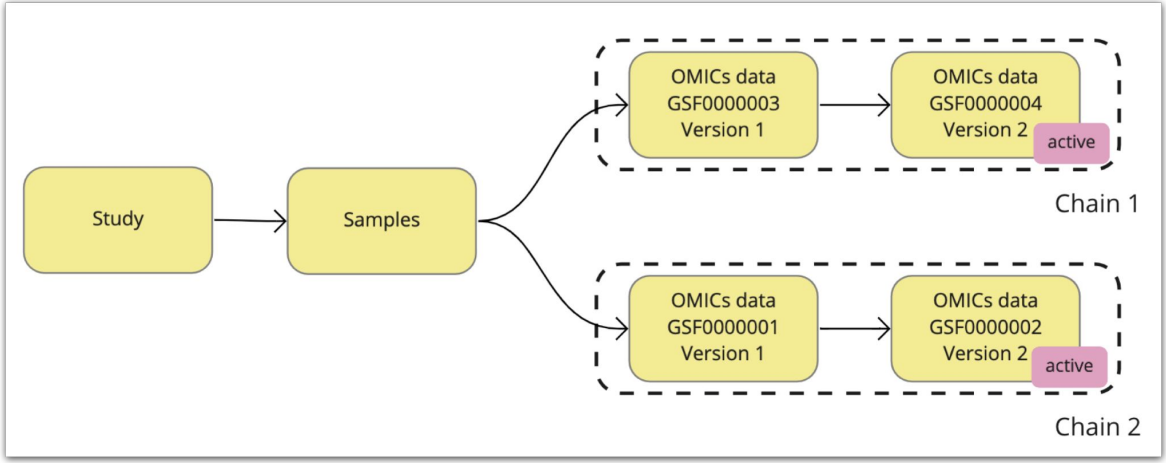

```
[
  {
    "author": "Maria Borodaenko",
    "message": "[Script] Commit created by API call",
    "timestamp": 1669260909268,
    "version": "c2738a64-48e9-45ec-8374-9216b9bc1cbc"
  },
  {
    "author": "Maria Borodaenko",
    "message": "[Script] Commit created by API call",
    "timestamp": 1669260835626,
    "version": "5387c171-6eae-41fa-ae94-7749eb2947db"

    nko",
    ion",
    49,
    e-4e52-8e67-40b862014f07"
  }
]
```


GET	/samples/{id}/versions	Retrieve a list of object versions by ID
GET	/samples/{id}/versions/{version}	Retrieve a single sample object by ID (accession)

Data Versioning

Samples can have multiple expression matrix types associated with them: different instrument type or normalisation method, e.g. TPM, Counts and other. Each expression matrix type can have multiple versions.

Matrix version is generated using different parameters, e.g. different genome version, pipeline options, gene-transcript mapping file.



Data Versioning

Data import jobs

POST /import/expression Import expression data and metadata from GCT

When job finishes successfully the following **result** object can be obtained using GET

```
{
    "groupAccession": "GSF1234567"
}
```

Parameters

Name	Description
body object *(body)*	Edit Value | Model

```
{
  "source": "HTTP",
  "metadataLink": "https://bio-test-data.s3.amazonaws.com/Demo-test/GSE120442/tx.gct.tsv",
  "dataLink": "https://bio-test-data.s3.amazonaws.com/Demo-test/GSE120442/tx.gct",
  "previousVersion": "GSF334953"
}
```

Metadata Editor | Study with 10 samples | Tasks 11 | Maria Borodaenko

Study with 10 samples

Study | Samples 10 | VAR Variant | EXP Expression | FCY Flow Cytometry

Datasets
- TX_10.gct
- TX_10.gct

v. 3 ACTIVE | v. 2 | v. 1

Metadata is valid

genestack:accession	GSF1126039
Experimental Platform	Illumina HiSeq 2500
Data Processing Method	
Scale	linear
Genome Version	GRCh38

Specify the accession of a previous version of a file during loading a new version.

Data Versioning Querying

Specific versions of omics data files can be queried via the **useVersions** parameter. Different versions of an omics data file are associated via their CHAIN_ID metadata value.

This CHAIN_ID can be supplied to the useVersions parameter along with the version number or specific omics data file accessions to include them in the query.

If nothing is supplied to the useVersions parameter then only the active version (which is usually the last one imported) is queried.

This acts as a filter before the rest of the query is carried out.

Training

- API access - Getting access token
- Introducing data model and API organisation
- Data loading - Loading and linking test dataset
- Data versioning - Adding a new version of expression results
- **Data querying - Applying filters to slice and dice your data**

Data Querying

Omics queries allows user to perform integrative queries: found objects based on the parameters of linked data. E.g. you can find samples suitable for your research based on study metadata, samples metadata, metadata of analysis performed and the gene expression of linked data.

- Omics queries provide also cross-study search.
- Allows to query metadata and data in one request.
- Omics are a part of the Integration Layer.

Omics queries

GET	/omics/expression/data	Retrieve expression data objects by searching across multiple data types
GET	/omics/expression/group	Retrieve group objects by searching across multiple data types
GET	/omics/expression/streamed-data	Stream expression data from a given GCT file
GET	/omics/flow-cytometry/data	Retrieve flow cytometry data objects by searching across multiple data types
GET	/omics/flow-cytometry/group	Retrieve group objects by searching across multiple data types
GET	/omics/samples	Retrieve sample metadata objects by searching across multiple data types
GET	/omics/variant/data	Retrieve variant data objects by searching across multiple data types
GET	/omics/variant/group	Retrieve group objects by searching across multiple data types

Omics queries

1. Get samples from your study

GET /omics/samples

Parameters

Name	Description
studyFilter string (query)	Filter by study metadata (key-value metadata pair(s)). E.g. `"Study Source"=ArrayExpress` "Study Title"="GTEx V7"
pageLimit integer($int32) (query)	How many results to retrieve per page. The default is 2000 10

Execute

studyFilter: "genestack:accession" ="GSF058582"

pageLimit: 10

Omics queries

2. Filter samples by samples metadata

GET /omics/samples

Parameters

Name	Description
studyFilter string (query)	Filter by study metadata (key-value metadata pair(s)). E.g. "Study Source"=ArrayExpress "Study Title"="GTEx V7"

pageLimit integer($int32) (query)	How many results to retrieve per page. The default is 2000 10

Execute

studyFilter:
"genestack:accession" ="GSF058582"

sampleFilter:
"Cell Type"="CD14-positive monocyte"

pageLimit: 10

Omics queries

3. Get expression for filtered samples

GET `/omics/expression/data` Retrieve expression data objects by searching across multiple data types

exQuery
string
(query)

Search for objects linked to expression data via data query (key-value pair(s)). E.g.
`Feature=ENSG00000230368,ENSG00000188976 MinValue=1.50`

Feature=ENSG00000000003.10

Expression metadata and data in the result

Code | Details

200

Response body


```
"Data Processing Method": null,
"Genome Version": "hg19",
"Scale": "Unknown ",
"Expression Source": "test source",
"Normalization Method": "Gene reads",
"Transcriptomics Source": "GTEx",
"Name": null,
"Pipeline ID": null,
"Data Species": null,
"Source ID": "GTEx V7",
"Import Source URL": null,
"Data Files / Processed": null,
"Data Files / Raw": null,
"Run Source ID": "GTEX-111CU-0326-SM-5GZXO"
},
"runId": "7020",
"groupId": "GSF028899",
"gene": "ENSG00000000003.10",
"feature": "ENSG00000000003.10",
"expression": 1248,
"relationships": {
"sample": "GSF014768"
}
},
```


studyFilter:
"genestack:accession" ="GSF058582"

sampleFilter:
"Cell Type"="CD14-positive monocyte"

exQuery:
Feature=TRAF3IP2-AS1

pageLimit: 10

Omics queries

4. Get expression for filtered samples

GET `/omics/expression/data` Retrieve expression data objects by searching across multiple data types

exQuery
string
(query)

Search for objects linked to expression data via data query (key-value pair(s)). E.g.
`Feature=ENSG00000230368,ENSG00000188976 MinValue=1.50`

Feature=ENSG00000000003.10

Expression metadata and data in the result

Code | Details

200

Response body


```
        "Data Processing Method": null,
        "Genome Version": "hg19",
        "Scale": "Unknown ",
        "Expression Source": "test source",
        "Normalization Method": "Gene reads",
        "Transcriptomics Source": "GTEx",
        "Name": null,
        "Pipeline ID": null,
        "Data Species": null,
        "Source ID": "GTEx V7",
        "Import Source URL": null,
        "Data Files / Processed": null,
        "Data Files / Raw": null,
        "Run Source ID": "GTEX-111CU-0326-SM-5GZXO"
      },
      "runId": "7020",
      "groupId": "GSF028899",
      "gene": "ENSG00000000003.10",
      "feature": "ENSG00000000003.10",
      "expression": 1248,
      "relationships": {
        "sample": "GSF014768"
      }
    },
```


studyFilter:
"genestack:accession" ="GSF058582"

sampleFilter:
"Cell Type"="CD14-positive monocyte"

exQuery:
Feature=TRAF3IP2-AS1 MinValue=200

pageLimit: 10

Streamed-Data

Streamed-data endpoint is an effective endpoint for streaming expression from a given .gct file. Expression group accession is a required parameter.

The expression can be filtered by samples and/or libraries metadata. So if you are analysing expression for a subset of samples this endpoint can be your choice.

The output will be in .csv format, not JSON as other endpoints.

GET	/omics/expression/streamed-data	Stream expression data from a given GCT file

Template Attributes in API

Users are able to choose how detailed the metadata will be in response:

- Get all available metadata;
- Get metadata from the template applied for the study;
- Get metadata from the Default template.

By default, all available metadata is returned.

returnedMetadataFields
string
(query)

The parameter defines amount of metadata attributes to return:

1. `minimal_data` - return metadata attributes according to the default template.
2. `extended_data_included` - return metadata attributes according to applied template, if object doesn't have applied template default template will be used. This is the default for User endpoints.
3. `original_data_included` - return all metadata attributes with values and null attributes, if they are present in the applied template. This is the default for Curator endpoints.

Swagger Purpose

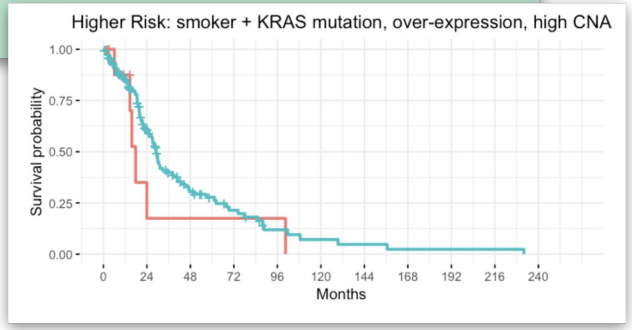
Swagger is a convenient way to document API endpoints and the parameters available for them. It can be considered as a learning platform. But is has its own limitations and times outs, so it is not suitable for the constant usage.

- ✓ Getting understanding of available operations
- ✓ Testing new endpoints
- ✗ Day-to-day usage
- ✗ Integration

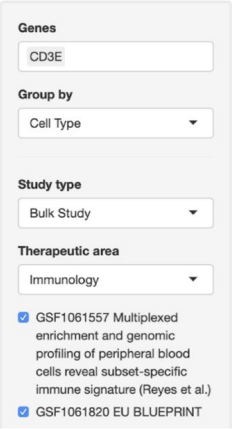
Using Search Services

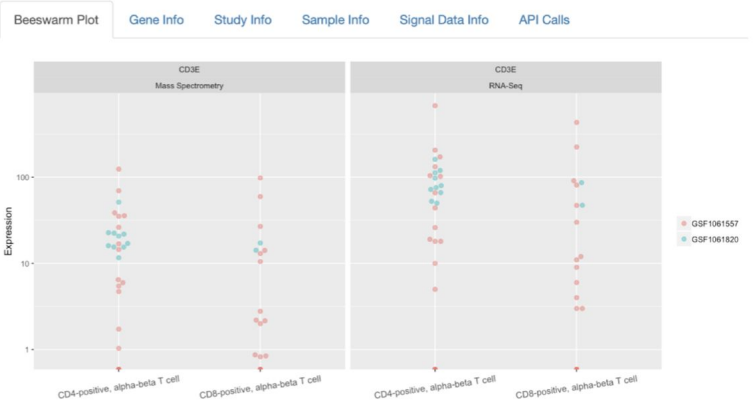
```
search_variant_data(
    study_filter='"Study Title"="TCGA-LUAD"',
    sample_filter='gender=male',
    vx_query='Gene=KRAS'
)
```

```
search_expression_data(
    study_filter='"Study Title"="TCGA-LUAD"',
    sample_filter='gender=male',
    ex_query='Gene=KRAS MinValue=0'
)
```



Example App: Expression Atlas





Congratulations

You are now
an **advanced** ODM user!

Program

- Learning goals
- Training
- **Q&A Session**

Q&A

Genestack

Thank You!

Maria Borodaenko
Senior Solution Consultant
maria.borodaenko@genestack.com

www.genestack.com

Genestack

Unlocking the Power of Life-Sciences Data