BioBlend is a Python library for interacting with the Galaxy API.

BioBlend is supported and tested on:

- Python 3.8 - 3.11
- Galaxy release 19.05 and later.

BioBlend’s goal is to make it easier to script and automate the running of Galaxy analyses and administering of a Galaxy server. In practice, it makes it possible to do things like this:

- Interact with Galaxy via a straightforward API:

  ```python
  from bioblend.galaxy import GalaxyInstance
  gi = GalaxyInstance('<Galaxy IP>', key='your API key')
  libs = gi.libraries.get_libraries()
  gi.workflows.show_workflow('workflow ID')
  wf_invocation = gi.workflows.invoke_workflow('workflow ID', inputs)
  ```

- Interact with Galaxy via an object-oriented API:

  ```python
  from bioblend.galaxy.objects import GalaxyInstance
  gi = GalaxyInstance('URL', 'API_KEY')
  wf = gi.workflows.list()[0]
  hist = gi.histories.list()[0]
  inputs = hist.get_datasets()[:2]
  input_map = dict(zip(wf.input_labels, inputs))
  params = {'Paste1': {'delimiter': 'U'}}
  wf_invocation = wf.invoke(input_map, params=params)
  ```

### 1.1 About the library name

The library was originally called just Blend but we renamed it to reflect more of its domain and make it bit more unique so it can be easier to find. The name was intended to be short and easily pronounceable. In its original implementation, the goal was to provide a lot more support for CloudMan and other integration capabilities, allowing them to be blended together via code. BioBlend fitted the bill.
Stable releases of BioBlend are best installed via pip from PyPI:

```bash
$ python3 -m pip install bioblend
```

Alternatively, the most current source code from our Git repository can be installed with:

```bash
$ python3 -m pip install git+https://github.com/galaxyproject/bioblend
```

After installing the library, you will be able to simply import it into your Python environment with `import bioblend`. For details on the available functionality, see the [API documentation](#).

BioBlend requires a number of Python libraries. These libraries are installed automatically when BioBlend itself is installed, regardless whether it is installed via PyPi or by running `python3 setup.py install` command. The current list of required libraries is always available from `setup.py` in the source code repository.

If you also want to run tests locally, some extra libraries are required. To install them, run:

```bash
$ python3 setup.py test
```
To get started using BioBlend, install the library as described above. Once the library becomes available on the given system, it can be developed against. The developed scripts do not need to reside in any particular location on the system. It is probably best to take a look at the example scripts in docs/examples source directory and browse the API documentation. Beyond that, it’s up to your creativity :).
Anyone interested in contributing or tweaking the library is more than welcome to do so. To start, simply fork the Git repository on Github and start playing with it. Then, issue pull requests.
BioBlend’s API focuses around and matches the services it wraps. Thus, there are two top-level sets of APIs, each corresponding to a separate service and a corresponding step in the automation process. Note that each of the service APIs can be used completely independently of one another.

Effort has been made to keep the structure and naming of those API’s consistent across the library but because they do bridge different services, some discrepancies may exist. Feel free to point those out and/or provide fixes.

For Galaxy, an alternative object-oriented API is also available. This API provides an explicit modeling of server-side Galaxy instances and their relationships, providing higher-level methods to perform operations such as retrieving all datasets for a given history, etc. Note that, at the moment, the oo API is still incomplete, providing access to a more restricted set of Galaxy modules with respect to the standard one.

5.1 Galaxy API

API used to manipulate genomic analyses within Galaxy, including data management and workflow execution.

5.1.1 API documentation for interacting with Galaxy

GalaxyInstance

class bioblend.galaxy.GalaxyInstance(url: str, key: str | None = None, email: str | None = None, password: str | None = None, *, verify: bool = True)

A base representation of a connection to a Galaxy instance, identified by the server URL and user credentials.

After you have created a GalaxyInstance object, access various modules via the class fields. For example, to work with histories and get a list of all the user’s histories, the following should be done:

```python
from bioblend import galaxy

gi = galaxy.GalaxyInstance(url='http://127.0.0.1:8000', key='your_api_key')

hl = gi.histories.get_histories()
```

Parameters

- **url (str)** – A FQDN or IP for a given instance of Galaxy. For example: http://127.0.0.1:8080. If a Galaxy instance is served under a prefix (e.g., http://127.0.0.1:8080/galaxy/), supply the entire URL including the prefix (note that the prefix must end with a slash). If a Galaxy instance has HTTP Basic authentication with username and password, then the credentials should be included in the URL, e.g. http://user:pass@host:port/galaxy/
BioBlend Documentation, Release 1.2.0

- **key** *(str)* – User’s API key for the given instance of Galaxy, obtained from the user preferences. If a key is not supplied, an email address and password must be and the key will automatically be created for the user.

- **email** *(str)* – Galaxy e-mail address corresponding to the user. Ignored if key is supplied directly.

- **password** *(str)* – Password of Galaxy account corresponding to the above e-mail address. Ignored if key is supplied directly.

- **verify** *(bool)* – Whether to verify the server’s TLS certificate

**init**(url: str, key: str | None = None, email: str | None = None, password: str | None = None, *, verify: bool = True) → None

A base representation of a connection to a Galaxy instance, identified by the server URL and user credentials.

After you have created a `GalaxyInstance` object, access various modules via the class fields. For example, to work with histories and get a list of all the user’s histories, the following should be done:

```python
from bioblend import galaxy

gi = galaxy.GalaxyInstance(url='http://127.0.0.1:8000', key='your_api_key')

hl = gi.histories.get_histories()
```

**Parameters**

- **url** *(str)* – A FQDN or IP for a given instance of Galaxy. For example: http://127.0.0.1:8080. If a Galaxy instance is served under a prefix (e.g., http://127.0.0.1:8080/galaxy/), supply the entire URL including the prefix (note that the prefix must end with a slash). If a Galaxy instance has HTTP Basic authentication with username and password, then the credentials should be included in the URL, e.g. http://user:pass@host:port/galaxy/

- **key** *(str)* – User’s API key for the given instance of Galaxy, obtained from the user preferences. If a key is not supplied, an email address and password must be and the key will automatically be created for the user.

- **email** *(str)* – Galaxy e-mail address corresponding to the user. Ignored if key is supplied directly.

- **password** *(str)* – Password of Galaxy account corresponding to the above e-mail address. Ignored if key is supplied directly.

- **verify** *(bool)* – Whether to verify the server’s TLS certificate
**Config**

Contains possible interaction dealing with Galaxy configuration.

```python
class bioblend.galaxy.config.ConfigClient(galaxy_instance: GalaxyInstance)
```

A generic Client interface defining the common fields.

All clients *must* define the following field (which will be used as part of the URL composition (e.g., `<galaxy_instance>/api/libraries`): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

**get_config() → dict**

Get a list of attributes about the Galaxy instance. More attributes will be present if the user is an admin.

**Return type**

```
list
```

**Returns**

A list of attributes. For example:

```python
{
'allow_library_path_paste': False,
'allow_user_creation': True,
'allow_user_dataset_purge': True,
'allow_user_deletion': False,
'enable_unique_workflow_defaults': False,
'ftp_upload_dir': '/SOMEWHERE/galaxy/ftp_dir',
'ftp_upload_site': 'galaxy.com',
'library_import_dir': 'None',
'logo_url': None,
'support_url': 'https://galaxyproject.org/support',
'terms_url': None,
'user_library_import_dir': None,
'wiki_url': 'https://galaxyproject.org/'
}
```

**get_version() → dict**

Get the current version of the Galaxy instance.

**Return type**

```
dict
```

**Returns**

Version of the Galaxy instance For example:

```python
{'extra': {}, 'version_major': '17.01'}
```

**module: str = 'configuration'**

**reload_toolbox() → None**

Reload the Galaxy toolbox (but not individual tools)

**Return type**

```
None
```

**Returns**

```
None
```
**whoami() → dict**

Return information about the current authenticated user.

**Return type**

dict

**Returns**

Information about current authenticated user For example:

```json
{
    'active': True,
    'deleted': False,
    'email': 'user@example.org',
    'id': '4aaaa85aacc9caa',
    'last_password_change': '2021-07-29T05:34:54.632345',
    'model_class': 'User',
    'username': 'julia'
}
```

---

**Datasets**

Contains possible interactions with the Galaxy Datasets

```python
class bioblend.galaxy.datasets.DatasetClient(galaxy_instance: GalaxyInstance)
    A generic Client interface defining the common fields.
```

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

```python
download_dataset(dataset_id: str, file_path: None = None, use_default_filename: bool = True, require_ok_state: bool = True, maxwait: float = 12000) → bytes
```

```python
download_dataset(dataset_id: str, file_path: str, use_default_filename: bool = True, require_ok_state: bool = True, maxwait: float = 12000) → str
```

Download a dataset to file or in memory. If the dataset state is not ‘ok’, a DatasetStateException will be thrown, unless `require_ok_state=False`.

**Parameters**

- **dataset_id (str)** – Encoded dataset ID
- **file_path (str)** – If this argument is provided, the dataset will be streamed to disk at that path (should be a directory if `use_default_filename=True`). If the `file_path` argument is not provided, the dataset content is loaded into memory and returned by the method (Memory consumption may be heavy as the entire file will be in memory).
- **use_default_filename (bool)** – If True, the exported file will be saved as `file_path/%s`, where %s is the dataset name. If False, `file_path` is assumed to contain the full file path including the filename.
- **require_ok_state (bool)** – If False, datasets will be downloaded even if not in an ‘ok’ state, issuing a DatasetStateWarning rather than raising a DatasetStateException.
- **maxwait (float)** – Total time (in seconds) to wait for the dataset state to become terminal. If the dataset state is not terminal within this time, a DatasetTimeoutException will be thrown.
Return type
bytes or str

Returns
If a file_path argument is not provided, returns the file content. Otherwise returns the local path of the downloaded file.

get_datasets(limit: int = 500, offset: int = 0, name: str | None = None, extension: str | List[str] | None = None, state: str | List[str] | None = None, visible: bool | None = None, deleted: bool | None = None, purged: bool | None = None, tool_id: str | None = None, tag: str | None = None, history_id: str | None = None, create_time_min: str | None = None, create_time_max: str | None = None, update_time_min: str | None = None, update_time_max: str | None = None, order: str = 'create_time-dsc') → List[Dict[str, Any]]

Get the latest datasets, or select another subset by specifying optional arguments for filtering (e.g. a history ID).

Since the number of datasets may be very large, limit and offset parameters are required to specify the desired range.

If the user is an admin, this will return datasets for all the users, otherwise only for the current user.

Parameters

• limit (int) – Maximum number of datasets to return.
• offset (int) – Return datasets starting from this specified position. For example, if limit is set to 100 and offset to 200, datasets 200-299 will be returned.
• name (str) – Dataset name to filter on.
• extension (str or list of str) – Dataset extension (or list of extensions) to filter on.
• state (str or list of str) – Dataset state (or list of states) to filter on.
• visible (bool) – Optionally filter datasets by their visible attribute.
• deleted (bool) – Optionally filter datasets by their deleted attribute.
• purged (bool) – Optionally filter datasets by their purged attribute.
• tool_id (str) – Tool ID to filter on.
• tag (str) – Dataset tag to filter on.
• history_id (str) – Encoded history ID to filter on.
• create_time_min (str) – Show only datasets created after the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
• create_time_max (str) – Show only datasets created before the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
• update_time_min (str) – Show only datasets last updated after the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
• update_time_max (str) – Show only datasets last updated before the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
• order (str) – One or more of the following attributes for ordering datasets: create_time (default), extension, hid, history_id, name, update_time. Optionally, -asc or -dsc (default) can be appended for ascending and descending order respectively. Multiple attributes can be stacked as a comma-separated list of values, e.g. create_time-asc, hid-dsc.
Return type
dict

Returns
Details of the updated dataset

Note: This method works only on Galaxy 19.05 or later.

show_dataset(dataset_id: str, deleted: bool = False, hda_ldda: Literal['hda', 'ldda'] = 'hda') → Dict[str, Any]
Get details about a given dataset. This can be a history or a library dataset.

Parameters
• dataset_id (str) – Encoded dataset ID
• deleted (bool) – Whether to return results for a deleted dataset
• hda_ldda (str) – Whether to show a history dataset ('hda' - the default) or library dataset ('ldda').

Return type
dict

Returns
Information about the HDA or LDDA

update_permissions(dataset_id: str, access_ids: list | None = None, manage_ids: list | None = None, modify_ids: list | None = None) → dict
Set access, manage or modify permissions for a dataset to a list of roles.

Parameters
• dataset_id (str) – dataset ID
• access_ids (list) – role IDs which should have access permissions for the dataset.
• manage_ids (list) – role IDs which should have manage permissions for the dataset.
• modify_ids (list) – role IDs which should have modify permissions for the dataset.

Return type
dict
Returns
Current roles for all available permission types.

Note: This method works only on Galaxy 19.05 or later.

```
```

Wait until a dataset is in a terminal state.

**Parameters**
- `dataset_id (str)` – dataset ID
- `maxwait (float)` – Total time (in seconds) to wait for the dataset state to become terminal. If the dataset state is not terminal within this time, a DatasetTimeoutException will be raised.
- `interval (float)` – Time (in seconds) to wait between 2 consecutive checks.
- `check (bool)` – Whether to check if the dataset terminal state is ‘ok’.

**Return type**
dict

**Returns**
Details of the given dataset.

*exception bioblend.galaxy.datasets.DatasetStateException*

*exception bioblend.galaxy.datasets.DatasetStateWarning*

*exception bioblend.galaxy.datasets.DatasetTimeoutException*

---

**Dataset collections**

```
class bioblend.galaxy.dataset_collections.CollectionDescription(name: str, type: str = 'list',
    elements: List[CollectionElement] | SimpleElement] | Dict[str, Any] | None = None)

to_dict() → Dict[str, str | List]
```

```
class bioblend.galaxy.dataset_collections.CollectionElement(name: str, type: str = 'list', elements:
    List[CollectionElement] | SimpleElement] | Dict[str, Any] | None = None)

to_dict() → Dict[str, str | List]
```

```
class bioblend.galaxy.dataset_collections.DatasetCollectionClient(galaxy_instance: GalaxyInstance)
```

A generic Client interface defining the common fields.
All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

download_dataset_collection(dataset_collection_id: str, file_path: str) → Dict[str, Any]

Download a history dataset collection as an archive.

Parameters

• dataset_collection_id (str) – Encoded dataset collection ID
• file_path (str) – The path to which the archive will be downloaded

Return type

dict

Returns

Information about the downloaded archive.

Note: This method downloads a zip archive for Galaxy 21.01 and later. For earlier versions of Galaxy this method downloads a tgz archive.

gi: GalaxyInstance

module: str = 'dataset_collections'

show_dataset_collection(dataset_collection_id: str, instance_type: str = 'history') → Dict[str, Any]

Get details of a given dataset collection of the current user

Parameters

• dataset_collection_id (str) – dataset collection ID
• instance_type (str) – instance type of the collection - ‘history’ or ‘library’

Return type

dict

Returns

element view of the dataset collection


Wait until all or a specified proportion of elements of a dataset collection are in a terminal state.

Parameters

• dataset_collection_id (str) – dataset collection ID
• maxwait (float) – Total time (in seconds) to wait for the dataset states in the dataset collection to become terminal. If not all datasets are in a terminal state within this time, a DatasetCollectionTimeoutException will be raised.
• interval (float) – Time (in seconds) to wait between two consecutive checks.
• proportion_complete (float) – Proportion of elements in this collection that have to be in a terminal state for this method to return. Must be a number between 0 and 1. For example: if the dataset collection contains 2 elements, and proportion_complete=0.5 is specified, then wait_for_dataset_collection will return as soon as 1 of the 2 datasets is in a terminal state. Default is 1, i.e. all elements must complete.
• **check** (*bool*) – Whether to check if all the terminal states of datasets in the dataset collection are 'ok'. This will raise an Exception if a dataset is in a terminal state other than 'ok'.

**Return type**
dict

**Returns**
Details of the given dataset collection.

class bioblend.galaxy.dataset_collections.
  HistoryDatasetCollectionElement(*name: str, id: str*)
class bioblend.galaxy.dataset_collections.
  HistoryDatasetElement(*name: str, id: str*)
class bioblend.galaxy.dataset_collections.
  LibraryDatasetElement(*name: str, id: str*)

### Datatypes

Contains possible interactions with the Galaxy Datatype

class bioblend.galaxy.datatypes.
  DatatypesClient(*galaxy_instance: GalaxyInstance*)
  A generic Client interface defining the common fields.

  All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

get_datatypes(*extension_only: bool = False, upload_only: bool = False*) → List[str]
  Get the list of all installed datatypes.

  **Parameters**

  • **extension_only** (*bool*) – Return only the extension rather than the datatype name

  • **upload_only** (*bool*) – Whether to return only datatypes which can be uploaded

  **Return type**
  list

  **Returns**

  A list of datatype names. For example:

  ```python
  ['snpmatrix',
   'snptest',
   'tabular',
   'taxonomy',
   'twobit',
   'txt',
   'vcf',
   'wig',
   'xgmm1',
   'xml']
  ```

get_sniffers() → List[str]
  Get the list of all installed sniffers.

5.1. Galaxy API
Return type
list

Returns
A list of sniffer names. For example:

```
['galaxy.datatypes.tabular:Vcf',
 'galaxy.datatypes.binary:TwoBit',
 'galaxy.datatypes.binary:Bam',
 'galaxy.datatypes.binary:Sff',
 'galaxy.datatypes.xml:Phyloxml',
 'galaxy.datatypes.xml:GenericXml',
 'galaxy.datatypes.sequence:Maf',
 'galaxy.datatypes.sequence:Lav',
 'galaxy.datatypes.sequence:csFasta']
```

```
module:  str = 'datatypes'
```

---

**Folders**

Contains possible interactions with the Galaxy library folders.

```python
class bioblend.galaxy.folders.FoldersClient(galaxy_instance: GalaxyInstance)
A generic Client interface defining the common fields.
```

All clients **must** define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

`create_folder(parent_folder_id: str, name: str, description: str | None = None) → Dict[str, Any]`

Create a folder.

**Parameters**

- **parent_folder_id** (str) – Folder's description
- **name** (str) – name of the new folder
- **description** (str) – folder's description

**Return type**
dict

**Returns**
details of the updated folder

`delete_folder(folder_id: str, undelete: bool = False) → Dict[str, Any]`

Marks the folder with the given `id` as `deleted` (or removes the `deleted` mark if the `undelete` param is True).

**Parameters**

- **folder_id** (str) – the folder’s encoded id, prefixed by ‘F’
- **undelete** (bool) – If set to True, the folder will be undeleted (i.e. the `deleted` mark will be removed)

**Returns**
detailed folder information
Return type
dict

get_permissions(folder_id: str, scope: Literal['current', 'available'] = 'current') → Dict[str, Any]
Get the permissions of a folder.

Parameters

• folder_id (str) – the folder’s encoded id, prefixed by ‘F’
• scope (str) – scope of permissions, either ‘current’ or ‘available’

Return type
dict

Returns
dictionary including details of the folder permissions

module: str = 'folders'

Set the permissions of a folder.

Parameters

• folder_id (str) – the folder’s encoded id, prefixed by ‘F’
• action (str) – action to execute, only “set_permissions” is supported.
• add_ids (list of str) – list of role IDs which can add datasets to the folder
• manage_ids (list of str) – list of role IDs which can manage datasets in the folder
• modify_ids (list of str) – list of role IDs which can modify datasets in the folder

Return type
dict

Returns
dictionary including details of the folder

show_folder(folder_id: str, contents: bool = False) → Dict[str, Any]
Display information about a folder.

Parameters

• folder_id (str) – the folder’s encoded id, prefixed by ‘F’
• contents (bool) – True to get the contents of the folder, rather than just the folder details.

Return type
dict

Returns
dictionary including details of the folder

update_folder(folder_id: str, name: str | None = None) → Dict[str, Any]
Update folder information.

Parameters

• folder_id (str) – the folder’s encoded id, prefixed by ‘F’
• name (str) – name of the new folder
• description (str) – folder’s description

Return type
dict

Returns
details of the updated folder

Forms

Contains possible interactions with the Galaxy Forms

class bioblend.galaxy.forms.FormsClient(galaxy_instance: GalaxyInstance)

A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

create_form(form_xml_text: str) → List[Dict[str, Any]]

Create a new form.

Parameters
form_xml_text (str) – Form xml to create a form on galaxy instance

Return type
list of dicts

Returns
List with a single dictionary describing the created form

get_forms() → List[Dict[str, Any]]

Get the list of all forms.

Return type
list

Returns
Displays a collection (list) of forms. For example:

```json
[{
    'id': 'f2db41e1fa331b3e',
    'model_class': 'FormDefinition',
    'name': 'First form',
    'url': '/api/forms/f2db41e1fa331b3e'},

  {
    'id': 'ebfb8f50c6abde6d',
    'model_class': 'FormDefinition',
    'name': 'second form',
    'url': '/api/forms/ebfb8f50c6abde6d'}]
```

module: str = 'forms'

show_form(form_id: str) → Dict[str, Any]

Get details of a given form.

Parameters
form_id (str) – Encoded form ID
**Return type**

dict

**Returns**

A description of the given form. For example:

```json
{
    'desc': 'here it is ',
    'fields': [],
    'form_definition_current_id': 'f2db41e1fa331b3e',
    'id': 'f2db41e1fa331b3e',
    'layout': [],
    'model_class': 'FormDefinition',
    'name': 'First form',
    'url': '/api/forms/f2db41e1fa331b3e'
}
```

---

**FTP files**

Contains possible interactions with the Galaxy FTP Files

```python
class bioblend.galaxy.ftpfiles.FTPFilesClient(galaxy_instance: GalaxyInstance)
    A generic Client interface defining the common fields.
    All clients must define the following field (which will be used as part of the URL composition
    (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

def get_ftp_files(deleted: bool = False) -> List[dict]
    Get a list of local files.
    Parameters
    deleted (bool) – Whether to include deleted files
    Return type
    list
    Returns
    A list of dicts with details on individual files on FTP
```

---

**Genomes**

Contains possible interactions with the Galaxy Histories

```python
class bioblend.galaxy.genomes.GenomeClient(galaxy_instance: GalaxyInstance)
    A generic Client interface defining the common fields.
    All clients must define the following field (which will be used as part of the URL composition
    (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...
```

---

5.1. Galaxy API
BioBlend Documentation, Release 1.2.0

**get_genomes() → list**

Returns a list of installed genomes

**Return type**

list

**Returns**

List of installed genomes

**install_genome(func: Literal['download', 'index'] = 'download', source: str | None = None, dbkey: str | None = None, ncbi_name: str | None = None, ensembl_dbkey: str | None = None, url_dbkey: str | None = None, indexers: list | None = None) → Dict[str, Any]**

Download and/or index a genome.

**Parameters**

- **func (str)** – Allowed values: ‘download’, Download and index; ‘index’, Index only
- **source (str)** – Data source for this build. Can be: UCSC, Ensembl, NCBI, URL
- **dbkey (str)** – DB key of the build to download, ignored unless ‘UCSC’ is specified as the source
- **ncbi_name (str)** – NCBI’s genome identifier, ignored unless NCBI is specified as the source
- **ensembl_dbkey (str)** – Ensembl’s genome identifier, ignored unless Ensembl is specified as the source
- **url_dbkey (str)** – DB key to use for this build, ignored unless URL is specified as the source
- **indexers (list)** – POST array of indexers to run after downloading (indexers[] = first, indexers[] = second, ...)

**Return type**

dict

**Returns**


**module: str = 'genomes'**

**show_genome(id: str, num: str | None = None, chrom: str | None = None, low: str | None = None, high: str | None = None) → Dict[str, Any]**

Returns information about build <id>

**Parameters**

- **id (str)** – Genome build ID to use
- **num (str)** – num
- **chrom (str)** – chrom
- **low (str)** – low
- **high (str)** – high

**Return type**

dict

**Returns**

Information about the genome build
Groups

Contains possible interactions with the Galaxy Groups

class bioblend.galaxy.groups.GroupsClient(galaxy_instance: GalaxyInstance)

A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

add_group_role(group_id: str, role_id: str) → Dict[str, Any]
Add a role to the given group.

Parameters
• group_id (str) – Encoded group ID
• role_id (str) – Encoded role ID to add to the group

Return type
dict

Returns
Added group role’s info

add_group_user(group_id: str, user_id: str) → Dict[str, Any]
Add a user to the given group.

Parameters
• group_id (str) – Encoded group ID
• user_id (str) – Encoded user ID to add to the group

Return type
dict

Returns
Added group user’s info

create_group(group_name: str, user_ids: List[str] | None = None, role_ids: List[str] | None = None) → List[Dict[str, Any]]
Create a new group.

Parameters
• group_name (str) – A name for the new group
• user_ids (list) – A list of encoded user IDs to add to the new group
• role_ids (list) – A list of encoded role IDs to add to the new group

Return type
list

Returns
A (size 1) list with newly created group details, like:

```
[{
    'id': '7c9636938c3e83bf',
    'model_class': 'Group',
    'name': 'My Group Name',
    'url': '/api/groups/7c9636938c3e83bf'}]
```
**delete_group_role**(*group_id*: *str*, *role_id*: *str*) → Dict[*str*, *Any*]

Remove a role from the given group.

**Parameters**
- **group_id** (*str*) – Encoded group ID
- **role_id** (*str*) – Encoded role ID to remove from the group

**Return type**
dict

**Returns**
The role which was removed

**delete_group_user**(*group_id*: *str*, *user_id*: *str*) → Dict[*str*, *Any*]

Remove a user from the given group.

**Parameters**
- **group_id** (*str*) – Encoded group ID
- **user_id** (*str*) – Encoded user ID to remove from the group

**Return type**
dict

**Returns**
The user which was removed

**get_group_roles**(*group_id*: *str*) → List[Dict[*str*, *Any*]]

Get the list of roles associated to the given group.

**Parameters**
- **group_id** (*str*) – Encoded group ID

**Return type**
list of dicts

**Returns**
List of group roles’ info

**get_group_users**(*group_id*: *str*) → List[Dict[*str*, *Any*]]

Get the list of users associated to the given group.

**Parameters**
- **group_id** (*str*) – Encoded group ID

**Return type**
list of dicts

**Returns**
List of group users’ info

**get_groups**() → List[Dict[*str*, *Any*]]

Get all (not deleted) groups.

**Return type**
list

**Returns**
A list of dicts with details on individual groups. For example:
module: str = 'groups'

show_group(group_id: str) → Dict[str, Any]

Get details of a given group.

Parameters

- group_id (str) – Encoded group ID

Return type
dict

Returns

A description of group For example:

```
{
'id': '33abac023ff186c2',
'model_class': 'Group',
'name': 'Listeria',
'url': '/api/groups/33abac023ff186c2',
}
```

update_group(group_id: str, group_name: str | None = None, user_ids: List[str] | None = None, role_ids: List[str] | None = None) → None

Update a group.

Parameters

- group_id (str) – Encoded group ID
- group_name (str) – A new name for the group. If None, the group name is not changed.
- user_ids (list) – New list of encoded user IDs for the group. It will substitute the previous list of users (with [] if not specified)
- role_ids (list) – New list of encoded role IDs for the group. It will substitute the previous list of roles (with [] if not specified)

Return type
None

Returns
None
Histories

Contains possible interactions with the Galaxy Histories

```python
class bioblend.galaxy.histories.HistoryClient(galaxy_instance: GalaxyInstance)

A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition
(e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries'
| 'histories' | ...)

copy_content(history_id: str, content_id: str, source: Literal['hda', 'hdca', 'library', 'library_folder'] = 'hda') → Dict[str, Any]

Copy existing content (e.g. a dataset) to a history.

Parameters

• **history_id** (str) – ID of the history to which the content should be copied
• **content_id** (str) – ID of the content to copy
• **source** (str) – Source of the content to be copied: ‘hda’ (for a history dataset, the default),
  ‘hdca’ (for a dataset collection), ‘library’ (for a library dataset) or ‘library_folder’ (for all
datasets in a library folder).

Return type
dict

Returns
Information about the copied content

copy_dataset(history_id: str, dataset_id: str, source: Literal['hda', 'library', 'library_folder'] = 'hda') → Dict[str, Any]

Copy a dataset to a history.

Parameters

• **history_id** (str) – history ID to which the dataset should be copied
• **dataset_id** (str) – dataset ID
• **source** (str) – Source of the dataset to be copied: ‘hda’ (the default), ‘library’ or ‘li-
  brary_folder’

Return type
dict

Returns
Information about the copied dataset

create_dataset_collection(history_id: str, collection_description: CollectionDescription | Dict[str,
  Any], copy_elements: bool = True) → Dict[str, Any]

Create a new dataset collection

Parameters

• **history_id** (str) – Encoded history ID
• **collection_description** (bioblend.galaxy.dataset_collections.
  CollectionDescription) – a description of the dataset collection For example:
```
{
  'collection_type': 'list',
  'element_identifiers': [
    {'id': 'f792763bee8d277a',
     'name': 'element 1',
     'src': 'hda'},
    {'id': 'f792763bee8d277a',
     'name': 'element 2',
     'src': 'hda'}],
  'name': 'My collection list'
}
```

- **copy_elements** (*bool*) – Whether to make a copy of the elements of the collection being created

  **Return type**
  
  dict

  **Returns**
  
  Information about the new HDCA

  **create_history**(*name: str | None = None*) → Dict[str, Any]

  Create a new history, optionally setting the name.

  **Parameters**
  
  - **name** (*str*) – Optional name for new history

  **Return type**
  
  dict

  **Returns**
  
  Dictionary containing information about newly created history

  **create_history_tag**(*history_id: str, tag: str*) → Dict[str, Any]

  Create history tag

  **Parameters**
  
  - **history_id** (*str*) – Encoded history ID
  - **tag** (*str*) – Add tag to history

  **Return type**
  
  dict

  **Returns**
  
  A dictionary with information regarding the tag. For example:

  ```
  {'id': 'f792763bee8d277a',
   'model_class': 'HistoryTagAssociation',
   'user_tname': 'NGS_PE_RUN',
   'user_value': None}
  ```

  **delete_dataset**(*history_id: str, dataset_id: str, purge: bool = False*) → None

  Mark corresponding dataset as deleted.

  **Parameters**
  
  - **history_id** (*str*) – Encoded history ID
  - **dataset_id** (*str*) – Encoded dataset ID
  - **purge** (*bool*) – if True, also purge (permanently delete) the dataset
Delete a history.

Parameters

• `history_id` (`str`): Encoded history ID

• `purge` (`bool`): if True, also purge (permanently delete) the history

Return type
dict

Returns
An error object if an error occurred or a dictionary containing: `id` (the encoded id of the history), `deleted` (if the history was marked as deleted), `purged` (if the history was purged).

Note: The `purge` option works only if the Galaxy instance has the `allow_user_dataset_purge` option set to true in the `config/galaxy.yml` configuration file.

Download a history export archive. Use `export_history()` to create an export.

Parameters

• `history_id` (`str`): history ID

• `jeha_id` (`str`): jeha ID (this should be obtained via `export_history()`)

• `outf` (`IO[bytes]`): output file object, open for writing in binary mode

• `chunk_size` (`int`): how many bytes at a time should be read into memory

Return type
None

Returns
None
**export_history** *(history_id: str, gzip: bool = True, include_hidden: bool = False, include_deleted: bool = False, wait: bool = False, maxwait: float | None = None) → str*

Start a job to create an export archive for the given history.

**Parameters**

- **history_id** *(str)* – history ID
- **gzip** *(bool)* – create .tar.gz archive if True, else .tar
- **include_hidden** *(bool)* – whether to include hidden datasets in the export
- **include_deleted** *(bool)* – whether to include deleted datasets in the export
- **wait** *(bool)* – if True, block until the export is ready; else, return immediately
- **maxwait** *(float | None)* – Total time (in seconds) to wait for the export to become ready. When set, implies that wait is True.

**Return type**

str

**Returns**

ejeha_id of the export, or empty if wait is False and the export is not ready.

**get_extra_files** *(history_id: str, dataset_id: str) → List[str]*

Get extra files associated with a composite dataset, or an empty list if there are none.

**Parameters**

- **history_id** *(str)* – history ID
- **dataset_id** *(str)* – dataset ID

**Return type**

list

**Returns**

List of extra files

**get_histories** *(history_id: str | None = None, name: str | None = None, deleted: bool = False, published: bool | None = None, slug: str | None = None, create_time_min: str | None = None, create_time_max: str | None = None, update_time_min: str | None = None, update_time_max: str | None = None, all: bool | None = False) → List[Dict[str, Any]]*

Get all histories, or select a subset by specifying optional arguments for filtering (e.g. a history name).

**Parameters**

- **name** *(str)* – History name to filter on.
- **deleted** *(bool)* – whether to filter for the deleted histories (True) or for the non-deleted ones (False)
- **published** *(bool or None)* – whether to filter for the published histories (True) or for the non-published ones (False). If not set, no filtering is applied. Note the filtering is only applied to the user’s own histories; to access all histories published by any user, use the get_published_histories method.
- **slug** *(str)* – History slug to filter on
- **create_time_min** *(str)* – Return histories created after the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
- **create_time_max** *(str)* – Return histories created before the provided time and date, which should be formatted as YYYY-MM-DDTHH-MM-SS.
• **update_time_min** (*str*) – Return histories last updated after the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

• **update_time_max** (*str*) – Return histories last updated before the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

• **all** (*bool*) – Whether to include histories from other users. This parameter works only on Galaxy 20.01 or later and can be specified only if the user is a Galaxy admin.

**Return type**
list

**Returns**
List of history dicts.

Changed in version 0.17.0: Using the deprecated `history_id` parameter now raises a `ValueError` exception.

**get_most_recently_used_history**() → Dict[*str*, *Any*]
Returns the current user’s most recently used history (not deleted).

**Return type**
dict

**Returns**
History representation

**get_published_histories**(name: *str* | *None* = *None*, deleted: *bool* = *False*, slug: *str* | *None* = *None*,
create_time_min: *str* | *None* = *None*, create_time_max: *str* | *None* = *None*,
update_time_min: *str* | *None* = *None*, update_time_max: *str* | *None* = *None*) → List[Dict[*str*, *Any*]]

Get all published histories (by any user), or select a subset by specifying optional arguments for filtering (e.g. a history name).

**Parameters**

• **name** (*str*) – History name to filter on.

• **deleted** (*bool*) – whether to filter for the deleted histories (True) or for the non-deleted ones (False)

• **slug** (*str*) – History slug to filter on

• **create_time_min** (*str*) – Return histories created after the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

• **create_time_max** (*str*) – Return histories created before the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

• **update_time_min** (*str*) – Return histories last updated after the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

• **update_time_max** (*str*) – Return histories last updated before the provided time and date, which should be formatted as *YYYY-MM-DDTHH-MM-SS*.

**Return type**
list

**Returns**
List of history dicts.
get_status(history_id: str) → Dict[str, Any]
Returns the state of this history

Parameters
history_id (str) – Encoded history ID

Return type
dict

Returns
A dict documenting the current state of the history. Has the following keys: ‘state’ = This is
the current state of the history, such as ok, error, new etc. ‘state_details’ = Contains individual
statistics for various dataset states. ‘percent_complete’ = The overall number of datasets
processed to completion.

gi: GalaxyInstance

import_history(file_path: str | None = None, url: str | None = None) → Dict[str, Any]
Import a history from an archive on disk or a URL.

Parameters
• file_path (str) – Path to exported history archive on disk.
• url (str) – URL for an exported history archive

Return type
dict

Returns
Dictionary containing information about the imported history

module: str = 'histories'

open_history(history_id: str) → None
Open Galaxy in a new tab of the default web browser and switch to the specified history.

Parameters
history_id (str) – ID of the history to switch to

Return type
NoneType

Returns
None

Warning: After opening the specified history, all previously opened Galaxy tabs in the browser ses-
sion will have the current history changed to this one, even if the interface still shows another history.
Refreshing any such tab is recommended.

show_dataset(history_id: str, dataset_id: str) → Dict[str, Any]
Get details about a given history dataset.

Parameters
• history_id (str) – Encoded history ID
• dataset_id (str) – Encoded dataset ID

Return type
dict
**Returns**
Information about the dataset

`show_dataset_collection(history_id: str, dataset_collection_id: str) → Dict[str, Any]`
Get details about a given history dataset collection.

**Parameters**

- `history_id (str)` – Encoded history ID
- `dataset_collection_id (str)` – Encoded dataset collection ID

**Return type**
dict

**Returns**
Information about the dataset collection

`show_dataset_provenance(history_id: str, dataset_id: str, follow: bool = False) → Dict[str, Any]`
Get details related to how dataset was created (id, job_id, tool_id, stdout, stderr, parameters, inputs, etc...).

**Parameters**

- `history_id (str)` – Encoded history ID
- `dataset_id (str)` – Encoded dataset ID
- `follow (bool)` – If True, recursively fetch dataset provenance information for all inputs and their inputs, etc.

**Return type**
dict

**Returns**
Dataset provenance information For example:

```python
{'id': '6fbd9b2274c62ebe',
 'job_id': '5471ba76f274f929',
 'parameters': {'chromInfo': '/usr/local/galaxy/galaxy-dist/tool-data/shared/ucsc/chrom/mm9.len',
 'dbkey': 'mm9',
 'experiment_name': 'H3K4me3_TAC_MACS2',
 'input_chipseq_file1': {'id': '6f0a311a444290f2',
 'uuid': 'null'},
 'input_control_file1': {'id': 'c21816a91f5dc24e',
 'uuid': '16f8ee5e-228f-41e2-921e-a07866edce06'},
 'major_command': {'gsize': '2716965481.0', 'bdg': False, '__current_case__': 0, 'advanced_options': {'advanced_options_selector': 'off', '__current_case__': 1}, 'input_chipseq_file1': '104715', 'xls_to_interval': 'False', 'major_command_selector': '', 'callpeak', 'input_control_file1': '104721', 'pq_options': {'pq_options_selector': 'qvalue', 'qvalue': '0.05', '__current_case__': 1}, 'bw': '300', 'nomodel_type': {'nomodel_type_selector': 'create_model', '__current_case__': 1}},
'stderr': '',
'stdout': '',
'tool_id': 'toolshed.g2.bx.psu.edu/repos/ziru-zhou/macs2/modencode_921e-a07866edce06'
}
```
(continues on next page)
show_history(history_id: str, contents: Literal[False] = False) → Dict[str, Any]
show_history(history_id: str, contents: Literal[True], deleted: bool | None = None, visible: bool | None = None, details: str | None = None, types: List[str] | None = None) → List[Dict[str, Any]]

Get details of a given history. By default, just get the history meta information.

Parameters

- **history_id (str)** – Encoded history ID to filter on
- **contents (bool)** – When True, instead of the history details, return a list with info for all datasets in the given history. Note that inside each dataset info dict, the id which should be used for further requests about this history dataset is given by the value of the id (not dataset_id) key.
- **deleted (bool or None)** – When contents=True, whether to filter for the deleted datasets (True) or for the non-deleted ones (False). If not set, no filtering is applied.
- **visible (bool or None)** – When contents=True, whether to filter for the visible datasets (True) or for the hidden ones (False). If not set, no filtering is applied.
- **details (str)** – When contents=True, include dataset details. Set to ‘all’ for the most information.
- **types (list)** – When contents=True, filter for history content types. If set to ['dataset'], return only datasets. If set to ['dataset_collection'], return only dataset collections. If not set, no filtering is applied.

Return type

dict or list of dicts

Returns

details of the given history or list of dataset info

Note: As an alternative to using the contents=True parameter, consider using gi.datasets.get_datasets(history_id=history_id) which offers more extensive functionality for filtering and ordering the results.

show_matching_datasets(history_id: str, name_filter: str | Pattern[str] | None = None) → List[Dict[str, Any]]

Get dataset details for matching datasets within a history.

Parameters

- **history_id (str)** – Encoded history ID
- **name_filter (str)** – Only datasets whose name matches the name_filter regular expression will be returned; use plain strings for exact matches and None to match all datasets in the history

Return type

list
Returns
List of dictionaries

**undelete_history**(history_id: str) → str
Undelete a history

Parameters

- **history_id** (str) – Encoded history ID

Return type
str

Returns
‘OK’ if it was deleted

**update_dataset**(history_id: str, dataset_id: str, **kwargs: Any) → Dict[str, Any]
Update history dataset metadata. Some of the attributes that can be modified are documented below.

Parameters

- **history_id** (str) – Encoded history ID
- **dataset_id** (str) – ID of the dataset
- **name** (str) – Replace history dataset name with the given string
- **datatype** (str) – Replace the datatype of the history dataset with the given string. The string must be a valid Galaxy datatype, both the current and the target datatypes must allow datatype changes, and the dataset must not be in use as input or output of a running job (including uploads), otherwise an error will be raised.
- **genome_build** (str) – Replace history dataset genome build (dbkey)
- **annotation** (str) – Replace history dataset annotation with given string
- **deleted** (bool) – Mark or unmark history dataset as deleted
- **visible** (bool) – Mark or unmark history dataset as visible

Return type
dict

Returns
details of the updated dataset

Changed in version 0.8.0: Changed the return value from the status code (type int) to a dict.

Update history dataset collection metadata. Some of the attributes that can be modified are documented below.

Parameters

- **history_id** (str) – Encoded history ID
- **dataset_collection_id** (str) – Encoded dataset_collection ID
- **name** (str) – Replace history dataset collection name with the given string
- **deleted** (bool) – Mark or unmark history dataset collection as deleted
- **visible** (bool) – Mark or unmark history dataset collection as visible

Return type
dict
Returns
the updated dataset collection attributes

Changed in version 0.8.0: Changed the return value from the status code (type int) to a dict.

`update_history(history_id: str, **kwargs: Any) → Dict[str, Any]`
Update history metadata information. Some of the attributes that can be modified are documented below.

Parameters
- `history_id (str)` – Encoded history ID
- `name (str)` – Replace history name with the given string
- `annotation (str)` – Replace history annotation with given string
- `deleted (bool)` – Mark or unmark history as deleted
- `purged (bool)` – If True, mark history as purged (permanently deleted).
- `published (bool)` – Mark or unmark history as published
- `importable (bool)` – Mark or unmark history as importable
- `tags (list)` – Replace history tags with the given list

Return type
dict

Returns
details of the updated history

Changed in version 0.8.0: Changed the return value from the status code (type int) to a dict.

`upload_dataset_from_library(history_id: str, lib_dataset_id: str) → Dict[str, Any]`
Upload a dataset into the history from a library. Requires the library dataset ID, which can be obtained from the library contents.

Parameters
- `history_id (str)` – Encoded history ID
- `lib_dataset_id (str)` – Encoded library dataset ID

Return type
dict

Returns
Information about the newly created HDA

---

**Invocations**

Contains possible interactions with the Galaxy workflow invocations

```python
class bioblend.galaxy.invocations.InvocationClient(galaxy_instance: GalaxyInstance)
A generic Client interface defining the common fields.

    All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...
```

---

5.1. Galaxy API
**cancel_invocation** *(invocation_id: str) → Dict[str, Any]*)

Cancel the scheduling of a workflow.

**Parameters**

- **invocation_id** *(str)* – Encoded workflow invocation ID

**Return type**

dict

**Returns**

The workflow invocation being cancelled

**get_invocation_biocompute_object** *(invocation_id: str) → Dict[str, Any]*)

Get a BioCompute object for an invocation.

**Parameters**

- **invocation_id** *(str)* – Encoded workflow invocation ID

**Return type**

dict

**Returns**

The BioCompute object

**get_invocation_report** *(invocation_id: str) → Dict[str, Any]*)

Get a Markdown report for an invocation.

**Parameters**

- **invocation_id** *(str)* – Encoded workflow invocation ID

**Return type**

dict

**Returns**

The invocation report. For example:

```json

{
    'markdown': '
    # Workflow Execution Summary of Example workflow
    ## Workflow Inputs
    ## Workflow Outputs
    ## Workflow
    galaxy
    workflow_display(workflow_id=f2db41e1fa331b3e)
    
    render_format: 'markdown',
    workflows: {'f2db41e1fa331b3e': {'name': 'Example workflow'}}
}
```

**get_invocation_report_pdf** *(invocation_id: str, file_path: str, chunk_size: int = 4096) → None*

Get a PDF report for an invocation.

**Parameters**

- **invocation_id** *(str)* – Encoded workflow invocation ID
- **file_path** *(str)* – Path to save the report

**get_invocation_step_jobs_summary** *(invocation_id: str) → List[Dict[str, Any]]*

Get a detailed summary of an invocation, listing all jobs with their job IDs and current states.

**Parameters**

- **invocation_id** *(str)* – Encoded workflow invocation ID

**Return type**

list of dicts
Returns

The invocation step jobs summary. For example:

```json
[
    {
        'id': 'e85a3be143d5905b',
        'model': 'Job',
        'populated_state': 'ok',
        'states': {'ok': 1}
    },
    {
        'id': 'c9468f6db6dc5c5f1',
        'model': 'Job',
        'populated_state': 'ok',
        'states': {'running': 1}
    },
    {
        'id': '2a56795cad3c7db3',
        'model': 'Job',
        'populated_state': 'ok',
        'states': {'new': 1}
    }
]
```

get_invocation_summary(invocation_id: str) → Dict[str, Any]

Get a summary of an invocation, stating the number of jobs which succeed, which are paused and which have errored.

Parameters

- **invocation_id** (str) – Encoded workflow invocation ID

Return type

dict

Returns

The invocation summary. For example:

```json
{
    'states': {'paused': 4, 'error': 2, 'ok': 2},
    'model': 'WorkflowInvocation',
    'id': 'a799d38679e985db',
    'populated_state': 'ok'}
```

get_invocations(workflow_id: str | None = None, history_id: str | None = None, user_id: str | None = None, include_terminal: bool = True, limit: int | None = None, view: str = 'collection', step_details: bool = False) → List[Dict[str, Any]]

Get all workflow invocations, or select a subset by specifying optional arguments for filtering (e.g. a workflow ID).

Parameters

- **workflow_id** (str) – Encoded workflow ID to filter on
- **history_id** (str) – Encoded history ID to filter on
- **user_id** (str) – Encoded user ID to filter on. This must be your own user ID if your are not an admin user.
- **include_terminal** (bool) – Whether to include terminal states.
- **limit** (int) – Maximum number of invocations to return - if specified, the most recent invocations will be returned.
- **view** (str) – Level of detail to return per invocation, either ‘element’ or ‘collection’.
- **step_details** (bool) – If ‘view’ is ‘element’, also include details on individual steps.
Return type

list

Returns

A list of workflow invocations. For example:

```json
[{
    'history_id': '2f94e8ae9edff68a',
    'id': 'df7a1f0c02a5b08e',
    'model_class': 'WorkflowInvocation',
    'state': 'new',
    'update_time': '2015-10-31T22:00:22',
    'uuid': 'c8aa2b1c-801a-11e5-a9e5-8ca98228593c',
    'workflow_id': '03501d7626bd192f'}]
```

gi: GalaxyInstance

module: str = 'invocations'

rerun_invocation(invocation_id: str, inputs_update: dict | None = None, params_update: dict | None = None, history_id: str | None = None, history_name: str | None = None, import_inputs_to_history: bool = False, replacement_params: dict | None = None, allow_tool_state_corrections: bool = False, inputs_by: Literal['step_index|step_uuid', 'step_index', 'step_id', 'step_uuid', 'name'] | None = None, parameters_normalized: bool = False) → Dict[str, Any]

Rerun a workflow invocation. For more extensive documentation of all parameters, see the `gi.workflows.invoke_workflow()` method.

Parameters

- **invocation_id** (str) – Encoded workflow invocation ID to be rerun
- **inputs_update** (dict) – If different datasets should be used to the original invocation, this should contain a mapping of workflow inputs to the new datasets and dataset collections.
- **params_update** (dict) – If different non-dataset tool parameters should be used to the original invocation, this should contain a mapping of the new parameter values.
- **history_id** (str) – The encoded history ID where to store the workflow outputs. Alternatively, `history_name` may be specified to create a new history.
- **history_name** (str) – Create a new history with the given name to store the workflow outputs. If both `history_id` and `history_name` are provided, `history_name` is ignored. If neither is specified, a new ‘Unnamed history’ is created.
- **import_inputs_to_history** (bool) – If True, used workflow inputs will be imported into the history. If False, only workflow outputs will be visible in the given history.
- **allow_tool_state_corrections** (bool) – If True, allow Galaxy to fill in missing tool state when running workflows. This may be useful for workflows using tools that have changed over time or for workflows built outside of Galaxy with only a subset of inputs defined.
- **replacement_params** (dict) – pattern-based replacements for post-job actions
- **inputs_by** (str) – Determines how inputs are referenced. Can be “step_index|step_uuid” (default), “step_index”, “step_id”, “step_uuid”, or “name”.

Chapter 5. API Documentation
• **parameters_normalized** *(bool)* – Whether Galaxy should normalize the input parameters to ensure everything is referenced by a numeric step ID. Default is `False`, but when setting parameters for a subworkflow, `True` is required.

**Return type**
dict

**Returns**
A dict describing the new workflow invocation.

**Note:** This method works only on Galaxy 21.01 or later.

### run_invocation_step_action(*invocation_id: str, step_id: str, action: Any*) → Dict[str, Any]
Execute an action for an active workflow invocation step. The nature of this action and what is expected will vary based on the the type of workflow step (the only currently valid action is True/False for pause steps).

**Parameters**
- **invocation_id** *(str)* – Encoded workflow invocation ID
- **step_id** *(str)* – Encoded workflow invocation step ID
- **action** *(object)* – Action to use when updating state, semantics depends on step type.

**Return type**
dict

**Returns**
Representation of the workflow invocation step

### show_invocation(*invocation_id: str*) → Dict[str, Any]
Get a workflow invocation dictionary representing the scheduling of a workflow. This dictionary may be sparse at first (missing inputs and invocation steps) and will become more populated as the workflow is actually scheduled.

**Parameters**
- **invocation_id** *(str)* – Encoded workflow invocation ID

**Return type**
dict

**Returns**
The workflow invocation. For example:

```python
{
    'history_id': '2f94e8ae9edff68a',
    'id': 'df71f0c02a5b08e',
    'inputs': {'0': {'id': 'a7db2fac67043c7e',
                     'src': 'hda',
                     'uuid': '7932ffe0-2340-4952-8857-dbaa50f1f46a'}},
    'model_class': 'WorkflowInvocation',
    'state': 'ready',
    'steps': [{'action': None,
               'id': 'd413a19dec13d11e',
               'job_id': None,
               'model_class': 'WorkflowInvocationStep',
               'order_index': 0,
               'state': 'ready',
               'steps': None}],
    'history_id': '3f94e8ae9edff68a',
    'id': 'df71f0c02a5b08e',
    'inputs': {'0': {'id': 'a7db2fac67043c7e',
                     'src': 'hda',
                     'uuid': '7932ffe0-2340-4952-8857-dbaa50f1f46a'}},
    'model_class': 'WorkflowInvocation',
    'state': 'ready',
    'steps': [{'action': None,
               'id': 'd413a19dec13d11e',
               'job_id': None,
               'model_class': 'WorkflowInvocationStep',
               'order_index': 0,
               'state': 'ready',
               'steps': None}],
```
BioBlend Documentation, Release 1.2.0

show_invocation_step(invasion_id: str, step_id: str) → Dict[str, Any]

See the details of a particular workflow invocation step.

Parameters

• **invasion_id** (str) – Encoded workflow invocation ID
• **step_id** (str) – Encoded workflow invocation step ID

Return type
dict

Returns

The workflow invocation step. For example:

```json
{
    'action': None,
    'id': '63cd3858d057a6d1',
    'job_id': None,
    'model_class': 'WorkflowInvocationStep',
    'order_index': 2,
    'state': None,
    'update_time': '2015-10-31T22:11:14',
    'workflow_step_id': '52e496b945151ee8',
    'workflow_step_label': None,
    'workflow_step_uuid': '4060554c-1dd5-4287-9040-8b4f281cf9dc'
}
```


Wait until an invocation is in a terminal state.

Parameters

• **invasion_id** (str) – Invocation ID to wait for.
• **maxwait** (float) – Total time (in seconds) to wait for the invocation state to become terminal. If the invocation state is not terminal within this time, a `TimeoutException` will be raised.
BioBlend Documentation, Release 1.2.0

- **interval** *(float)* – Time (in seconds) to wait between 2 consecutive checks.
- **check** *(bool)* – Whether to check if the invocation terminal state is ‘scheduled’.

**Return type**

dict

**Returns**
Details of the workflow invocation.

### Jobs

Contains possible interactions with the Galaxy Jobs

**class** `bioblend.galaxy.jobs.JobsClient(galaxy_instance: GalaxyInstance)`

A generic Client interface defining the common fields.

All clients *must* define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

**cancel_job**(job_id: str) → bool

Cancel a job, deleting output datasets.

**Parameters**

- **job_id** *(str)* – job ID

**Return type**

bool

**Returns**

True if the job was successfully cancelled, False if it was already in a terminal state before the cancellation.

**get_common_problems**(job_id: str) → Dict[str, Any]

Query inputs and jobs for common potential problems that might have resulted in job failure.

**Parameters**

- **job_id** *(str)* – job ID

**Return type**

dict

**Returns**

dict containing potential problems

**Note:** This method works only on Galaxy 19.05 or later.

**get_destination_params**(job_id: str) → Dict[str, Any]

Get destination parameters for a job, describing the environment and location where the job is run.

**Parameters**

- **job_id** *(str)* – job ID

**Return type**

dict

5.1. Galaxy API
Returns
Destination parameters for the given job

Note: This method works only on Galaxy 20.05 or later and if the user is a Galaxy admin.

get_inputs

\[\text{get_inputs}(\text{job_id: str}) \rightarrow \text{List[Dict[str, Any]]}\]

Get dataset inputs used by a job.

Parameters
job_id (str) – job ID

Return type
list of dicts

Returns
Inputs for the given job

get_jobs

\[\text{get_jobs}(\text{state: str | None = None, history_id: str | None = None, invocation_id: str | None = None, tool_id: str | None = None, workflow_id: str | None = None, user_id: str | None = None, date_range_min: str | None = None, date_range_max: str | None = None, limit: int = 500, offset: int = 0, user_details: bool = False, order_by: Literal['create_time', 'update_time'] = 'update_time') \rightarrow \text{List[Dict[str, Any]]}\]

Get all jobs, or select a subset by specifying optional arguments for filtering (e.g. a state).

If the user is an admin, this will return jobs for all the users, otherwise only for the current user.

Parameters
• state (str or list of str) – Job states to filter on.
• history_id (str) – Encoded history ID to filter on.
• invocation_id (str) – Encoded workflow invocation ID to filter on.
• tool_id (str or list of str) – Tool IDs to filter on.
• workflow_id (string) – Encoded workflow ID to filter on.
• user_id (str) – Encoded user ID to filter on. Only admin users can access the jobs of other users.
• date_range_min (str) – Minimum job update date (in YYYY-MM-DD format) to filter on.
• date_range_max (str) – Maximum job update date (in YYYY-MM-DD format) to filter on.
• limit (int) – Maximum number of jobs to return.
• offset (int) – Return jobs starting from this specified position. For example, if limit is set to 100 and offset to 200, jobs 200-299 will be returned.
• user_details (bool) – If True and the user is an admin, add the user email to each returned job dictionary.
• order_by (str) – Whether to order jobs by create_time or update_time (the default).

Return type
list of dict

Returns
Summary information for each selected job. For example:
Note: The following parameters work only on Galaxy 21.05 or later: user_id, limit, offset, workflow_id, invocation_id.

get_metrics(job_id: str) → List[Dict[str, Any]]

Return job metrics for a given job.

Parameters

job_id (str) – job ID

Return type

list

Returns

list containing job metrics

Note: Calling show_job() with full_details=True also returns the metrics for a job if the user is an admin. This method allows to fetch metrics even as a normal user as long as the Galaxy instance has the expose_potentially_sensitive_job_metrics option set to true in the config/galaxy.yml configuration file.

get_outputs(job_id: str) → List[Dict[str, Any]]

Get dataset outputs produced by a job.

Parameters

job_id (str) – job ID

Return type

list of dicts

Returns

Outputs of the given job

get_state(job_id: str) → str

Display the current state for a given job of the current user.

Parameters

job_id (str) – job ID

Return type

str
Returns

state of the given job among the following values: new, queued, running, waiting, ok. If the
state cannot be retrieved, an empty string is returned.

New in version 0.5.3.

module: str = 'jobs'

report_error(job_id: str, dataset_id: str, message: str, email: str | None = None) → Dict[str, Any]

Report an error for a given job and dataset to the server administrators.

Parameters

- job_id (str) – job ID
- dataset_id (str) – Dataset ID
- message (str) – Error message
- email (str) – Email for error report submission. If not specified, the email associated
  with the Galaxy user account is used by default.

Return type
dict

Returns
dict containing job error reply

Note: This method works only on Galaxy 20.01 or later.

rerun_job(job_id: str, remap: bool = False, tool_inputs_update: Dict[str, Any] | None = None, history_id: str | None = None) → Dict[str, Any]

Rerun a job.

Parameters

- job_id (str) – job ID
- remap (bool) – when True, the job output(s) will be remapped onto the dataset(s) created
  by the original job; if other jobs were waiting for this job to finish successfully, they will
  be resumed using the new outputs of this tool run. When False, new job output(s) will be
  created. Note that if Galaxy does not permit remapping for the job in question, specifying
  True will result in an error.
- tool_inputs_update (dict) – dictionary specifying any changes which should be made
  to tool parameters for the rerun job. This dictionary should have the same structure as is
  required when submitting the tool_inputs dictionary to gi.tools.run_tool(), but
  only needs to include the inputs or parameters to be updated for the rerun job.
- history_id (str) – ID of the history in which the job should be executed; if not specified,
  the same history will be used as the original job run.

Return type
dict

Returns
Information about outputs and the rerun job

Note: This method works only on Galaxy 21.01 or later.
resume_job(job_id: str) → List[Dict[str, Any]]
Resume a job if it is paused.

Parameters
job_id (str) – job ID

Return type
list of dicts

Returns
list of dictionaries containing output dataset associations

search_jobs(tool_id: str, inputs: Dict[str, Any], state: str | None = None) → List[Dict[str, Any]]
Return jobs matching input parameters.

Parameters
• tool_id (str) – only return jobs associated with this tool ID
• inputs (dict) – return only jobs that have matching inputs
• state (str) – only return jobs in this state

Return type
list of dicts

Returns
Summary information for each matching job

This method is designed to scan the list of previously run jobs and find records of jobs with identical input parameters and datasets. This can be used to minimize the amount of repeated work by simply recycling the old results.

Changed in version 0.16.0: Replaced the job_info parameter with separate tool_id, inputs and state.

show_job(job_id: str, full_details: bool = False) → Dict[str, Any]
Get details of a given job of the current user.

Parameters
• job_id (str) – job ID
• full_details (bool) – when True, the complete list of details for the given job.

Return type
dict

Returns
A description of the given job. For example:

```json
{
'create_time': '2014-03-01T16:17:29.828624',
'exit_code': 0,
'id': 'a799d38679e985db',
'inputs': {
'input': {
'id': 'ebfb8f50c6abde6d', 'src': 'hda'}},
'model_class': 'Job',
'outputs': {
'output': {
'id': 'a799d38679e985db', 'src': 'hda'}},
'params': {
'chromInfo': "/opt/galaxy-central/tool-data/shared/ucsc/chrom/?.len"},
'dbkey': "?",
' seq_col': "2",
'title_col': "["1"]",
```
(continues on next page)
show_job_lock() → bool
Show whether the job lock is active or not. If it is active, no jobs will dispatch on the Galaxy server.

    Return type
    bool

    Returns
    Status of the job lock

    Note: This method works only on Galaxy 20.05 or later and if the user is a Galaxy admin.

update_job_lock(active: bool = False) → bool
Update the job lock status by setting active to either True or False. If True, all job dispatching will be blocked.

    Return type
    bool

    Returns
    Updated status of the job lock

    Note: This method works only on Galaxy 20.05 or later and if the user is a Galaxy admin.

Wait until a job is in a terminal state.

    Parameters
    • job_id (str) – job ID
    • maxwait (float) – Total time (in seconds) to wait for the job state to become terminal. If the job state is not terminal within this time, a TimeoutException will be raised.
    • interval (float) – Time (in seconds) to wait between 2 consecutive checks.
    • check (bool) – Whether to check if the job terminal state is ‘ok’.

    Return type
    dict

    Returns
    Details of the given job.
Libraries

Contains possible interactions with the Galaxy Data Libraries

```python
class bioblend.galaxy.libraries.LibraryClient(galaxy_instance: GalaxyInstance)
```

A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

```python
copy_from_dataset(library_id: str, dataset_id: str, folder_id: str | None = None, message: str = '') → Dict[str, Any]
```

Copy a Galaxy dataset into a library.

**Parameters**

- `library_id` (str) – id of the library where to place the uploaded file
- `dataset_id` (str) – id of the dataset to copy from
- `folder_id` (str) – id of the folder where to place the uploaded files. If not provided, the root folder will be used
- `message` (str) – message for copying action

**Return type**

dict

**Returns**

LDDA information

```python
create_folder(library_id: str, folder_name: str, description: str | None = None, base_folder_id: str | None = None) → List[Dict[str, Any]]
```

Create a folder in a library.

**Parameters**

- `library_id` (str) – library id to use
- `folder_name` (str) – name of the new folder in the data library
- `description` (str) – description of the new folder in the data library
- `base_folder_id` (str) – id of the folder where to create the new folder. If not provided, the root folder will be used

**Return type**

list

**Returns**

List with a single dictionary containing information about the new folder

```python
create_library(name: str, description: str | None = None, synopsis: str | None = None) → Dict[str, Any]
```

Create a data library with the properties defined in the arguments.

**Parameters**

- `name` (str) – Name of the new data library
- `description` (str) – Optional data library description
- `synopsis` (str) – Optional data library synopsis
Return type
dict

Returns
Details of the created library. For example:

```json
{'id': 'f740ab636b360a70',
 'name': 'Library from bioblend',
 'url': '/api/libraries/f740ab636b360a70'}
```

delete_library(library_id: str) → Dict[str, Any]
Delete a data library.

Parameters
- **library_id** (str) – Encoded data library ID identifying the library to be deleted

Return type
dict

Returns
Information about the deleted library

Warning: Deleting a data library is irreversible - all of the data from the library will be permanently deleted.

delete_library_dataset(library_id: str, dataset_id: str, purged: bool = False) → Dict[str, Any]
Delete a library dataset in a data library.

Parameters
- **library_id** (str) – library id where dataset is found in
- **dataset_id** (str) – id of the dataset to be deleted
- **purged** (bool) – Indicate that the dataset should be purged (permanently deleted)

Return type
dict

Returns
A dictionary containing the dataset id and whether the dataset has been deleted. For example:

```json
{'deleted': True,
 'id': '60e680a037f41974'}
```

get_dataset_permissions(dataset_id: str) → Dict[str, Any]
Get the permissions for a dataset.

Parameters
- **dataset_id** (str) – id of the dataset

Return type
dict

Returns
dictionary with all applicable permissions’ values
get_folders(library_id: str, folder_id: str | None = None, name: str | None = None) \rightarrow List[Dict[str, Any]]

Get all the folders in a library, or select a subset by specifying a folder name for filtering.

Parameters

- **library_id** (str) – library id to use
- **name** (str) – Folder name to filter on. For name specify the full path of the folder starting from the library’s root folder, e.g. /subfolder/subsubfolder.

Return type

list

Returns

list of dicts each containing basic information about a folder

Changed in version 1.1.1: Using the deprecated folder_id parameter now raises a ValueError exception.

get_libraries(library_id: str | None = None, name: str | None = None, deleted: bool | None = False) \rightarrow List[Dict[str, Any]]

Get all libraries, or select a subset by specifying optional arguments for filtering (e.g. a library name).

Parameters

- **name** (str) – Library name to filter on.
- **deleted** (bool) – If False (the default), return only non-deleted libraries. If True, return only deleted libraries. If None, return both deleted and non-deleted libraries.

Return type

list

Returns

list of dicts each containing basic information about a library

Changed in version 1.1.1: Using the deprecated library_id parameter now raises a ValueError exception.

get_library_permissions(library_id: str) \rightarrow Dict[str, Any]

Get the permissions for a library.

Parameters

- **library_id** (str) – id of the library

Return type

dict

Returns

dictionary with all applicable permissions’ values

module: str = 'libraries'

set_dataset_permissions(dataset_id: str, access_in: List[str] | None = None, modify_in: List[str] | None = None, manage_in: List[str] | None = None) \rightarrow Dict[str, Any]

Set the permissions for a dataset. Note: it will override all security for this dataset even if you leave out a permission type.

Parameters

- **dataset_id** (str) – id of the dataset
- **access_in** (list) – list of role ids
- **modify_in** (list) – list of role ids
- **manage_in** (list) – list of role ids

**Return type**

dict

**Returns**
dictionary with all applicable permissions’ values

**set_library_permissions**

```python
```

Set the permissions for a library. Note: it will override all security for this library even if you leave out a permission type.

**Parameters**

- **library_id** (str) – id of the library
- **access_in** (list) – list of role ids
- **modify_in** (list) – list of role ids
- **add_in** (list) – list of role ids
- **manage_in** (list) – list of role ids

**Return type**

dict

**Returns**

General information about the library

**show_dataset**

```python
show_dataset(library_id: str, dataset_id: str) → Dict[str, Any]
```

Get details about a given library dataset. The required library_id can be obtained from the datasets’s library content details.

**Parameters**

- **library_id** (str) – library id where dataset is found in
- **dataset_id** (str) – id of the dataset to be inspected

**Return type**

dict

**Returns**

A dictionary containing information about the dataset in the library

**show_folder**

```python
show_folder(library_id: str, folder_id: str) → Dict[str, Any]
```

Get details about a given folder. The required folder_id can be obtained from the folder’s library content details.

**Parameters**

- **library_id** (str) – library id to inspect folders in
- **folder_id** (str) – id of the folder to be inspected

**Return type**

dict

**Returns**

Information about the folder
show_library(library_id: str, contents: bool = False) \rightarrow Dict[str, Any]

Get information about a library.

Parameters

- **library_id**(str) – filter for library by library id
- **contents**(bool) – whether to get contents of the library (rather than just the library de-
tails)

Return type
dict

Returns
details of the given library

update_library_dataset(dataset_id: str, **kwargs: Any) \rightarrow Dict[str, Any]

Update library dataset metadata. Some of the attributes that can be modified are documented below.

Parameters

- **dataset_id**(str) – id of the dataset to be updated
- **name**(str) – Replace library dataset name with the given string
- **misc_info**(str) – Replace library dataset misc_info with given string
- **file_ext**(str) – Replace library dataset extension (must exist in the Galaxy registry)
- **genome_build**(str) – Replace library dataset genome build (dbkey)
- **tags**(list) – Replace library dataset tags with the given list

Return type
dict

Returns
details of the updated dataset

upload_file_contents(library_id: str, pasted_content: str, folder_id: str | None = None, file_type: str = 'auto', dbkey: str = '?', tags: List[str] | None = None) \rightarrow List[Dict[str, Any]]

Upload pasted_content to a data library as a new file.

Parameters

- **library_id**(str) – id of the library where to place the uploaded file
- **pasted_content**(str) – Content to upload into the library
- **folder_id**(str) – id of the folder where to place the uploaded file. If not provided, the root folder will be used
- **file_type**(str) – Galaxy file format name
- **dbkey**(str) – Dbkey
- **tags**(list) – A list of tags to add to the datasets

Return type
list

Returns
List with a single dictionary containing information about the LDDA
upload_file_from_local_path(library_id: str, file_local_path: str, folder_id: str | None = None, file_type: str = 'auto', dbkey: str = '?', tags: List[str] | None = None) → List[Dict[str, Any]]

Read local file contents from file_local_path and upload data to a library.

**Parameters**

- **library_id (str)** – id of the library where to place the uploaded file
- **file_local_path (str)** – path of local file to upload
- **folder_id (str)** – id of the folder where to place the uploaded file. If not provided, the root folder will be used
- **file_type (str)** – Galaxy file format name
- **dbkey (str)** – Dbkey
- **tags (list)** – A list of tags to add to the datasets

**Return type**

list

**Returns**

List with a single dictionary containing information about the LDDA


Upload all files in the specified subdirectory of the Galaxy library import directory to a library.

**Parameters**

- **library_id (str)** – id of the library where to place the uploaded file
- **server_dir (str)** – relative path of the subdirectory of `library_import_dir` to upload. All and only the files (i.e. no subdirectories) contained in the specified directory will be uploaded
- **folder_id (str)** – id of the folder where to place the uploaded files. If not provided, the root folder will be used
- **file_type (str)** – Galaxy file format name
- **dbkey (str)** – Dbkey
- **link_data_only (str)** – either ‘copy_files’ (default) or ‘link_to_files’. Setting to ‘link_to_files’ symlinks instead of copying the files
- **roles (str)** – ???
- **preserve_dirs (bool)** – Indicate whether to preserve the directory structure when importing dir
- **tag_using_filenames (bool)** – Indicate whether to generate dataset tags from filenames.

Changed in version 0.14.0: Changed the default from True to False.
- **tags (list)** – A list of tags to add to the datasets

**Return type**

list
Returns
List with a single dictionary containing information about the LDDA

Note: This method works only if the Galaxy instance has the library_import_dir option configured in the config/galaxy.yml configuration file.


Upload a file to a library from a URL.

Parameters
- library_id (str) – id of the library where to place the uploaded file
- file_url (str) – URL of the file to upload
- folder_id (str) – id of the folder where to place the uploaded file. If not provided, the root folder will be used
- file_type (str) – Galaxy file format name
- dbkey (str) – Dbkey
- tags (list) – A list of tags to add to the datasets

Return type
list

Returns
List with a single dictionary containing information about the LDDA


Upload a set of files already present on the filesystem of the Galaxy server to a library.

Parameters
- library_id (str) – id of the library where to place the uploaded file
- filesystem_paths (str) – file paths on the Galaxy server to upload to the library, one file per line
- folder_id (str) – id of the folder where to place the uploaded files. If not provided, the root folder will be used
- file_type (str) – Galaxy file format name
- dbkey (str) – Dbkey
- link_data_only (str) – either ‘copy_files’ (default) or ‘link_to_files’. Setting to ‘link_to_files’ symlinks instead of copying the files
- roles (str) – ???
- preserve_dirs (bool) – Indicate whether to preserve the directory structure when importing dir
- tag_using_filenames (bool) – Indicate whether to generate dataset tags from filenames.
BioBlend Documentation, Release 1.2.0

Changed in version 0.14.0: Changed the default from True to False.

• **tags** (list) – A list of tags to add to the datasets

**Return type**

list

**Returns**

List of dictionaries containing information about each uploaded LDDA.

---

**Note:** This method works only if the Galaxy instance has the `allow_path_paste` option set to True in the `config/galaxy.yml` configuration file.

---

**wait_for_dataset**

```python
library_id: str, dataset_id: str, maxwait: float = 12000, interval: float = 3) \n\n→ Dict[str, Any]
```

Wait until the library dataset state is terminal (‘ok’, ‘empty’, ‘error’, ‘discarded’ or ‘failed_metadata’).

**Parameters**

• **library_id** (str) – library id where dataset is found in

• **dataset_id** (str) – id of the dataset to wait for

• **maxwait** (float) – Total time (in seconds) to wait for the dataset state to become terminal.

If the dataset state is not terminal within this time, a `DatasetTimeoutException` will be thrown.

• **interval** (float) – Time (in seconds) to wait between 2 consecutive checks.

**Return type**

dict

**Returns**

A dictionary containing information about the dataset in the library

---

**Quotas**

Contains possible interactions with the Galaxy Quota

```python
class bioblend.galaxy.quotas.QuotaClient(galaxy_instance: GalaxyInstance)
```

A generic Client interface defining the common fields.

All clients *must* define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

```python
create_quota(name: str, description: str, amount: str, operation: Literal['+', '-', '='], default: Literal['no', 'registered', 'unregistered'] | None = 'no', in_users: List[str] | None = None, in_groups: List[str] | None = None) \n\n→ Dict[str, Any]
```

Create a new quota

**Parameters**

• **name** (str) – Name for the new quota. This must be unique within a Galaxy instance.

• **description** (str) – Quota description

• **amount** (str) – Quota size (E.g. 10000MB, 99 gb, 0.2T, unlimited)

---
• **operation** *(str)* – One of (+, -, =)

• **default** *(str)* – Whether or not this is a default quota. Valid values are no, unregistered, registered. None is equivalent to no.

• **in_users** *(list of str)* – A list of user IDs or user emails.

• **in_groups** *(list of str)* – A list of group IDs or names.

**Return type**
dict

**Returns**
A description of quota. For example:

```json
{
    'url': '/galaxy/api/quotas/386f14984287a0f7',
    'model_class': 'Quota',
    'message': "Quota 'Testing' has been created with 1 associated users
and 0 associated groups.",
    'id': '386f14984287a0f7',
    'name': 'Testing'}
```

**delete_quota**(quota_id: str) → str
Delete a quota

Before a quota can be deleted, the quota must not be a default quota.

**Parameters**
quota_id *(str)* – Encoded quota ID.

**Return type**
str

**Returns**
A description of the changes, mentioning the deleted quota. For example:

"Deleted 1 quotas: Testing-B"

**get_quotas**(deleted: bool = False) → List[Dict[str, Any]]
Get a list of quotas

**Parameters**
deaded *(bool)* – Only return quota(s) that have been deleted

**Return type**
list

**Returns**
A list of dicts with details on individual quotas. For example:

```json
[{ 'id': '0604c8a56abe9a50',
  'model_class': 'Quota',
  'name': 'test',
  'url': '/api/quotas/0604c8a56abe9a50'},
 { 'id': 'lee267091d0190af',
  'model_class': 'Quota',
  'name': 'workshop',
  'url': '/api/quotas/lee267091d0190af'}]
```
module: str = 'quotas'

show_quota(quota_id: str, deleted: bool = False) → Dict[str, Any]
    Display information on a quota

    Parameters
    • quota_id (str) – Encoded quota ID
    • deleted (bool) – Search for quota in list of ones already marked as deleted

    Return type
    dict

    Returns
    A description of quota. For example:

    ```
    {'bytes': 107374182400,
    'default': [],
    'description': 'just testing',
    'display_amount': '100.0 GB',
    'groups': [],
    'id': '0604c8a56abe9a50',
    'model_class': 'Quota',
    'name': 'test',
    'operation': '=',
    'users': []}
    ```

undelete_quota(quota_id: str) → str
    Undelete a quota

    Parameters
    quota_id (str) – Encoded quota ID.

    Return type
    str

    Returns
    A description of the changes, mentioning the undeleted quota. For example:

    "Undeleted 1 quotas: Testing-B"

update_quota(quota_id: str, name: str | None = None, description: str | None = None, amount: str | None = None, operation: Literal['+', '-', '='] | None = None, default: str = 'no', in_users: List[str] | None = None, in_groups: List[str] | None = None) → str
    Update an existing quota

    Parameters
    • quota_id (str) – Encoded quota ID
    • name (str) – Name for the new quota. This must be unique within a Galaxy instance.
    • description (str) – Quota description. If you supply this parameter, but not the name, an error will be thrown.
    • amount (str) – Quota size (E.g. 10000MB, 99 gb, 0.2T, unlimited)
    • operation (str) – One of (+, -, =). If you wish to change this value, you must also provide the amount, otherwise it will not take effect.
• **default** *(str)* – Whether or not this is a default quota. Valid values are *no*, *unregistered*, *registered*. Calling this method with `default="no"` on a non-default quota will throw an error. Not passing this parameter is equivalent to passing *no*.

• **in_users** *(list of str)* – A list of user IDs or user emails.

• **in_groups** *(list of str)* – A list of group IDs or names.

**Return type**

*str*

**Returns**

A semicolon separated list of changes to the quota. For example:

```
"Quota 'Testing-A' has been renamed to 'Testing-B'; Quota 'Testing-e' is now '-100.0 GB'; Quota 'Testing-B' is now the default for unregistered users"
```

---

**Roles**

Contains possible interactions with the Galaxy Roles

```python
class bioblend.galaxy.roles.RolesClient(galaxy_instance: GalaxyInstance)
A generic Client interface defining the common fields.
```

All clients *must* define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

```python
def create_role(role_name: str, description: str, user_ids: List[str] | None = None, group_ids: List[str] | None = None) -> Dict[str, Any]
Create a new role.
```

**Parameters**

• **role_name** *(str)* – A name for the new role

• **description** *(str)* – Description for the new role

• **user_ids** *(list)* – A list of encoded user IDs to add to the new role

• **group_ids** *(list)* – A list of encoded group IDs to add to the new role

**Return type**

*dict*

**Returns**

Details of the newly created role. For example:

```
{
'description': 'desc',
'url': '/api/roles/ebfb8f50c6abde6d',
'model_class': 'Role',
'type': 'admin',
'id': 'ebfb8f50c6abde6d',
'name': 'Foo'
}
```

Changed in version 0.15.0: Changed the return value from a 1-element list to a dict.
get_roles() → List[Dict[str, Any]]
Displays a collection (list) of roles.

**Return type**
list

**Returns**
A list of dicts with details on individual roles. For example:

```
[{
  "id": "f2db41e1fa331b3e",
  "model_class": "Role",
  "name": "Foo",
  "url": "/api/roles/f2db41e1fa331b3e"},
{
  "id": "f597429621d6eb2b",
  "model_class": "Role",
  "name": "Bar",
  "url": "/api/roles/f597429621d6eb2b"}
```

**module**: str = 'roles'

**show_role**(role_id: str) → Dict[str, Any]
Display information on a single role

**Parameters**
role_id (str) – Encoded role ID

**Return type**
dict

**Returns**
Details of the given role. For example:

```
{"description": "Private Role for Foo",
 "id": "f2db41e1fa331b3e",
 "model_class": "Role",
 "name": "Foo",
 "type": "private",
 "url": "/api/roles/f2db41e1fa331b3e"}
```

**Tools**
Contains possible interaction dealing with Galaxy tools.

**class** bioblend.galaxy.tools.ToolClient(**galaxy_instance**: GalaxyInstance)**
A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

**build**(tool_id: str, inputs: Dict[str, Any] | None = None, tool_version: str | None = None, history_id: str | None = None) → Dict[str, Any]
This method returns the tool model, which includes an updated input parameter array for the given tool, based on user-defined “inputs”.

58 Chapter 5. API Documentation
Parameters

- **inputs** (dict) – (optional) inputs for the payload. For example:

  ```
  {
    "num_lines": "1",
    "input": {
      "values": [
        {
          "src": "hda",
          "id": "4d366c1196c36d18"
        }
      ]
    },
    "seed_source|seed_source_selector": "no_seed",
  }
  ```

- **tool_id** (str) – id of the requested tool
- **history_id** (str) – id of the requested history
- **tool_version** (str) – version of the requested tool

Return type
dict

Returns

Returns a tool model including dynamic parameters and updated values, repeats block etc. For example:

```
{
  "model_class": "Tool",
  "id": "random_lines1",
  "name": "Select random lines",
  "version": "2.0.2",
  "description": "from a file",
  "labels": [],
  "edam_operations": [],
  "edam_topics": [],
  "hidden": "",
  "is_workflow_compatible": True,
  "xrefs": [],
  "config_file": "/Users/joshij/galaxy/tools/filters/randomlines.xml",
  "panel_section_id": "textutil",
  "panel_section_name": "Text Manipulation",
  "form_style": "regular",
  "inputs": [
    {
      "model_class": "IntegerToolParameter",
      "name": "num_lines",
      "argument": None,
      "type": "integer",
      "label": "Randomly select",
      "help": "lines",
      "refresh_on_change": False,
    }
  ]
}
```
This tool selects N random lines from a file, with no repeats, and preserving ordering.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tool_id</td>
<td>id of the requested tool</td>
</tr>
<tr>
<td>Return type</td>
<td>list of dicts</td>
</tr>
<tr>
<td>Param</td>
<td>list containing the citations</td>
</tr>
</tbody>
</table>

get_tool_panel() → List[Dict[str, Any]]

Get a list of available tool elements in Galaxy’s configured toolbox.
Return type
list

Returns
List containing tools (if not in sections) or tool sections with nested tool descriptions.

See also:
bioblend.galaxy.toolshed.get_repositories()

get_tools(tool_id: str | None = None, name: str | None = None, trackster: bool | None = None) → List[Dict[str, Any]]
Get all tools, or select a subset by specifying optional arguments for filtering (e.g. a tool name).

Parameters
- name (str) – Tool name to filter on.
- trackster (bool) – whether to return only tools that are compatible with Trackster

Return type
list

Returns
List of tool descriptions.

See also:
bioblend.galaxy.toolshed.get_repositories()

Changed in version 1.1.1: Using the deprecated tool_id parameter now raises a ValueError exception.

gi: GalaxyInstance

install_dependencies(tool_id: str) → List[Dict[str, Any]]
Install dependencies for a given tool via a resolver. This works only for Conda currently.

Parameters
- tool_id (str) – id of the requested tool

Return type
list of dicts

Returns
List of tool requirement status dictionaries

Note: This method works only if the user is a Galaxy admin.

module: str = 'tools'

paste_content(content: str, history_id: str, **kwargs: Any) → Dict[str, Any]
Upload a string to a new dataset in the history specified by history_id.

Parameters
- content (str) – content of the new dataset to upload or a list of URLs (one per line) to upload
- history_id (str) – id of the history where to upload the content

Return type
dict
Returns
Information about the created upload job

See `upload_file()` for the optional parameters.

**post_to_fetch**(path: str, history_id: str, session_id: str, **kwargs: Any) → Dict[str, Any]
Make a POST request to the Fetch API after performing a tus upload.

This is called by `upload_file()` after performing an upload. This method is useful if you want to control
the tus uploader yourself (e.g. to report on progress):

```python
uploader = gi.get_tus_uploader(path, storage=storage)
while uploader.offset < uploader.file_size:
    uploader.upload_chunk()
    # perform other actions...
gi.tools.post_to_fetch(path, history_id, uploader.session_id, **upload_kwargs)
```

**Parameters**

- **session_id** (str) – Session ID returned by the tus service

See `upload_file()` for additional parameters.

**Return type**
dict

**Returns**
Information about the created upload job

**put_url**(content: str, history_id: str, **kwargs: Any) → Dict[str, Any]

Upload a string to a new dataset in the history specified by `history_id`.

**Parameters**

- **content** (str) – content of the new dataset to upload or a list of URLs (one per line) to upload
- **history_id** (str) – id of the history where to upload the content

**Return type**
dict

**Returns**
Information about the created upload job

See `upload_file()` for the optional parameters.

**reload**(tool_id: str) → dict

Reload the specified tool in the toolbox.

Any changes that have been made to the wrapper since the tool was last reloaded will take effect.

**Parameters**

- **tool_id** (str) – id of the requested tool

**Return type**
dict

**Param**
dict containing the id, name, and version of the reloaded tool. For example:
Note: This method works only if the user is a Galaxy admin.

**requirements**(*tool_id: str*) \rightarrow List[Dict[str, Any]]

Return the resolver status for a specific tool.

**Parameters**

- **tool_id** *(str)* – id of the requested tool

**Return type**

- list

**Returns**

List containing a resolver status dict for each tool requirement. For example:

```
[{
'cacheable': False,
'dependency_resolver': {'auto_init': True,
    'auto_install': False,
    'can_uninstall_dependencies': True,
    'ensure_channels': 'iuc,conda-forge,
bioconda,defaults',
'model_class': 'CondaDependencyResolver',
'prefix': '/mnt/galaxy/tool_dependencies/_conda',
'resolver_type': 'conda',
'resolves_simple_dependencies': True,
'use_local': False,
'versionless': False},
'dependency_type': 'conda',
environment_path: '/mnt/galaxy/tool_dependencies/_conda/envs/__blast@2.10.1',
'exact': True,
'model_class': 'MergedCondaDependency',
'name': 'blast',
'version': '2.10.1'}]
```

Note: This method works only if the user is a Galaxy admin.

**run_tool**(*history_id: str, tool_id: str, tool_inputs: InputsBuilder | dict, input_format: Literal[21.01, legacy] = legacy, data_manager_mode: Literal[populate, dry_run, bundle] | None = None*)

Runs tool specified by tool_id in history indicated by history_id with inputs from dict tool_inputs.

**Parameters**

- **history_id** *(str)* – encoded ID of the history in which to run the tool
- **tool_id** *(str)* – ID of the tool to be run
• **data_manager_mode** (**str**) – Possible values are ‘populate’, ‘dry_run’ and ‘bundle’.

  ‘populate’ is the default behavior for data manager tools and results in tool data table files being updated after the data manager job completes.

  ‘dry_run’ will skip any processing after the data manager job completes.

  ‘bundle’ will create a data manager bundle that can be imported on other Galaxy servers.

• **tool_inputs** (**dict**) – dictionary of input datasets and parameters for the tool (see below)

• **input_format** (**string**) – input format for the payload. Possible values are the default ‘legacy’ (where inputs nested inside conditionals or repeats are identified with e.g. ‘<conditional_name>|<input_name>’) or ‘21.01’ (where inputs inside conditionals or repeats are nested elements).

**Return type**

dict

**Returns**

Information about outputs and job For example:

```py
{
    'implicit_collections': [],
    'jobs': [{
        'create_time': '2019-05-08T12:26:16.067372',
        'exit_code': None,
        'id': '7dd125b61b35d782',
        'model_class': 'Job',
        'state': 'new',
        'tool_id': 'cut1',
        'update_time': '2019-05-08T12:26:16.067389'
    }],
    'output_collections': [],
    'outputs': [{
        'create_time': '2019-05-08T12:26:15.997739',
        'data_type': 'galaxy.datatypes.tabular.Tabular',
        'deleted': False,
        'file_ext': 'tabular',
        'file_size': 0,
        'genome_build': '?',
        'hda_ldda': 'hda',
        'hid': 42,
        'history_content_type': 'dataset',
        'history_id': 'df8fe5ddabf3ab1',
        'id': 'aeb6580396167f3',
        'metadata_column_names': None,
        'metadata_column_types': None,
        'metadata_columns': None,
        'metadata_comment_lines': None,
        'metadata_data_lines': None,
        'metadata_dbkey': '?',
        'metadata_delimiter': ' ',
        'misc_blurb': 'queued',
        'misc_info': None,
        'model_class': 'HistoryDatasetAssociation',
        'name': 'Cut on data 1',
        'output_name': 'out_file1',
        'peek': None,
        'purged': False,
    }]
```
The `tool_inputs` dict should contain input datasets and parameters in the (largely undocumented) format used by the Galaxy API. If you are unsure how to construct this dict for the tool you want to run, you can obtain a template by executing the `build()` method and taking the value of `state_inputs` from its output, then modifying it as you require. You can also check the examples in Galaxy’s API test suite.

```python
show_tool(tool_id: str, io_details: bool = False, link_details: bool = False) → Dict[str, Any]
```

Get details of a given tool.

**Parameters**

- `tool_id` *(str)* – id of the requested tool
- `io_details` *(bool)* – whether to get also input and output details
- `link_details` *(bool)* – whether to get also link details

**Return type**

`dict`

**Returns**

Information about the tool’s interface

```python
uninstall_dependencies(tool_id: str) → dict
```

Uninstall dependencies for a given tool via a resolver. This works only for Conda currently.

**Parameters**

- `tool_id` *(str)* – id of the requested tool

**Return type**

`dict`

**Returns**

Tool requirement status

---

**Note:** This method works only if the user is a Galaxy admin.

```python
upload_file(path: str, history_id: str, storage: str | None = None, metadata: dict | None = None, chunk_size: int | None = 10000000, **kwargs: Any) → Dict[str, Any]
```

Upload the file specified by `path` to the history specified by `history_id`.

**Parameters**

- `path` *(str)* – path of the file to upload
- `history_id` *(str)* – id of the history where to upload the file
- `storage` *(str)* – Local path to store URLs resuming uploads
- `metadata` *(dict)* – Metadata to send with upload request
- `chunk_size` *(int)* – Number of bytes to send in each chunk
- `file_name` *(str)* – (optional) name of the new history dataset
• **file_type** (*str*) – (optional) Galaxy datatype for the new dataset, default is auto
• **dbkey** (*str*) – (optional) genome dbkey
• **to POSIX lines** (*bool*) – if True (the default), convert universal line endings to POSIX line endings. Set to False when uploading a gzip, bz2 or zip archive containing a binary file
• **space_to_tab** (*bool*) – whether to convert spaces to tabs. Default is False. Applicable only if to POSIX lines is True
• **auto decompress** (*bool*) – Automatically decompress files if the uploaded file is compressed and the file type is not one that supports compression (e.g. `fastqsanger.gz`). Default is False.

**Return type**
dict

**Returns**
Information about the created upload job

**Note:** The following parameters work only on Galaxy 22.01 or later: `storage`, `metadata`, `chunk_size`, `auto decompress`.

### upload_from_ftp

**upload_from_ftp**(*path: str, history_id: str, **kwargs: Any*) → Dict[str, Any]

Upload the file specified by `path` from the user's FTP directory to the history specified by `history_id`.

**Parameters**
• **path** (*str*) – path of the file in the user's FTP directory
• **history_id** (*str*) – id of the history where to upload the file

See `upload_file()` for the optional parameters.

**Return type**
dict

**Returns**
Information about the created upload job

---

### Tool data tables

Contains possible interactions with the Galaxy Tool data tables

**class** `bioblend.galaxy.tool_data.ToolDataClient(galaxy_instance: GalaxyInstance)`

A generic Client interface defining the common fields.

All clients *must* define the following field (which will be used as part of the URL composition (e.g., `http://<galaxy_instance>/api/libraries`): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

**delete_data_table**(*data_table_id: str, values: str*) → Dict[str, Any]

Delete an item from a data table.

**Parameters**
• **data_table_id** (*str*) – ID of the data table
• **values (str)** – a “|” separated list of column contents, there must be a value for all the columns of the data table

**Return type**
dict

**Returns**
Remaining contents of the given data table

### get_data_tables()

→ List[Dict[str, Any]]

Get the list of all data tables.

**Return type**
list

**Returns**
A list of dicts with details on individual data tables. For example:

```python
[{
    "model_class": "TabularToolDataTable",
    "name": "fasta_indexes"
},
{
    "model_class": "TabularToolDataTable",
    "name": "bwa_indexes"
}]
```

**module**: str = 'tool_data'

### reload_data_table(data_table_id: str) → Dict[str, Any]

Reload a data table.

**Parameters**

- **data_table_id (str)** – ID of the data table

**Return type**
dict

**Returns**
A description of the given data table and its content. For example:

```python
{
    'columns': ['value', 'dbkey', 'name', 'path'],
    'fields': [[
        'test id',
        'test',
        'test name',
        '/opt/galaxy-dist/tool-data/test/seq/test id.fa'
    ],
    'model_class': 'TabularToolDataTable',
    'name': 'all_fasta'
}
```

### show_data_table(data_table_id: str) → Dict[str, Any]

Get details of a given data table.

**Parameters**

- **data_table_id (str)** – ID of the data table

**Return type**
dict

**Returns**
A description of the given data table and its content. For example:

```python
{
    'columns': ['value', 'dbkey', 'name', 'path'],
    'fields': [[
        'test id',
        ...
```
Tool dependencies

Contains interactions dealing with Galaxy dependency resolvers.

```python
class bioblend.galaxy.tool_dependencies.ToolDependenciesClient(galaxy_instance: GalaxyInstance)
    A generic Client interface defining the common fields.
    All clients must define the following field (which will be used as part of the URL composition
    (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries'
    | 'histories' | ...

delete_unused_dependency_paths(paths: List[str]) → None
    Delete unused paths

    Parameters
    paths (list) – paths to delete

    module: str = 'dependency_resolvers'

summarize_toolbox(index: int | None = None, tool_ids: List[str] | None = None, resolver_type: str | None =
    None, include_containers: bool = False, container_type: str | None = None, index_by:
    Literal['requirements', 'tools'] = 'requirements') → list
    Summarize requirements across toolbox (for Tool Management grid).

    Parameters
    • index (int) – index of the dependency resolver with respect to the dependency resolvers
      config file
    • tool_ids (List[str]) – tool_ids to return when index_by=tools
    • resolver_type (str) – restrict to specified resolver type
    • include_containers (bool) – include container resolvers in resolution
    • container_type (str) – restrict to specified container type
    • index_by (str) – By default results are grouped by requirements. Set to ‘tools’ to return
      one entry per tool.

    Return type
    list of dicts

    Returns
    dictified descriptions of the dependencies, with attribute dependency_type: None if no match
    was found. For example:
```
A generic Client interface defining the common fields.

All clients _must_ define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

**get_repositories() → List[Dict[str, Any]]**

Get the list of all installed Tool Shed repositories on this Galaxy instance.

**Return type**

* list

**Returns**

A list of dictionaries containing information about repositories present in the Tool Shed. For example:
Install a specified repository revision from a specified Tool Shed into this Galaxy instance. This example demonstrates installation of a repository that contains valid tools, loading them into a section of the Galaxy tool panel or creating a new tool panel section. You can choose if tool dependencies or repository dependencies should be installed through the Tool Shed, (use `install_tool_dependencies` or `install_repository_dependencies`) or through a resolver that supports installing dependencies (use `install_resolver_dependencies`). Note that any combination of the three dependency resolving variables is valid.

Installing the repository into an existing tool panel section requires the tool panel config file (e.g., tool_conf.xml, shed_tool_conf.xml, etc) to contain the given tool panel section:

```xml
<section id="from_test_tool_shed" name="From Test Tool Shed" version=""/>
```

### Parameters

- **tool_shed_url** *(str)* – URL of the Tool Shed from which the repository should be installed from (e.g., https://testtoolshed.g2.bx.psu.edu)
- **name** *(str)* – The name of the repository that should be installed
- **owner** *(str)* – The name of the repository owner
- **changeset_revision** *(str)* – The revision of the repository to be installed
- **install_tool_dependencies** *(bool)* – Whether or not to automatically handle tool dependencies (see https://galaxyproject.org/toolshed/tool-dependency-recipes/ for more details)
- **install_repository_dependencies** *(bool)* – Whether or not to automatically handle repository dependencies (see https://galaxyproject.org/toolshed/defining-repository-dependencies/ for more details)
- **install_resolver_dependencies** *(bool)* – Whether or not to automatically install resolver dependencies (e.g. conda).
- **tool_panel_section_id** *(str)* – The ID of the Galaxy tool panel section where the tool should be inserted under. Note that you should specify either this parameter or the `new_tool_panel_section_label`. If both are specified, this one will take precedence.
BioBlend Documentation, Release 1.2.0

- **new_tool_panel_section_label** (str) – The name of a Galaxy tool panel section that should be created and the repository installed into.

**module**: str = 'tool_shed_repositories'

**show_repository**(toolShed_id: str) → Dict[str, Any]

Get details of a given Tool Shed repository as it is installed on this Galaxy instance.

**Parameters**
- **toolShed_id** (str) – Encoded Tool Shed ID

**Return type**
- dict

**Returns**
Information about the tool For example:

```json
{
  'changeset_revision': 'b17455fb6222',
  'ctx_rev': '8',
  'owner': 'aaron',
  'status': 'Installed',
  'url': '/api/tool_shed_repositories/82de4a4c7135b20a'}
```

Changed in version 0.4.1: Changed method name from show_tool to show_repository to better align with the Tool Shed concepts

**uninstall_repository_revision**(name: str, owner: str, changeset_revision: str, tool_shed_url: str, remove_from_disk: bool = True) → Dict[str, Any]

Uninstalls a specified repository revision from this Galaxy instance.

**Parameters**
- **name** (str) – The name of the repository
- **owner** (str) – The owner of the repository
- **changeset_revision** (str) – The revision of the repository to uninstall
- **tool_shed_url** (str) – URL of the Tool Shed from which the repository was installed from (e.g., https://testtoolshed.g2.bx.psu.edu)
- **remove_from_disk** (bool) – whether to also remove the repository from disk (the default) or only deactivate it

**Return type**
- dict

**Returns**
If successful, a dictionary with a message noting the removal
**Users**

Contains possible interaction dealing with Galaxy users.

Most of these methods must be executed by a registered Galaxy admin user.

```python
class bioblend.galaxy.users.UserClient(galaxy_instance: GalaxyInstance)
    A generic Client interface defining the common fields.
    All clients must define the following field (which will be used as part of the URL composition
    (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries'
    | 'histories' | ...

    create_local_user(username: str, user_email: str, password: str) → Dict[str, Any]
        Create a new Galaxy local user.

        **Note:** This method works only if the Galaxy instance has the allow_user_creation
        option set to true and use_remote_user option set to false in the config/galaxy.yml configuration file.

        **Parameters**
        - **username (str)** – username of the user to be created
        - **user_email (str)** – email of the user to be created
        - **password (str)** – password of the user to be created

        **Return type**
        dict

        **Returns**
        a dictionary containing information about the created user

    create_remote_user(user_email: str) → Dict[str, Any]
        Create a new Galaxy remote user.

        **Note:** This method works only if the Galaxy instance has the allow_user_creation and
        use_remote_user options set to true in the config/galaxy.yml configuration file. Also note that
        setting use_remote_user will require an upstream authentication proxy server; however, if you do not
        have one, access to Galaxy via a browser will not be possible.

        **Parameters**
        - **user_email (str)** – email of the user to be created

        **Return type**
        dict

        **Returns**
        a dictionary containing information about the created user

    create_user_apikey(user_id: str) → str
        Create a new API key for a given user.

        **Parameters**
        - **user_id (str)** – encoded user ID
```
Return type
str

Returns
the API key for the user

delete_user(user_id: str, purge: bool = False) → Dict[str, Any]
Delete a user.

Note: This method works only if the Galaxy instance has the allow_user_deletion option set to true in the config/galaxy.yml configuration file.

Parameters

• user_id (str) – encoded user ID
• purge (bool) – if True, also purge (permanently delete) the history

Return type
dict

Returns
a dictionary containing information about the deleted user

get_current_user() → Dict[str, Any]
Display information about the user associated with this Galaxy connection.

Return type
dict

Returns
a dictionary containing information about the current user

get_or_create_user_apikey(user_id: str) → str
Get the current API key for a given user, creating one if it doesn’t exist yet.

Parameters
user_id (str) – encoded user ID

Return type
str

Returns
the API key for the user

Note: This method works only on Galaxy 21.01 or later.

get_user_apikey(user_id: str) → str
Get the current API key for a given user.

Parameters
user_id (str) – encoded user ID

Return type
str

Returns
the API key for the user, or ‘Not available.’ if it doesn’t exist yet.
get_users(deleted: bool = False, f_email: str | None = None, f_name: str | None = None, f_any: str | None = None) -> List[Dict[str, Any]]

Get a list of all registered users. If deleted is set to True, get a list of deleted users.

Parameters

- **deleted** (bool) – Whether to include deleted users
- **f_email** (str) – filter for user emails. The filter will be active for non-admin users only if the Galaxy instance has the `expose_user_email` option set to true in the config/galaxy.yml configuration file.
- **f_name** (str) – filter for user names. The filter will be active for non-admin users only if the Galaxy instance has the `expose_user_name` option set to true in the config/galaxy.yml configuration file.
- **f_any** (str) – filter for user email or name. Each filter will be active for non-admin users only if the Galaxy instance has the corresponding `expose_user_*` option set to true in the config/galaxy.yml configuration file.

Return type

list

Returns

a list of dicts with user details. For example:

```python
[{
    'email': 'a_user@example.org',
    'id': 'dda4709d9189f15',
    'url': '/api/users/dda4709d9189f15'
}]
```

module: str = 'users'

show_user(user_id: str, deleted: bool = False) -> Dict[str, Any]

Display information about a user.

Parameters

- **user_id** (str) – encoded user ID
- **deleted** (bool) – whether to return results for a deleted user

Return type

dict

Returns

a dictionary containing information about the user

update_user(user_id: str, user_data: Dict | None = None, **kwargs: Any) -> Dict[str, Any]

Update user information. You can either pass the attributes you want to change in the user_data dictionary, or provide them separately as keyword arguments. For attributes that cannot be expressed as keywords (e.g. extra_user_preferences use a | sign), pass them in user_data.

Parameters

- **user_id** (str) – encoded user ID
- **user_data** (dict) – a dict containing the values to be updated, eg. {
  “username”: “newUsername”,
  “email”: “new@email”
}
- **username** (str) – Replace user name with the given string
- **email** (str) – Replace user email with the given string
Return type
dict

Returns
details of the updated user

Visual

Contains possible interactions with the Galaxy visualization

class bioblend.galaxy.visual.VisualClient(galaxy_instance: GalaxyInstance)
A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

get_visualizations() → List[Dict[str, Any]]
Get the list of all visualizations.

Return type
dict

Returns
A list of dicts with details on individual visualizations. For example:

```python
[{ 'dbkey': 'eschColi_K12',
  'id': 'df1c7c96fc427c2d',
  'title': 'AVTest1',
  'type': 'trackster',
  'url': '/api/visualizations/df1c7c96fc427c2d'},
 { 'dbkey': 'mm9',
  'id': 'a669f50f8bf55b02',
  'title': 'Bam to Bigwig',
  'type': 'trackster',
  'url': '/api/visualizations/a669f50f8bf55b02'}]
```

module: str = 'visualizations'

show_visualization(visual_id: str) → Dict[str, Any]
Get details of a given visualization.

Parameters
visual_id (str) – Encoded visualization ID

Return type
dict

Returns
A description of the given visualization. For example:

```python
{ 'annotation': None,  
  'dbkey': 'mm9', 
  'id': '18df9134ea75e49c',
``` (continues on next page)
Workflows

Contains possible interactions with the Galaxy Workflows

```python
class bioblend.galaxy.workflows.WorkflowClient(galaxy_instance: GalaxyInstance)
    # A generic Client interface defining the common fields.
    # All clients must define the following field (which will be used as part of the URL composition
    # (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries'
    # 'histories' | ...
    cancel_invocation(workflow_id: str, invocation_id: str) -> Dict[str, Any]
    # Cancel the scheduling of a workflow.
    # Parameters
    # - workflow_id (str) – Encoded workflow ID
    # - invocation_id (str) – Encoded workflow invocation ID
    # Return type
    # dict
    # Returns
    # The workflow invocation being cancelled
    delete_workflow(workflow_id: str) -> None
    # Delete a workflow identified by workflow_id.
    # Parameters
    # - workflow_id (str) – Encoded workflow ID
    # Warning: Deleting a workflow is irreversible in Galaxy versions < 23.01 - all workflow data will be
    # permanently deleted.
    export_workflow_dict(workflow_id: str, version: int | None = None) -> Dict[str, Any]
    # Exports a workflow.
    # Parameters
    # - workflow_id (str) – Encoded workflow ID
    # - version (int) – Workflow version to export
    # Return type
    # dict
```

76 Chapter 5. API Documentation
Returns
Dictionary representing the requested workflow

export_workflow_to_local_path(workflow_id: str, file_local_path: str, use_default_filename: bool = True) → None

Exports a workflow in JSON format to a given local path.

Parameters
• workflow_id (str) – Encoded workflow ID
• file_local_path (str) – Local path to which the exported file will be saved. (Should not contain filename if use_default_name=True)
• use_default_filename (bool) – If the use_default_name parameter is True, the exported file will be saved as file_local_path/Galaxy-Workflow-%s.ga, where %s is the workflow name. If use_default_name is False, file_local_path is assumed to contain the full file path including filename.

Return type
None

Returns
None


Extract a workflow from a history.

Parameters
• history_id (str) – Encoded history ID
• workflow_name (str) – Name of the workflow to create
• job_ids (List[str]) – Optional list of job IDs to filter the jobs to extract from the history
• dataset_hids (List[str]) – Optional list of dataset hids corresponding to workflow inputs when extracting a workflow from history
• dataset_collection_hids (List[str]) – Optional list of dataset collection hids corresponding to workflow inputs when extracting a workflow from history

Return type
dict

Returns
A description of the created workflow

get_invocations(workflow_id: str) → List[Dict[str, Any]]

Get a list containing all the workflow invocations corresponding to the specified workflow.

For more advanced filtering use InvocationClient.get_invocations().

Parameters
• workflow_id (str) – Encoded workflow ID

Return type
list

Returns
A list of workflow invocations. For example:
get_workflow_inputs(workflow_id: str, label: str) → List[str]

Get a list of workflow input IDs that match the given label. If no input matches the given label, an empty list is returned.

Parameters

- workflow_id (str) – Encoded workflow ID
- label (str) – label to filter workflow inputs on

Return type

list

Returns

list of workflow inputs matching the label query

get_workflows(workflow_id: str | None = None, name: str | None = None, published: bool = False) → List[Dict[str, Any]]

Get all workflows, or select a subset by specifying optional arguments for filtering (e.g. a workflow name).

Parameters

- name (str) – Workflow name to filter on.
- published (bool) – if True, return also published workflows

Return type

list

Returns

A list of workflow dicts. For example:

```json
[
  {  'id': '92c56938c2f9b315',  'name': 'Simple',  'url': '/api/workflows/92c56938c2f9b315'  }
]
```

Changed in version 1.1.1: Using the deprecated workflow_id parameter now raises a `ValueError` exception.

import_shared_workflow(workflow_id: str) → Dict[str, Any]

Imports a new workflow from the shared published workflows.

Parameters

- workflow_id (str) – Encoded workflow ID

Return type

dict

Returns

A description of the workflow. For example:
BioBlend Documentation, Release 1.2.0

```json
{'id': 'ee0e2b4b696d9092',
 'model_class': 'StoredWorkflow',
 'name': 'Super workflow that solves everything!',
 'published': False,
 'tags': [],
 'url': '/api/workflows/ee0e2b4b696d9092'}
```

**import_workflow_dict**(*workflow_dict: Dict[str, Any], publish: bool = False*) → Dict[str, Any]

Imports a new workflow given a dictionary representing a previously exported workflow.

**Parameters**

- **workflow_dict** (*dict*) – dictionary representing the workflow to be imported
- **publish** (*bool*) – if True the uploaded workflow will be published; otherwise it will be visible only by the user which uploads it (default)

**Return type**

*dict*

**Returns**

Information about the imported workflow. For example:

```json
{'name': 'Training: 16S rRNA sequencing with mothur: main tutorial',
 'tags': [],
 'deleted': false,
 'latest_workflow_uuid': '368c6165-ccbe-4945-8a3c-d27982206d66',
 'url': '/api/workflows/94bac0a90086bdcf',
 'number_of_steps': 44,
 'published': false,
 'owner': 'jane-doe',
 'model_class': 'StoredWorkflow',
 'id': '94bac0a90086bdcf'}
```

**import_workflow_from_local_path**(*file_local_path: str, publish: bool = False*) → Dict[str, Any]

Imports a new workflow given the path to a file containing a previously exported workflow.

**Parameters**

- **file_local_path** (*str*) – File to upload to the server for new workflow
- **publish** (*bool*) – if True the uploaded workflow will be published; otherwise it will be visible only by the user which uploads it (default)

**Return type**

*dict*

**Returns**

Information about the imported workflow. For example:

```json
{'name': 'Training: 16S rRNA sequencing with mothur: main tutorial',
 'tags': [],
 'deleted': false,
 'latest_workflow_uuid': '368c6165-ccbe-4945-8a3c-d27982206d66',
 'url': '/api/workflows/94bac0a90086bdcf',
 'number_of_steps': 44,
 'published': false,
 'owner': 'jane-doe',
 'model_class': 'StoredWorkflow',
 'id': '94bac0a90086bdcf'}
```

(continues on next page)
invoke_workflow(workflow_id: str | None = None, params: dict | None = None, history_id: str | None = None, history_name: str | None = None, import_inputs_to_history: bool = False, replacement_params: dict | None = None, allow_tool_state_corrections: bool = False, inputs_by: Literal['step_index|step_uuid', 'step_index', 'step_id', 'step_uuid', 'name'] | None = None, parameters_normalized: bool = False, require_exact_tool_versions: bool = True) → Dict[str, Any]

Invoke the workflow identified by workflow_id. This will cause a workflow to be scheduled and return an object describing the workflow invocation.

Parameters

- **workflow_id** (str) – Encoded workflow ID
- **inputs** (dict) – A mapping of workflow inputs to datasets and dataset collections. The datasets source can be a LibraryDatasetDatasetAssociation (dda), LibraryDataset (d), HistoryDatasetAssociation (hda), or HistoryDatasetCollectionAssociation (hdca).

  The map must be in the following format: {'<input_index>': {'id': <encoded dataset ID>, 'src': [ldda, ld, hda, hdca]}} (e.g. {'2': {'id': '29beef4fadaed00f', 'src': 'hda'}})

  This map may also be indexed by the UUIDs of the workflow steps, as indicated by the uuid property of steps returned from the Galaxy API. Alternatively workflow steps may be addressed by the label that can be set in the workflow editor. If using uuid or label you need to also set the inputs_by parameter to step_uuid or name.
- **params** (dict) – A mapping of non-datasets tool parameters (see below)
- **history_id** (str) – The encoded history ID where to store the workflow output. Alternatively, history_name may be specified to create a new history.
- **history_name** (str) – Create a new history with the given name to store the workflow output. If both history_id and history_name are provided, history_name is ignored. If neither is specified, a new ‘Unnamed history’ is created.
- **import_inputs_to_history** (bool) – If True, used workflow inputs will be imported into the history. If False, only workflow outputs will be visible in the given history.
- **allow_tool_state_corrections** (bool) – If True, allow Galaxy to fill in missing tool state when running workflows. This may be useful for workflows using tools that have changed over time or for workflows built outside of Galaxy with only a subset of inputs defined.
- **replacement_params** (dict) – pattern-based replacements for post-job actions (see below)
- **inputs_by** (str) – Determines how inputs are referenced. Can be “step_index|step_uuid” (default), “step_index”, “step_id”, “step_uuid”, or “name”.
- **parameters_normalized** (bool) – Whether Galaxy should normalize params to ensure everything is referenced by a numeric step ID. Default is False, but when setting params for a subworkflow, True is required.
- **require_exact_tool_versions** (bool) – Whether invocation should fail if Galaxy does not have the exact tool versions. Default is True. Parameter does not any effect
for Galaxy versions < 22.05.

**Return type**

dict

**Returns**

A dict containing the workflow invocation describing the scheduling of the workflow. For example:

```json
{
    'history_id': '2f94e8ae9edff68a',
    'id': 'df7a1f0c02a5b08e',
    'inputs': '{0: {id: 'a7db2fac67043c7e',
                  src: 'hda',
                  uuid: '7932ffe0-2340-4952-8857-db3a50f1f46a'}},
    'model_class': 'WorkflowInvocation',
    'state': 'ready',
    'steps': [{'action': None,
               'id': 'd413a19dec13d11e',
               'job_id': None,
               'model_class': 'WorkflowInvocationStep',
               'order_index': 0,
               'state': None,
               'update_time': '2015-10-31T22:00:26',
               'workflow_step_id': 'cbbbf59e8f08c98c',
               'workflow_step_label': None,
               'workflow_step_uuid': 'b81250fd-3278-4e6a-b269-56a1f01e4f58'},
              {'action': None,
               'id': '2f94e8ae9edff68a',
               'job_id': 'e89067bb68beef7a0',
               'model_class': 'WorkflowInvocationStep',
               'order_index': 1,
               'state': 'new',
               'update_time': '2015-10-31T22:00:26',
               'workflow_step_id': '964b7715ec9bd22',
               'workflow_step_label': None,
               'workflow_step_uuid': 'e62440b8-e911-408b-b124-e05435d3125e'}],
    'update_time': '2015-10-31T22:00:26',
    'uuid': 'c8aa2b1c-801a-11e5-a9e5-8ca98228593c',
    'workflow_id': '03501d7626bd192f'}
```

The **params** dict should be specified as follows:

```json
{STEP_ID: PARAM_DICT, ...}
```

where **PARAM_DICT** is:

```json
{PARAM_NAME: VALUE, ...}
```

For backwards compatibility, the following (deprecated) format is also supported for **params**:

```json
{TOOL_ID: PARAM_DICT, ...}
```

in which case **PARAM_DICT** affects all steps with the given tool id. If both by-tool-id and by-step-id
specifications are used, the latter takes precedence.

Finally (again, for backwards compatibility), PARAM_DICT can also be specified as:

```python
{'param': PARAM_NAME, 'value': VALUE}
```

Note that this format allows only one parameter to be set per step.

For a repeat parameter, the names of the contained parameters needs to be specified as `<repeat name>_<repeat index>|<param name>`, with the repeat index starting at 0. For example, if the tool XML contains:

```xml
<repeat name="cutoff" title="Parameters used to filter cells" min="1">
  <param name="name" type="text" value="n_genes" label="Name of param...">
    <option value="n_genes">n_genes</option>
    <option value="n_counts">n_counts</option>
  </param>
  <param name="min" type="float" min="0" value="0" label="Min value"/>
</repeat>
```

then the PARAM_DICT should be something like:

```python
...
"cutoff_0|name": "n_genes",
"cutoff_0|min": "2",
"cutoff_1|name": "n_counts",
"cutoff_1|min": "4",
...
```

At the time of this writing, it is not possible to change the number of times the contained parameters are repeated. Therefore, the parameter indexes can go from 0 to n-1, where n is the number of times the repeated element was added when the workflow was saved in the Galaxy UI.

The replacement_params dict should map parameter names in post-job actions (PJAs) to their runtime values. For instance, if the final step has a PJA like the following:

```python
{'RenameDatasetActionout_file1': {'action_arguments': {'newname': '${output}'},
                                 'action_type': 'RenameDatasetAction',
                                 'output_name': 'out_file1'}}
```

then the following renames the output dataset to ‘foo’:

```python
replacement_params = {'output': 'foo'}
```

see also this email thread.

**Warning:** Historically, workflow invocation consumed a dataset_map data structure that was indexed by unencoded workflow step IDs. These IDs would not be stable across Galaxy instances. The new inputs property is instead indexed by either the order_index property (which is stable across workflow imports) or the step UUID which is also stable.

```python
module: str = 'workflows'

```
Refactor workflow with given actions.

Parameters

- **workflow_id** *(str)* – Encoded workflow ID
- **actions** *(list of dicts)* –

Actions to use for refactoring the workflow. The following actions are supported: `update_step_label`, `update_step_position`, `update_output_label`, `update_name`, `update_annotation`, `update_license`, `update_creator`, `update_report`, `add_step`, `add_input`, `disconnect`, `connect`, `fill_defaults`, `fill_step_defaults`, `extract_input`, `extract_legacy_parameter`, `remove_unlabeled_workflow_outputs`, `upgrade_all_steps`, `upgrade_subworkflow`, `upgrade_tool`.

An example value for the `actions` argument might be:

```python
code
actions = [
    {"action_type": "add_input", "type": "data", "label": "foo"},
    {"action_type": "update_step_label", "label": "bar", "step": { "label": "foo"}}
]
```

- **dry_run** *(bool)* – When true, perform a dry run where the existing workflow is preserved. The refactored workflow is returned in the output of the method, but not saved on the Galaxy server.

Return type
dict

Returns
Dictionary containing logged messages for the executed actions and the refactored workflow.


Execute an action for an active workflow invocation step. The nature of this action and what is expected will vary based on the the type of workflow step (the only currently valid action is True/False for pause steps).

Parameters

- **workflow_id** *(str)* – Encoded workflow ID
- **invocation_id** *(str)* – Encoded workflow invocation ID
- **step_id** *(str)* – Encoded workflow invocation step ID
- **action** *(object)* – Action to use when updating state, semantics depends on step type.

Return type
dict

Returns
Representation of the workflow invocation step

**show_invocation** *(workflow_id: str, invocation_id: str) → Dict[str, Any]*)

Get a workflow invocation object representing the scheduling of a workflow. This object may be sparse at first (missing inputs and invocation steps) and will become more populated as the workflow is actually scheduled.

Parameters
• **workflow_id** (str) – Encoded workflow ID

• **invocation_id** (str) – Encoded workflow invocation ID

**Return type**
dict

**Returns**
The workflow invocation. For example:

```json
{
  'history_id': '2f94e8ae9edff68a',
  'id': 'df7a1f0c02a5b08e',
  'inputs': {
    '0': {
      'id': 'a7db2fac67043c7e',
      'src': 'hda',
      'uuid': '7932ffe0-2340-4952-8857-db0550f1f46a'
    }
  },
  'model_class': 'WorkflowInvocation',
  'state': 'ready',
  'steps': [{
    'action': None,
    'id': 'd413a19dec13d11e',
    'job_id': None,
    'model_class': 'WorkflowInvocationStep',
    'order_index': 0,
    'state': None,
    'update_time': '2015-10-31T22:00:26',
    'workflow_step_id': 'cbbbf59e8f08c98c',
    'workflow_step_label': None,
    'workflow_step_uuid': 'b81250fd-3278-4e6a-b269-56a1f01ef485'},
  {
    'action': None,
    'id': '2f94e8ae9edff68a',
    'job_id': 'e89067bb68be7a0',
    'model_class': 'WorkflowInvocationStep',
    'order_index': 1,
    'state': 'new',
    'update_time': '2015-10-31T22:00:26',
    'workflow_step_id': '964b37715ec9bd22',
    'workflow_step_label': None,
    'workflow_step_uuid': 'e62440b8-e911-408b-b124-e05435d3125e'
  }],
  'update_time': '2015-10-31T22:00:26',
  'uuid': 'c8aa2b1c-801a-11e5-a9e5-8ca9828593c',
  'workflow_id': '03501d7626bd192f'
}
```


See the details of a particular workflow invocation step.

**Parameters**

• **workflow_id** (str) – Encoded workflow ID

• **invocation_id** (str) – Encoded workflow invocation ID

• **step_id** (str) – Encoded workflow invocation step ID

**Return type**
dict

**Returns**
The workflow invocation step. For example:

```
{"action": None,
'id': '63cd3858d057a6d1',
'job_id': None,
'model_class': 'WorkflowInvocationStep',
'order_index': 2,
'state': None,
'update_time': '2015-10-31T22:11:14',
'workflow_step_id': '52e496b945151ee8',
'workflow_step_label': None,
'workflow_step_uuid': '4060554c-1dd5-4287-9040-8b4f281cf9dc'}
```

**show_versions**(*workflow_id: str*) → List[Dict[str, Any]]

Get versions for a workflow.

**Parameters**

- **workflow_id**: str – Encoded workflow ID

**Return type**

list of dicts

**Returns**

Ordered list of version descriptions for this workflow

**show_workflow**(*workflow_id: str, version: int | None = None*) → Dict[str, Any]

Display information needed to run a workflow.

**Parameters**

- **workflow_id**: str – Encoded workflow ID
- **version**: int – Workflow version to show

**Return type**

dict

**Returns**

A description of the workflow and its inputs. For example:

```
{"id": '92c56938c2f9b315',
'inputs': {'23': {'label': 'Input Dataset', 'value': ''},
'name': 'Simple',
'url': '/api/workflows/92c56938c2f9b315'}
```

**update_workflow**(*workflow_id: str, **kwargs: Any*) → Dict[str, Any]

Update a given workflow.

**Parameters**

- **workflow_id**: str – Encoded workflow ID
- **kwargs**: Any – dictionary representing the workflow to be updated
  - **name**: str – New name of the workflow
  - **annotation**: str – New annotation for the workflow
  - **menu_entry**: bool – Whether the workflow should appear in the user’s menu
  - **tags**: list of str – Replace workflow tags with the given list
BioBlend Documentation, Release 1.2.0

- **published** *(bool)* – Whether the workflow should be published or unpublished

**Return type**

dict

**Returns**

Dictionary representing the updated workflow

### 5.1.2 Object-oriented Galaxy API

**class** `bioblend.galaxy.objects.galaxy_instance.GalaxyInstance` *(url: str, api_key: str | None = None, email: str | None = None, password: str | None = None, *, verify: bool = True)*

A representation of an instance of Galaxy, identified by a URL and a user’s API key.

**Parameters**

- **url** *(str)* – a FQDN or IP for a given instance of Galaxy. For example: `http://127.0.1:8080`
- **api_key** *(str)* – user’s API key for the given instance of Galaxy, obtained from the Galaxy web UI.

This is actually a factory class which instantiates the entity-specific clients.

Example: get a list of all histories for a user with API key ‘foo’:

```python
from bioblend.galaxy.objects import GalaxyInstance
gi = GalaxyInstance('http://127.0.0.1:8080', api_key='foo')
histories = gi.histories.list()
```

**Client**

Clients for interacting with specific Galaxy entity types.

Classes in this module should not be instantiated directly, but used via their handles in `GalaxyInstance`.

**class** `bioblend.galaxy.objects.client.ObjClient` *(obj_gi: GalaxyInstance)*

**abstract get** *(id_: str) -> Wrapper*

Retrieve the object corresponding to the given id.

**abstract get_previews** *(**kwargs: Any) -> list*

Get a list of object previews.

Previews entity summaries provided by REST collection URIs, e.g. `http://host:port/api/libraries`. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

**Return type**

list

**Returns**

a list of object previews
abstract list() → list
Get a list of objects.
This method first gets the entity summaries, then gets the complete description for each entity with an additional GET call, so may be slow.

Return type
list

Returns
a list of objects

class bioblend.galaxy.objects.client.ObjDatasetClient(obj_gi: GalaxyInstance)
Interacts with Galaxy datasets.

get(id_: str, hda_ldda: Literal['hda']) = 'hda') → HistoryDatasetAssociation
get(id_: str, hda_ldda: Literal['ldda']) 
Retrieve the dataset corresponding to the given id.

Parameters
hda_ldda (str) – Whether to show a history dataset ('hda' - the default) or library dataset ('ldda')

Return type
HistoryDatasetAssociation or LibraryDatasetDatasetAssociation

Returns
the history or library dataset corresponding to id_

get_previews(**kwargs: Any) → list
Get a list of object previews.
Previews entity summaries provided by REST collection URIs, e.g. http://host:port/api/libraries. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

Return type
list

Returns
a list of object previews

list() → list
Get a list of objects.
This method first gets the entity summaries, then gets the complete description for each entity with an additional GET call, so may be slow.

Return type
list

Returns
a list of objects

class bioblend.galaxy.objects.client.ObjDatasetCollectionClient(obj_gi: GalaxyInstance)
Interacts with Galaxy dataset collections.

get(id_: str) → HistoryDatasetCollectionAssociation
Retrieve the dataset collection corresponding to the given id.

Return type
HistoryDatasetCollectionAssociation

5.1. Galaxy API
Returns
the history dataset collection corresponding to \texttt{id}.

\texttt{get\_previews(**kwargs: Any) \rightarrow list}

Get a list of object previews.

Previews entity summaries provided by REST collection URIs, e.g. http://host:port/api/libraries. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

Return type
list

Returns
a list of object previews

\texttt{list() \rightarrow list}

Get a list of objects.

This method first gets the entity summaries, then gets the complete description for each entity with an additional GET call, so may be slow.

Return type
list

Returns
a list of objects

\texttt{class bioblend.galaxy.objects.client.ObjDatasetContainerClient(obj\_gi: GalaxyInstance)}

\texttt{CONTAINER\_PREVIEW\_TYPE: Type[DatasetContainerPreview]}

\texttt{CONTAINER\_TYPE: Type[DatasetContainer]}

\texttt{get(id_: str) \rightarrow DatasetContainerSubtype}

Retrieve the dataset container corresponding to the given id.

\texttt{get\_previews(name: str | None = None, deleted: bool = False, **kwargs: Any) \rightarrow List[DatasetContainerPreviewSubtype]}

Get a list of object previews.

Previews entity summaries provided by REST collection URIs, e.g. http://host:port/api/libraries. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

Return type
list

Returns
a list of object previews

\texttt{class bioblend.galaxy.objects.client.ObjHistoryClient(obj\_gi: GalaxyInstance)}

Interacts with Galaxy histories.

\texttt{CONTAINER\_PREVIEW\_TYPE}
alias of \texttt{HistoryPreview}

\texttt{CONTAINER\_TYPE}
alias of \texttt{History}
**create**(*name: str | None = None*) ➔ *History*
Create a new Galaxy history, optionally setting its name.

**Return type**

*History*

**Returns**

the history just created

**delete**(*id_: str | None = None, name: str | None = None, purge: bool = False*) ➔ None
Delete the history with the given id or name.

Fails if multiple histories have the same name.

**Parameters**

- **purge (bool)** – if True, also purge (permanently delete) the history

**Note:** The purge option works only if the Galaxy instance has the `allow_user_dataset_purge` option set to true in the `config/galaxy.yml` configuration file.

**list**(*name: str | None = None, deleted: bool = False*) ➔ List[*History*]
Get histories owned by the user of this Galaxy instance.

**Parameters**

- **name (str)** – return only histories with this name
- **deleted (bool)** – if True, return histories that have been deleted

**Return type**

List of *History*

---

**class** bioblend.galaxy.objects.client.ObjInvocationClient(*obj_gi: GalaxyInstance*)
Interacts with Galaxy Invocations.

**get**(*id_: str*) ➔ Invocation
Get an invocation by ID.

**Return type**

Invocation

**Param**

invocation object

**get_previews**(**kwargs: Any**) ➔ List[InvocationPreview]
Get previews of all invocations.

**Return type**

List of InvocationPreview

**Param**

previews of invocations

**list**(*workflow: Workflow | None = None, history: History | None = None, include_terminal: bool = True, limit: int | None = None*) ➔ List[Invocation]
Get full listing of workflow invocations, or select a subset by specifying optional arguments for filtering (e.g. a workflow).

**Parameters**
• **workflow** ([wrappers.Workflow](#)) – Include only invocations associated with this workflow

• **history** ([wrappers.History](#)) – Include only invocations associated with this history

• **include_terminal** – bool

• **limit** (int) – Maximum number of invocations to return - if specified, the most recent invocations will be returned.

**Param**
Whether to include invocations in terminal states

**Return type**
list of Invocation

**Param**
invocation objects

class bioblend.galaxy.objects.client.ObjJobClient(obj_gi: GalaxyInstance)
Interacts with Galaxy jobs.

def get(id_: str, full_details: bool = False) -> Job
Retrieve the job corresponding to the given id.

**Parameters**

- **full_details** (bool) – if True, return the complete list of details for the given job.

**Return type**
Job

**Returns**
the job corresponding to id_

def get_previews(**kwargs: Any) -> List[JobPreview]
Get a list of object previews.

Previews entity summaries provided by REST collection URIs, e.g. http://host:port/api/libraries. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

**Return type**
list

**Returns**
a list of object previews

def list() -> List[Job]
Get the list of jobs of the current user.

**Return type**
list of Job

class bioblend.galaxy.objects.client.ObjLibraryClient(obj_gi: GalaxyInstance)
Interacts with Galaxy libraries.

CONTAINER_PREVIEW_TYPE
alias of LibraryPreview

CONTAINER_TYPE
alias of Library
create (name: str, description: str | None = None, synopsis: str | None = None) → Library
Create a data library with the properties defined in the arguments.

Return type
Library

Returns
the library just created

delete (id_: str | None = None, name: str | None = None) → None
Delete the library with the given id or name.
Fails if multiple libraries have the specified name.

Warning: Deleting a data library is irreversible - all of the data from the library will be permanently deleted.

list (name: str | None = None, deleted: bool = False) → List[Library]
Get libraries owned by the user of this Galaxy instance.

Parameters
• name (str) – return only libraries with this name
• deleted (bool) – if True, return libraries that have been deleted

Return type
list of Library

class bioblend.galaxy.objects.client.OgjToolClient (obj_gi: GalaxyInstance)
Interacts with Galaxy tools.

get (id_: str, io_details: bool = False, link_details: bool = False) → Tool
Retrieve the tool corresponding to the given id.

Parameters
• io_details (bool) – if True, get also input and output details
• link_details (bool) – if True, get also link details

Return type
Tool

Returns
the tool corresponding to id_

get_previews (name: str | None = None, trackster: bool = False, **kwargs: Any) → List[Tool]
Get the list of tools installed on the Galaxy instance.

Parameters
• name (str) – return only tools with this name
• trackster (bool) – if True, only tools that are compatible with Trackster are returned

Return type
list of Tool
**list** *(name: str | None = None, trackster: bool = False) → List[Tool]*  
Get the list of tools installed on the Galaxy instance.

**Parameters**

- **name** *(str)* – return only tools with this name
- **trackster** *(bool)* – if True, only tools that are compatible with Trackster are returned

**Return type**

list of Tool

**class** bioblend.galaxy.objects.client.ObjWorkflowClient*(obj_gi: GalaxyInstance)*  
Interacts with Galaxy workflows.

**delete** *(id_: str | None = None, name: str | None = None) → None*  
Delete the workflow with the given id or name.

Fails if multiple workflows have the specified name.

**Warning:** Deleting a workflow is irreversible - all of the data from the workflow will be permanently deleted.

**get** *(id_: str) → Workflow*  
Retrieve the workflow corresponding to the given id.

**Return type**

Workflow

**Returns**

the workflow corresponding to id_

**get_previews** *(name: str | None = None, published: bool = False, **kwargs: Any) → List[WorkflowPreview]*  
Get a list of object previews.

Previews entity summaries provided by REST collection URIs, e.g. http://host:port/api/libraries. Being the most lightweight objects associated to the various entities, these are the ones that should be used to retrieve their basic info.

**Return type**

list

**Returns**

a list of object previews

**import_new** *(src: str | Dict[str, Any], publish: bool = False) → Workflow*  
Imports a new workflow into Galaxy.

**Parameters**

- **src** *(dict or str)* – deserialized (dictionary) or serialized (str) JSON dump of the workflow (this is normally obtained by exporting a workflow from Galaxy).
- **publish** *(bool)* – if True the uploaded workflow will be published; otherwise it will be visible only by the user which uploads it (default).

**Return type**

Workflow

**Returns**

the workflow just imported
**import_shared(id_: str) → Workflow**
Imports a shared workflow to the user’s space.

**Parameters**
- **id** (str) – workflow id

**Return type**
Workflow

**Returns**
the workflow just imported

**list(name: str | None = None, published: bool = False) → List[Workflow]**
Get workflows owned by the user of this Galaxy instance.

**Parameters**
- **name** (str) – return only workflows with this name
- **published** (bool) – if True, return also published workflows

**Return type**
list of Workflow

---

**Wrappers**
A basic object-oriented interface for Galaxy entities.

```python
class bioblend.galaxy.objects.wrappers.Dataset(*args: Any, **kwargs: Any)
```
Abstract base class for Galaxy datasets.

**Parameters**
- **wrapped** (dict) – JSON-serializable dictionary
- **parent** (Wrapper) – the parent of this wrapper
- **gi** (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

**BASE_ATTRS**: Tuple[str, ...] = ('id', 'data_type', 'file_ext', 'file_name', 'file_size', 'genome_build', 'misc_info', 'name', 'state')

**POLLING_INTERVAL** = 1

**container**: DatasetContainer

**download(file_object: IO[bytes], chunk_size: int = 4096) → None**
Open dataset for reading and save its contents to file_object.

**Parameters**
- **file_object** (file) – output file object

See get_stream() for info on other params.

**genome_build**: str

**get_contents(chunk_size: int = 4096) → bytes**
Open dataset for reading and return its full contents.

See get_stream() for param info.
get_stream(chunk_size: int = 4096) → Iterator[bytes]
Open dataset for reading and return an iterator over its contents.

Parameters
chunk_size (int) – read this amount of bytes at a time

gi: GalaxyInstance
misc_info: str
name: str

peek(chunk_size: int = 4096) → bytes
Open dataset for reading and return the first chunk.
See get_stream() for param info.

refresh() → DatasetSubtype
Re-fetch the attributes pertaining to this object.

Returns
self

state: str

wait(polling_interval: float = 1, break_on_error: bool = True) → None
Wait for this dataset to come out of the pending states.

Parameters
• polling_interval (float) – polling interval in seconds
• break_on_error (bool) – if True, raise a RuntimeError exception if the dataset ends in
  the ‘error’ state.

Warning: This is a blocking operation that can take a very long time. Also, note that this method
does not return anything; however, this dataset is refreshed (possibly multiple times) during the execution.

class bioblend.galaxy.objects.wrappers.DatasetCollection(*args: Any, **kwargs: Any)
Abstract base class for Galaxy dataset collections.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

API_MODULE = 'dataset_collections'
BASE_ATTRS: Tuple[str, ...] = ('id', 'collection_type', 'deleted', 'name', 'state')

collection_type: str

container: DatasetCollection | History

abstract delete() → None
Delete this dataset collection.
deleted: bool  

gi: GalaxyInstance  

refresh() → DatasetCollectionSubtype  
Re-fetch the attributes pertaining to this object.

Returns  
self

class bioblend.galaxy.objects.wrappers.DatasetContainer(*args: Any, **kwargs: Any)  
Abstract base class for dataset containers (histories and libraries).

Parameters  
content_infos (list of ContentInfo) – info objects for the container’s contents

API_MODULE: str  
BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'name')  
CONTENT_INFO_TYPE: Type[ContentInfo]  
DS_TYPE: ClassVar[Callable]  
content_infos: List[ContentInfo]  

property dataset_ids: List[str]  
Return the ids of the contained datasets.

abstract delete() → None  
Delete this dataset container.

deleted: bool

get_dataset(ds_id: str) → DatasetSubtype  
Retrieve the dataset corresponding to the given id.

Parameters  

ds_id (str) – dataset id

Return type  
HistoryDatasetAssociation or LibraryDataset

Returns  
the dataset corresponding to ds_id

get_datasets(name: str | None = None) → List[DatasetSubtype]  
Get all datasets contained inside this dataset container.

Parameters  

name (str) – return only datasets with this name

Return type  
list of HistoryDatasetAssociation or list of LibraryDataset

Returns  
datasets with the given name contained inside this container

5.1. Galaxy API
Note: when filtering library datasets by name, specify their full paths starting from the library’s root folder, e.g., /seqdata/reads.fastq. Full paths are available through the content_infos attribute of Library objects.

```
gi:  GalaxyInstance
name:  str

obj_gi_client:  client.ObjDatasetContainerClient
preview() → DatasetContainerPreview
refresh() → DatasetContainerSubtype

    Re-fetch the attributes pertaining to this object.

    Returns
    self
```

```
class bioblend.galaxy.objects.wrappers.Folder(*args: Any, **kwargs: Any)
Maps to a folder in a Galaxy library.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'description', 'item_count', 'name')

container:  Library
description:  str
gi:  GalaxyInstance
name:  str

property parent:  Folder | None

    The parent folder of this folder. The parent of the root folder is None.

    Return type
    Folder

    Returns
    the parent of this folder

refresh() → Folder

    Re-fetch the attributes pertaining to this object.

    Returns
    self
```

```
class bioblend.galaxy.objects.wrappers.History(*args: Any, **kwargs: Any)
Maps to a Galaxy history.

Parameters
• content_infos (list of ContentInfo) – info objects for the container’s contents
API_MODULE: str = 'histories'

BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'name', 'annotation', 'published', 'state', 'state_ids', 'state_details', 'tags')

CONTENT_INFO_TYPE
    alias of HistoryContentInfo

DSC_TYPE
    alias of HistoryDatasetCollectionAssociation

DS_TYPE
    alias of HistoryDatasetAssociation

annotation: str

create_dataset_collection(collection_description: CollectionDescription, copy_elements: bool = True) → HistoryDatasetCollectionAssociation

Create a new dataset collection in the history by providing a collection description.

Parameters
    • collection_description (bioblend.galaxy.dataset_collections.CollectionDescription) – a description of the dataset collection
    • copy_elements (bool) – Whether to make a copy of the elements of the collection being created

Return type
    HistoryDatasetCollectionAssociation

Returns
    the new dataset collection

delete(purge: bool = False) → None

Delete this history.

Parameters
    purge (bool) – if True, also purge (permanently delete) the history

Note: The purge option works only if the Galaxy instance has the allow_user_dataset_purge option set to true in the config/galaxy.yml configuration file.

download(jeha_id: str, outf: IO[bytes], chunk_size: int = 4096) → None

Download an export archive for this history. Use export() to create an export and get the required jeha_id. See download_history() for parameter and return value info.

export(gzip: bool = True, include_hidden: bool = False, include_deleted: bool = False, wait: bool = False, maxwait: int | None = None) → str

Start a job to create an export archive for this history. See export_history() for parameter and return value info.

get_dataset_collection(dsc_id: str) → HistoryDatasetCollectionAssociation

Retrieve the dataset collection corresponding to the given id.

Parameters
    dsc_id (str) – dataset collection id
Return type

`HistoryDatasetCollectionAssociation`

Returns

the dataset collection corresponding to `dsc_id`

`import_dataset(lds: LibraryDataset) → HistoryDatasetAssociation`

Import a dataset into the history from a library.

Parameters

- `lds`: `LibraryDataset` – the library dataset to import

Return type

`HistoryDatasetAssociation`

Returns

the imported history dataset

`paste_content(content: str, **kwargs: Any) → HistoryDatasetAssociation`

Upload a string to a new dataset in this history.

Parameters

- `content`: `str` – content of the new dataset to upload

See `upload_file()` for the optional parameters (except file_name).

Return type

`HistoryDatasetAssociation`

Returns

the uploaded dataset

`published: bool`

`tags: List[str]`

`update(**kwargs: Any) → History`

Update history metadata information. Some of the attributes that can be modified are documented below.

Parameters

- `name`: `str` – Replace history name with the given string
- `annotation`: `str` – Replace history annotation with the given string
- `deleted`: `bool` – Mark or unmark history as deleted
- `purged`: `bool` – If True, mark history as purged (permanently deleted).
- `published`: `bool` – Mark or unmark history as published
- `importable`: `bool` – Mark or unmark history as importable
- `tags`: `List[str]` – Replace history tags with the given list

`upload_dataset(path: str, **kwargs: Any) → HistoryDatasetAssociation`

Upload the file specified by `path` to this history.

Parameters

- `path`: `str` – path of the file to upload

See `upload_file()` for the optional parameters.

Return type

`HistoryDatasetAssociation`
Returns the uploaded dataset

**upload_file**(*path: str, **kwargs: Any*) → *HistoryDatasetAssociation*

Upload the file specified by path to this history.

**Parameters**

- **path (str)** – path of the file to upload

See **upload_file()** for the optional parameters.

**Return type**

*HistoryDatasetAssociation*

Returns the uploaded dataset

**upload_from_ftp**(*path: str, **kwargs: Any*) → *HistoryDatasetAssociation*

Upload the file specified by path from the user’s FTP directory to this history.

**Parameters**

- **path (str)** – path of the file in the user’s FTP directory

See **upload_file()** for the optional parameters.

**Return type**

*HistoryDatasetAssociation*

Returns the uploaded dataset

```python
class bioblend.galaxy.objects.wrappers.HistoryContentInfo(*args: Any, **kwargs: Any)

Instances of this class wrap dictionaries obtained by getting /api/histories/<ID>/contents from Galaxy.

**Parameters**

- **wrapped (dict)** – JSON-serializable dictionary
- **parent (Wrapper)** – the parent of this wrapper
- **gi (GalaxyInstance)** – the GalaxyInstance through which we can access this wrapper

**BASE_ATTRS**: Tuple[str, ...] = ('id', 'name', 'type', 'deleted', 'state', 'visible')
```

```python
class bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation(*args: Any, **kwargs: Any)

Maps to a Galaxy HistoryDatasetAssociation.

**Parameters**

- **wrapped (dict)** – JSON-serializable dictionary
- **parent (Wrapper)** – the parent of this wrapper
- **gi (GalaxyInstance)** – the GalaxyInstance through which we can access this wrapper

**BASE_ATTRS**: Tuple[str, ...] = ('id', 'data_type', 'file_ext', 'file_name', 'file_size', 'genome_build', 'misc_info', 'name', 'state', 'annotation', 'deleted', 'purged', 'tags', 'visible')
```

**SRC** = 'hda'

**annotation**: str
delete(purge: bool = False) → None
Delete this history dataset.

Parameters
purge (bool) – if True, also purge (permanently delete) the dataset

Note: The purge option works only if the Galaxy instance has the allow_user_dataset_purge option set to true in the config/galaxy.yml configuration file.

deleted:  bool
get_stream(chunk_size: int = 4096) → Iterator[bytes]
Open dataset for reading and return an iterator over its contents.

Parameters
chunk_size (int) – read this amount of bytes at a time

purged:  bool
update(**kwargs: Any) → HistoryDatasetAssociation
Update this history dataset metadata. Some of the attributes that can be modified are documented below.

Parameters
• name (str) – Replace history dataset name with the given string
• genome_build (str) – Replace history dataset genome build (dbkey)
• annotation (str) – Replace history dataset annotation with given string
• deleted (bool) – Mark or unmark history dataset as deleted
• visible (bool) – Mark or unmark history dataset as visible

class bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation(*args: Any, **kwargs: Any)
Maps to a Galaxy HistoryDatasetCollectionAssociation.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'collection_type', 'deleted', 'name', 'state', 'tags', 'visible', 'elements')

SRC = 'hdca'
delete() → None
Delete this dataset collection.

elements:  List[Dict]
class bioblend.galaxy.objects.wrappers.HistoryPreview(*args: Any, **kwargs: Any)
Models Galaxy history ‘previews’.

Instances of this class wrap dictionaries obtained by getting /api/histories from Galaxy.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'name', 'annotation', 'published', 'purged', 'tags')

```python
class bioblend.galaxy.objects.wrappers.Job(*args: Any, **kwargs: Any)
Maps to a Galaxy job.

    Parameters

    • wrapped (dict) – JSON-serializable dictionary
    • parent (Wrapper) – the parent of this wrapper
    • gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

    BASE_ATTRS: Tuple[str, ...] = ('id', 'state')
```

```python
class bioblend.galaxy.objects.wrappers.Library(*args: Any, **kwargs: Any)
Maps to a Galaxy library.

    Parameters

    content_infos (list of ContentInfo) – info objects for the container's contents

    API_MODULE: str = 'libraries'

    BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'name', 'description', 'synopsis')

    CONTENT_INFO_TYPE
    alias of LibraryContentInfo

    DS_TYPE
    alias of LibraryDataset

    copy_from_dataset(hda: HistoryDatasetAssociation, folder: Folder | None = None, message: str = '') -> LibraryDataset
    Copy a history dataset into this library.

    Parameters

    hda (HistoryDatasetAssociation) – history dataset to copy into the library

    See upload_data() for info on other params.

    create_folder(name: str, description: str | None = None, base_folder: Folder | None = None) -> Folder
    Create a folder in this library.

    Parameters

    • name (str) – folder name
    • description (str) – optional folder description
    • base_folder (Folder) – parent folder, or None to create in the root folder

    Return type
    Folder

    Returns
    the folder just created
```

### 5.1. Galaxy API
**Description**: str

**Property folder_ids**: List[str]

Return the ids of the contained folders.

**get_folder**(*f_id*: str) → Folder

Retrieves the folder corresponding to the given id.

**Return type**

Folder

**Returns**

the folder corresponding to *f_id*

**Property root_folder**: Folder

The root folder of this library.

**Return type**

Folder

**Returns**

the root folder of this library

**Synopsis**: str

**upload_data**(*data*: str, *folder*: Folder | None = None, **kwargs: Any) → LibraryDataset

Upload data to this library.

**Parameters**

- *data* (str) – dataset contents
- *folder* (Folder) – a folder object, or None to upload to the root folder

**Return type**

LibraryDataset

**Returns**

the dataset object that represents the uploaded content

Optional keyword arguments: file_type, dbkey.

**upload_from_galaxy_fs**(*paths*: str | Iterable[str], *folder*: Folder | None = None, *link_data_only*: Literal['copy_files', 'link_to_files'] = 'copy_files', **kwargs: Any) → List[LibraryDataset]

Upload data to this library from filesystem paths on the server.

**Parameters**

- *paths* (str or Iterable of str) – server-side file paths from which data should be read
- *link_data_only* (str) – either ‘copy_files’ (default) or ‘link_to_files’. Setting to ‘link_to_files’ symlinks instead of copying the files

**Return type**

list of LibraryDataset

**Returns**

the dataset objects that represent the uploaded content
See upload_data() for info on other params.

Note: This method works only if the Galaxy instance has the allow_path_paste option set to true in the config/galaxy.yml configuration file.

upload_from_local(path: str, folder: Folder | None = None, **kwargs: Any) → LibraryDataset
Upload data to this library from a local file.

Parameters
path (str) – local file path from which data should be read

See upload_data() for info on other params.

upload_from_url(url: str, folder: Folder | None = None, **kwargs: Any) → LibraryDataset
Upload data to this library from the given URL.

Parameters
url (str) – URL from which data should be read

See upload_data() for info on other params.

class bioblend.galaxy.objects.wrappers.LibraryContentInfo(*args: Any, **kwargs: Any)
Instances of this class wrap dictionaries obtained by getting /api/libraries/<ID>/contents from Galaxy.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

class bioblend.galaxy.objects.wrappers.LibraryDataset(*args: Any, **kwargs: Any)
Maps to a Galaxy LibraryDataset.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

SRC = 'ld'

delete(purged: bool = False) → None
Delete this library dataset.

Parameters
purged (bool) – if True, also purge (permanently delete) the dataset

file_name: str

update(**kwargs: Any) → LibraryDataset
Update this library dataset metadata. Some of the attributes that can be modified are documented below.

Parameters
• name (str) – Replace history dataset name with the given string
• genome_build (str) – Replace history dataset genome build (dbkey)
class bioblend.galaxy.objects.wrappers.LibraryDatasetDatasetAssociation(*args: Any, **kwargs: Any)

Maps to a Galaxy LibraryDatasetDatasetAssociation.

Parameters

- `wrapped (dict)` – JSON-serializable dictionary
- `parent (Wrapper)` – the parent of this wrapper
- `gi (GalaxyInstance)` – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'data_type', 'file_ext', 'file_name', 'file_size', 'genome_build', 'misc_info', 'name', 'state', 'deleted')

SRC = 'ldda'

class bioblend.galaxy.objects.wrappers.LibraryPreview(*args: Any, **kwargs: Any)

Models Galaxy library ‘previews’.

Instances of this class wrap dictionaries obtained by getting /api/libraries from Galaxy.

Parameters

- `wrapped (dict)` – JSON-serializable dictionary
- `parent (Wrapper)` – the parent of this wrapper
- `gi (GalaxyInstance)` – the GalaxyInstance through which we can access this wrapper

class bioblend.galaxy.objects.wrappers.Step(*args: Any, **kwargs: Any)

Workflow step.

Steps are the main building blocks of a Galaxy workflow. A step can be: an input (type data_collection_input, data_input or parameter_input), a computational tool (type tool), a sub-workflow (type subworkflow) or a pause (type pause).

Parameters

- `wrapped (dict)` – JSON-serializable dictionary
- `parent (Wrapper)` – the parent of this wrapper
- `gi (GalaxyInstance)` – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'input_steps', 'name', 'tool_id', 'tool_inputs', 'tool_version', 'type')

    input_steps: Dict[str, Dict]
    tool_id: str | None
    tool_inputs: Dict
    tool_version: str | None
    type: str

class bioblend.galaxy.objects.wrappers.Tool(*args: Any, **kwargs: Any)

Maps to a Galaxy tool.

Parameters

- `wrapped (dict)` – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'name', 'version')

POLLOUNG_INTERVAL = 10

gi: GalaxyInstance

run(inputs: Dict[str, Any], history: History, wait: bool = False, polling_interval: float = 10) -> List[HistoryDatasetAssociation]

Execute this tool in the given history with inputs from dict inputs.

Parameters
• inputs (dict) – dictionary of input datasets and parameters for the tool (see below)
• history (History) – the history where to execute the tool
• wait (bool) – whether to wait while the returned datasets are in a pending state
• polling_interval (float) – polling interval in seconds

Return type
list of HistoryDatasetAssociation

Returns
list of output datasets

The inputs dict should contain input datasets and parameters in the (largely undocumented) format used by the Galaxy API. Some examples can be found in Galaxy’s API test suite. The value of an input dataset can also be a Dataset object, which will be automatically converted to the needed format.

class bioblend.galaxy.objects.wrappers.Workflow(*args: Any, **kwargs: Any)

Workflows represent ordered sequences of computations on Galaxy.

A workflow defines a sequence of steps that produce one or more results from an input dataset.

Parameters
• wrapped (dict) – JSON-serializable dictionary
• parent (Wrapper) – the parent of this wrapper
• gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'inputs', 'latest_workflow_uuuid', 'name', 'owner', 'published', 'steps', 'tags')

POLLOUNG_INTERVAL = 10

dag: Dict[str, Set[str]]

property data_collection_input_ids: Set[str]

Return the ids of data collection input steps for this workflow.

property data_input_ids: Set[str]

Return the ids of data input steps for this workflow.
delete() → None
Delete this workflow.

**Warning:** Deleting a workflow is irreversible - all of the data from the workflow will be permanently deleted.

deleted: bool
export() → Dict[str, Any]
Export a re-importable representation of the workflow.

Return type
dict

Returns
a JSON-serializable dump of the workflow

property input_labels: Set[str]
Return the labels of this workflow’s input steps.

input_labels_to_ids: Dict[str, Set[str]]
inputs: Dict[str, Dict]
inv_dag: Dict[str, Set[str]]

invoke(inputs: Dict[str, Any] | None = None, params: Dict[str, Any] | None = None, history: History | None = None, import_inputs_to_history: bool = False, replacement_params: Dict[str, Any] | None = None, step_index | step_uuid, step_index | step_id | step_uuid, name | None = None, parameters_normalized: bool = False) → Invocation
Invoke the workflow. This will cause a workflow to be scheduled and return an object describing the workflow invocation.

Parameters

• **inputs** (dict) – A mapping of workflow inputs to datasets and dataset collections. The datasets source can be a LibraryDatasetDatasetAssociation (ldda), LibraryDataset (ld), HistoryDatasetAssociation (hda), or HistoryDatasetCollectionAssociation (hdca).

The map must be in the following format: {'<input_index>': dataset or collection} (e.g. {'2': HistoryDatasetAssociation()})

This map may also be indexed by the UUIDs of the workflow steps, as indicated by the uuid property of steps returned from the Galaxy API. Alternatively workflow steps may be addressed by the label that can be set in the workflow editor. If using uuid or label you need to also set the inputs_by parameter to step_uuid or name.

• **params** (dict) – A mapping of non-datasets tool parameters (see below)

• **history**(History or str) – The history in which to store the workflow output, or the name of a new history to create. If None, a new ‘Unnamed history’ is created.

• **import_inputs_to_history**(bool) – If True, used workflow inputs will be imported into the history. If False, only workflow outputs will be visible in the given history.

• **allow_tool_state_corrections**(bool) – If True, allow Galaxy to fill in missing tool state when running workflows. This may be useful for workflows using tools that have
changed over time or for workflows built outside of Galaxy with only a subset of inputs defined.

- **replacement_params** (*dict*) – pattern-based replacements for post-job actions (see below)
- **inputs_by** (*str*) – Determines how inputs are referenced. Can be “step_index|step_uuid” (default), “step_index”, “step_id”, “step_uuid”, or “name”.
- **parameters_normalized** (*bool*) – Whether Galaxy should normalize params to ensure everything is referenced by a numeric step ID. Default is False, but when setting params for a subworkflow, True is required.

**Return type**

Invocation

**Returns**

the workflow invocation

The params dict should be specified as follows:

```
{STEP_ID: PARAM_DICT, ...}
```

where PARAM_DICT is:

```
{PARAM_NAME: VALUE, ...}
```

For backwards compatibility, the following (deprecated) format is also supported for params:

```
{TOOL_ID: PARAM_DICT, ...}
```

in which case PARAM_DICT affects all steps with the given tool id. If both by-tool-id and by-step-id specifications are used, the latter takes precedence.

Finally (again, for backwards compatibility), PARAM_DICT can also be specified as:

```
{param: PARAM_NAME, 'value': VALUE}
```

Note that this format allows only one parameter to be set per step.

For a repeat parameter, the names of the contained parameters needs to be specified as `<repeat name>_<repeat index>|<param name>`, with the repeat index starting at 0. For example, if the tool XML contains:

```
<repeat name="cutoff" title="Parameters used to filter cells" min="1">
   <param name="name" type="text" value="n_genes" label="Name of param...">n_genes</param>
   <param name="min" type="float" min="0" value="0" label="Min value"/>
</repeat>
```

then the PARAM_DICT should be something like:

```
{...
   "cutoff_0|name": "n_genes",
   "cutoff_0|min": "2",
   "cutoff_1|name": "n_counts",
```

(continues on next page)
At the time of this writing, it is not possible to change the number of times the contained parameters are repeated. Therefore, the parameter indexes can go from 0 to n-1, where n is the number of times the repeated element was added when the workflow was saved in the Galaxy UI.

The `replacement_params` dict should map parameter names in post-job actions (PJAs) to their runtime values. For instance, if the final step has a PJA like the following:

```json
{ 'RenameDatasetActionout_file1': { 'action_arguments': { 'newname': '${output}' },
  'action_type': 'RenameDatasetAction',
  'output_name': 'out_file1' }
}
```

then the following renames the output dataset to ‘foo’:

```python
replacement_params = { 'output': 'foo' }
```

see also this email thread.

**Warning:** Historically, workflow invocation consumed a `dataset_map` data structure that was indexed by unencoded workflow step IDs. These IDs would not be stable across Galaxy instances. The new `inputs` property is instead indexed by either the `order_index` property (which is stable across workflow imports) or the step UUID which is also stable.

**property isRunnable:** bool

Return True if the workflow can be run on Galaxy.

A workflow is considered runnable on a Galaxy instance if all of the tools it uses are installed in that instance.

**missing_ids:** List

**name:** str

**owner:** str

**property parameter_input_ids:** Set[str]

Return the ids of parameter input steps for this workflow.

**preview()** → `WorkflowPreview`

**published:** bool

**sink_ids:** Set[str]

**sorted_step_ids()** → List[str]

Return a topological sort of the workflow’s DAG.

**source_ids:** Set[str]

**steps:** Dict[str, `Step`]

**tags:** List[str]
property tool_ids: Set[str]
    Return the ids of tool steps for this workflow.

tool_labels_to_ids: Dict[str, Set[str]]

class bioblend.galaxy.objects.wrappers.WorkflowPreview(*args: Any, **kwargs: Any)
    Models Galaxy workflow ‘previews’.
    Instances of this class wrap dictionaries obtained by getting /api/workflows from Galaxy.

    Parameters
    • wrapped (dict) – JSON-serializable dictionary
    • parent (Wrapper) – the parent of this wrapper
    • gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id', 'deleted', 'latest_workflow_uuid', 'name', 'number_of_steps', 'owner', 'published', 'show_in_tool_panel', 'tags')

    deleted: bool
    name: str
    owner: str
    published: bool
    show_in_tool_panel: bool
    tags: List[str]

class bioblend.galaxy.objects.wrappers_WRAPPER(*args: Any, **kwargs: Any)
    Abstract base class for Galaxy entity wrappers.
    Wrapper instances wrap deserialized JSON dictionaries such as the ones obtained by the Galaxy web API, converting key-based access to attribute-based access (e.g., library['name'] -> library.name).

    Dict keys that are converted to attributes are listed in the BASE_ATTRS class variable: this is the ‘stable’ interface. Note that the wrapped dictionary is accessible via the wrapped attribute.

    Parameters
    • wrapped (dict) – JSON-serializable dictionary
    • parent (Wrapper) – the parent of this wrapper
    • gi (GalaxyInstance) – the GalaxyInstance through which we can access this wrapper

BASE_ATTRS: Tuple[str, ...] = ('id',)

    clone() -> WrapperSubtype
    Return an independent copy of this wrapper.

    classmethod from_json(jdef: str) -> WrapperSubtype
    Build a new wrapper from a JSON dump.

    gi: GalaxyInstance | None

    id: str
property is_mapped: bool
    True if this wrapper is mapped to an actual Galaxy entity.

is_modified: bool

property parent: Wrapper | None
    The parent of this wrapper.

to_json() → str
    Return a JSON dump of this wrapper.

touch() → None
    Mark this wrapper as having been modified since its creation.

unmap() → None
    Disconnect this wrapper from Galaxy.

wrapped: dict

5.1.3 Usage documentation

This page describes some sample use cases for the Galaxy API and provides examples for these API calls. In addition to this page, there are functional examples of complete scripts in the docs/examples directory of the BioBlend source code repository.

Connect to a Galaxy server

To connect to a running Galaxy server, you will need an account on that Galaxy instance and an API key for the account. Instructions on getting an API key can be found at https://galaxyproject.org/develop/api/.

To open a connection call:

```python
from bioblend.galaxy import GalaxyInstance

gi = GalaxyInstance(url='http://example.galaxy.url', key='your-API-key')
```

We now have a GalaxyInstance object which allows us to interact with the Galaxy server under our account, and access our data. If the account is a Galaxy admin account we also will be able to use this connection to carry out admin actions.

View Histories and Datasets

Methods for accessing histories and datasets are grouped under GalaxyInstance.histories.* and GalaxyInstance.datasets.* respectively.

To get information on the Histories currently in your account, call:

```python
>>> gi.histories.get_histories()
[{'id': 'f3c2b0f3ecac9f02',
  'name': 'RNAseq_DGE_BASIC_Prep',
  'url': '/api/histories/f3c2b0f3ecac9f02'},
 {'id': '8a91dcf1866a80c2',
  'name': 'June demo',
  'url': '/api/histories/8a91dcf1866a80c2'}]
```
This returns a list of dictionaries containing basic metadata, including the id and name of each History. In this case, we have two existing Histories in our account, ‘RNAseq_DGE_BASIC_Prep’ and ‘June demo’. To get more detailed information about a History we can pass its id to the `show_history` method:

```python
>>> gi.histories.show_history('f3c2b0f3ecac9f02', contents=False)
{'annotation': '',
 'contents_url': '/api/histories/f3c2b0f3ecac9f02/contents',
 'id': 'f3c2b0f3ecac9f02',
 'name': 'RNAseq_DGE_BASIC_Prep',
 'nice_size': '93.5 MB',
 'state': 'ok',
 'state_details': {'discarded': 0,
 'empty': 0,
 'error': 0,
 'failed_metadata': 0,
 'new': 0,
 'ok': 7,
 'paused': 0,
 'queued': 0,
 'running': 0,
 'setting_metadata': 0,
 'upload': 0},
 'state_ids': {'discarded': [],
 'empty': [],
 'error': [],
 'failed_metadata': [],
 'new': [],
 'ok': ['d6842fb08a76e351',
 '10a4b652da44e82a',
 '81c601a2549966a0',
 'a154f05e3bcee26b',
 '1352fe19ddce0400',
 '06d549c52d753e53',
 '9ec54455d6279cc7'],
 'paused': [],
 'queued': [],
 'running': [],
 'setting_metadata': [],
 'upload': []}}
```

This gives us a dictionary containing the History’s metadata. With `contents=False` (the default), we only get a list of ids of the datasets contained within the History; with `contents=True` we would get metadata on each dataset. We can also directly access more detailed information on a particular dataset by passing its id to the `show_dataset` method:

```python
>>> gi.datasets.show_dataset('10a4b652da44e82a')
{'data_type': 'fastqsanger',
 'deleted': False,
 'file_size': 16527060,
 'genome_build': 'dm3',
 'id': 17499,
 'metadata_data_lines': None,
 'metadata_dbkey': 'dm3',
 'metadata_sequences': None,
 'misc_blurb': '15.8 MB'}
```

(continues on next page)
Uploading Datasets to a History

To upload a local file to a Galaxy server, you can run the `upload_file` method, supplying the path to a local file:

```python
>>> gi.tools.upload_file('test.txt', 'f3c2b0f3ecac9f02')
{'implicit_collections': [],
 'jobs': [{'create_time': '2015-07-28T17:52:39.756488',
 'id': '9752b387803d3e1e',
 'model_class': 'Job',
 'state': 'new',
 'tool_id': 'upload1',
 'update_time': '2015-07-28T17:52:39.987509'}],
 'output_collections': [],
 'outputs': [{'create_time': '2015-07-28T17:52:39.331176',
 'data_type': 'galaxy.datatypes.data.Text',
 'deleted': False,
 'file_ext': 'auto',
 'file_size': 0,
 'genome_build': '?',
 'hda_ldda': 'hda',
 'hid': 16,
 'history_content_type': 'dataset',
 'history_id': 'f3c2b0f3ecac9f02',
 'id': '59c76a119581e190',
 'metadata_data_lines': None,
 'metadata_dbkey': '?',
 'misc_blurb': None,
 'misc_info': None,
 'model_class': 'HistoryDatasetAssociation',
 'name': 'test.txt',
 'output_name': 'output0',
 'peek': '<table cellspacing="0" cellpadding="3">
</table>',
 'purged': False,
 'state': 'queued',
 'tags': [],
 'uuid': 'ff0ee99b-7542-4125-802d-7a193f388e7e',
 'visible': True}
}`

If files are greater than 2GB in size, they will need to be uploaded via FTP. Importing files from the user’s FTP folder can be done via running the upload tool again:
BioBlend Documentation, Release 1.2.0

>>> gi.tools.upload_from_ftp('test.txt', 'f3c2b0f3ecac9f02')
{'implicit_collections': [],
 'jobs': [{'create_time': '2015-07-28T17:57:43.704394',
           'exit_code': None,
           'id': '82b264d8c3d11790',
           'model_class': 'Job',
           'state': 'new',
           'tool_id': 'upload1',
           'update_time': '2015-07-28T17:57:43.910958'}],
 'output_collections': [],
 'outputs': [{'create_time': '2015-07-28T17:57:43.209041',
              'data_type': 'galaxy.datatypes.data.Text',
              'deleted': False,
              'file_ext': 'auto',
              'file_size': 0,
              'genome_build': '?',
              'hda_ldda': 'hda',
              'hid': 17,
              'history_content_type': 'dataset',
              'history_id': 'f3c2b0f3ecac9f02',
              'id': 'a676e8f07209a3be',
              'metadata_data_lines': None,
              'metadata_dbkey': '?',
              'misc_blurb': None,
              'misc_info': None,
              'model_class': 'HistoryDatasetAssociation',
              'name': 'test.txt',
              'output_name': 'output0',
              'peek': '<table cellspacing="0" cellpadding="3"></table>',
              'purged': False,
              'state': 'queued',
              'tags': [],
              'update_time': '2015-07-28T17:57:43.544407',
              'uuid': '2cbe8f0a-4019-47c4-87e2-005ce35b8449',
              'visible': True}]}

View Data Libraries

Methods for accessing Data Libraries are grouped under GalaxyInstance.libraries.*. Most Data Library methods are available to all users, but as only administrators can create new Data Libraries within Galaxy, the create_folder and create_library methods can only be called using an API key belonging to an admin account.

We can view the Data Libraries available to our account using:

>>> gi.libraries.get_libraries()
[{'id': '866f930d00d123ea',
  'name': 'RNA-seq workshop data',
  'url': '/api/libraries/866f930d00d123ea'},
 {'id': 'f740ab636b360a70',
  'name': '1000 genomes',
  'url': '/api/libraries/f740ab636b360a70'}]

This gives a list of metadata dictionaries with basic information on each library. We can get more information on a
particular Data Library by passing its id to the `show_library` method:

```python
>>> gi.libraries.show_library('8e6f930d00d123ea')
{'contents_url': '/api/libraries/8e6f930d00d123ea/contents',
 'description': 'RNA-Seq workshop data',
 'name': 'RNA-Seq',
 'synopsis': 'Data for the RNA-Seq tutorial'}
```

### Upload files to a Data Library

We can get files into Data Libraries in several ways: by uploading from our local machine, by retrieving from a URL, by passing the new file content directly into the method, or by importing a file from the filesystem on the Galaxy server. For instance, to upload a file from our machine we might call:

```python
>>> gi.libraries.upload_file_from_local_path('8e6f930d00d123ea', '/local/path/to/mydata.fastq', file_type='fastqsanger')
```

Note that we have provided the id of the destination Data Library, and in this case we have specified the type that Galaxy should assign to the new dataset. The default value for `file_type` is ‘auto’, in which case Galaxy will attempt to guess the dataset type.

### View Workflows

Methods for accessing workflows are grouped under `GalaxyInstance.workflows.*`. To get information on the Workflows currently in your account, use:

```python
>>> gi.workflows.get_workflows()
[{'id': 'e8b85ad72aefca86',
 'name': 'TopHat + cufflinks part 1',
 'url': '/api/workflows/e8b85ad72aefca86'},
 {'id': 'b0631c44aa74526d',
 'name': 'CuffDiff',
 'url': '/api/workflows/b0631c44aa74526d'}]
```

This returns a list of metadata dictionaries. We can get the details of a particular Workflow, including its steps, by passing its id to the `show_workflow` method:

```python
>>> gi.workflows.show_workflow('e8b85ad72aefca86')
{'id': 'e8b85ad72aefca86',
 'inputs': {'252': {'label': 'Input RNA-seq fastq', 'value': ''},
 'name': 'TopHat + cufflinks part 1',
 'steps': {'250': {'id': 250,
 'input_steps': {'input1': {'source_step': 252,
 'step_output': 'output'}},
 'tool_id': 'tophat',
 'type': 'tool'}},
 '251': {'id': 251,
 'input_steps': {'input': {'source_step': 250,
 'step_output': 'accepted_hits'}},
 'tool_id': 'cufflinks',
 'type': 'tool'}},
(continues on next page)
Export or import a workflow

Workflows can be exported from or imported into Galaxy. This makes it possible to archive workflows, or to move them between Galaxy instances.

To export a workflow, we can call:

```python
>>> workflow_dict = gi.workflows.export_workflow_dict('e8b85ad72aefca86')
```

This gives us a complex dictionary representing the workflow. We can import this dictionary as a new workflow with:

```python
>>> gi.workflows.import_workflow_dict(workflow_dict)
```

This call returns a dictionary containing basic metadata on the new workflow. Since in this case we have imported the dictionary into the original Galaxy instance, we now have a duplicate of the original workflow in our account:

```python
>>> gi.workflows.get_workflows()
[{
    'id': 'c0bacafdfe211f9a',
    'name': 'TopHat + cufflinks part 1 (imported from API)',
    'url': '/api/workflows/c0bacafdfe211f9a'},
{
    'id': 'e8b85ad72aefca86',
    'name': 'TopHat + cufflinks part 1',
    'url': '/api/workflows/e8b85ad72aefca86'},
{
    'id': 'b0631c44aa74526d',
    'name': 'CuffDiff',
    'url': '/api/workflows/b0631c44aa74526d'}]
```

Instead of using dictionaries directly, workflows can be exported to or imported from files on the local disk using the `export_workflow_to_local_path` and `import_workflow_from_local_path` methods. See the API reference for details.

**Note:** If we export a workflow from one Galaxy instance and import it into another, Galaxy will only run it without modification if it has the same versions of the tool wrappers installed. This is to ensure reproducibility. Otherwise, we will need to manually update the workflow to use the new tool versions.
Invoke a workflow

To invoke a workflow, we need to tell Galaxy which datasets to use for which workflow inputs. We can use datasets from histories or data libraries.

Examine the workflow above. We can see that it takes only one input file. That is:

```python
>>> wf = gi.workflows.show_workflow('e8b85ad72aeefca86')
>>> wf['inputs']['252']
{'label': 'Input RNA-seq fastq', 'value': ''}
```

There is one input, labelled 'Input RNA-seq fastq'. This input is passed to the Tophat tool and should be a fastq file. We will use the dataset we examined above, under View Histories and Datasets, which had name 'C1_R2_1 chr4 fq' and id '10a4b652da44e82a'.

To specify the inputs, we build a data map and pass this to the `invoke_workflow` method. This data map is a nested dictionary object which maps inputs to datasets. We call:

```python
>>> datamap = {'252': {'src': 'hda', 'id': '10a4b652da44e82a'}}
>>> gi.workflows.invoke_workflow('e8b85ad72aeefca86', inputs=datamap, history_name='New output history')
```

In this case the only input id is '252' and the corresponding dataset id is '10a4b652da44e82a'. We have specified the dataset source to be 'hda' (HistoryDatasetAssociation) since the dataset is stored in a History. See the API reference for allowed dataset specifications. We have also requested that a new History be created and used to store the results of the run, by setting `history_name='New output history'`.

The `invoke_workflow` call submits all the jobs which need to be run to the Galaxy workflow engine, with the appropriate dependencies so that they will run in order. The call returns immediately, so we can continue to submit new jobs while waiting for this workflow to execute. `invoke_workflow` returns the a dictionary describing the workflow invocation.

If we view the output History immediately after calling `invoke_workflow`, we will see something like:

```python
>>> gi.histories.show_history('0a7b7992a7cabaec')

{'annotation': '',
 'contents_url': '/api/histories/0a7b7992a7cabaec/contents',
 'id': '0a7b7992a7cabaec',
 'name': 'New output history',
 'nice_size': '0 bytes',
 'state': 'queued',
 'state_details': {'discarded': 0,
                   'empty': 0,
                   'error': 0,
                   'failed_metadata': 0,
                   'new': 0,
                   'ok': 0,
```

(continues on next page)
BioBlend Documentation, Release 1.2.0

[Continued from previous page]

In this case, because the submitted jobs have not had time to run, the output History contains 8 datasets in the 'queued' state and has a total size of 0 bytes. If we make this call again later we should instead see completed output files.

View Users

Methods for managing users are grouped under GalaxyInstance.users.*. User management is only available to Galaxy administrators, that is, the API key used to connect to Galaxy must be that of an admin account.

To get a list of users, call:

```python
>>> gi.users.get_users()
[{'email': 'userA@example.org',
  'id': '975a9ce09b49502a',
  'quota_percent': None,
  'url': '/api/users/975a9ce09b49502a'},
{'email': 'userB@example.org',
  'id': '0193a95acf427d2c',
  'quota_percent': None,
  'url': '/api/users/0193a95acf427d2c'}]
```
Using BioBlend for raw API calls

BioBlend can be used to make HTTP requests to the Galaxy API in a more convenient way than using e.g. the `requests` Python library. There are 5 available methods corresponding to the most common HTTP methods: `make_get_request`, `make_post_request`, `make_put_request`, `make_delete_request` and `make_patch_request`. One advantage of using these methods is that the API keys stored in the `GalaxyInstance` object is automatically added to the request.

To make a GET request to the Galaxy API with BioBlend, call:

```python
>>> gi.make_get_request(gi.base_url + '/api/version').json(){
    'version_major': '19.05',
    'extra': {}
}
```

To make a POST request to the Galaxy API with BioBlend, call:

```python
>>> gi.make_post_request(gi.base_url + '/api/histories', payload={'name': 'test history'})
...
{
    'importable': False,
    'create_time': '2019-07-05T20:04:823716',
    'contents_url': '/api/histories/a77b3f95070d689a/contents',
    'id': 'a77b3f95070d689a',
    'size': 0, 'user_id': '5b732999121d4593',
    'username_and_slug': None,
    'annotation': None,
    'state_details': {
        'discarded': 0,
        'ok': 0,
        'failed_metadata': 0,
        'upload': 0,
        'paused': 0,
        'running': 0,
        'setting_metadata': 0,
        'error': 0,
        'new': 0,
        'queued': 0,
        'empty': 0,
        'state': 'new',
        'empty': True,
        'update_time': '2019-07-05T20:04:823742',
        'tags': [],
        'deleted': False,
        'genome_build': None,
        'slug': None,
        'name': 'test history',
        'url': '/api/histories/a77b3f95070d689a',
        'state_ids': {
            'discarded': [],
            'ok': [],
            'failed_metadata': [],
            'upload': [],
            'paused': [],
            'running': [],
            'setting_metadata': [],
            'error': [],
            'new': []
        }
    }
}
```
5.2 Toolshed API

API used to interact with the Galaxy Toolshed, including repository management.

5.2.1 API documentation for interacting with the Galaxy Toolshed

**ToolShedInstance**

class bioblend.toolshed.ToolShedInstance(url: str, key: str | None = None, email: str | None = None, password: str | None = None, *, verify: bool = True)

A base representation of a connection to a ToolShed instance, identified by the ToolShed URL and user credentials.

After you have created a ToolShedInstance object, access various modules via the class fields. For example, to work with repositories and get a list of all public repositories, the following should be done:

```python
from bioblend import toolshed

ts = toolshed.ToolShedInstance(url='https://testtoolshed.g2.bx.psu.edu')

rl = ts.repositories.get_repositories()

tools = ts.tools.search_tools('fastq')
```

**Parameters**

- **url** (str) – A FQDN or IP for a given instance of ToolShed. For example: https://testtoolshed.g2.bx.psu.edu. If a ToolShed instance is served under a prefix (e.g. http://127.0.0.1:8080/toolshed/), supply the entire URL including the prefix (note that the prefix must end with a slash).

- **key** (str) – If required, user’s API key for the given instance of ToolShed, obtained from the user preferences.

- **email** (str) – ToolShed e-mail address corresponding to the user. Ignored if key is supplied directly.

- **password** (str) – Password of ToolShed account corresponding to the above e-mail address. Ignored if key is supplied directly.

- **verify** (bool) – Whether to verify the server’s TLS certificate

```
__init__(url: str, key: str | None = None, email: str | None = None, password: str | None = None, *, verify: bool = True) → None
```

A base representation of a connection to a ToolShed instance, identified by the ToolShed URL and user credentials.
After you have created a ToolShedInstance object, access various modules via the class fields. For example, to work with repositories and get a list of all public repositories, the following should be done:

```python
from bioblend import toolshed
ts = toolshed.ToolShedInstance(url='https://testtoolshed.g2.bx.psu.edu')
rl = ts.repositories.get_repositories()
tools = ts.tools.search_tools('fastq')
```

### Parameters

- **url (str)** – A FQDN or IP for a given instance of ToolShed. For example: https://testtoolshed.g2.bx.psu.edu. If a ToolShed instance is served under a prefix (e.g. http://127.0.0.1:8080/toolshed/), supply the entire URL including the prefix (note that the prefix must end with a slash).

- **key (str)** – If required, user’s API key for the given instance of ToolShed, obtained from the user preferences.

- **email (str)** – ToolShed e-mail address corresponding to the user. Ignored if key is supplied directly.

- **password (str)** – Password of ToolShed account corresponding to the above e-mail address. Ignored if key is supplied directly.

- **verify (bool)** – Whether to verify the server’s TLS certificate

### Categories

Interaction with a ToolShed instance categories

```python
class bioblend.toolshed.categories.ToolShedCategoryClient(toolshed_instance: ToolShedInstance)
    A generic Client interface defining the common fields.

    All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): `self.module = 'workflows' | 'libraries' | 'histories' | ...`

    `get_categories(deleted: bool = False) → List[Dict[str, Any]]`

    Returns a list of dictionaries that contain descriptions of the repository categories found on the given ToolShed instance.

    Parameters

    deleted (bool) – whether to show deleted categories. Requires administrator access to the ToolShed instance.

    Return type

    list

    Returns

    A list of dictionaries containing information about repository categories present in the ToolShed. For example:
```
New in version 0.5.2.

get_repositories(category_id: str, sort_key: Literal['name', 'owner'] = 'name', sort_order: Literal['asc', 'desc'] = 'asc') → Dict[str, Any]

Returns a dictionary of information for a repository category including a list of repositories belonging to the category.

Parameters

- **category_id** (str) – Encoded category ID
- **sort_key** (str) – key for sorting. Options are ‘name’ or ‘owner’ (default ‘name’).
- **sort_order** (str) – ordering of sorted output. Options are ‘asc’ or ‘desc’ (default ‘asc’).

Returns

da dict

A dict containing information about the category including a list of repository dicts. For example:

```json
{
    'deleted': False,
    'description': 'Tools for constructing and analyzing 3-dimensional shapes and their properties',
    'id': '589548af7e391bcf',
    'model_class': 'Category',
    'name': 'Constructive Solid Geometry',
    'repositories': [
        {
            'create_time': '2016-08-23T18:53:23.845013',
            'deleted': False,
            'deprecated': False,
            'description': 'Adds a surface field to a selected shape based on a given mathematical expression',
            'homepage_url': 'https://github.com/gregvonkuster/galaxy-csg',
            'id': 'af2ccc53697b064c',
            'metadata': {
                'changeset_revision': 'e12b55e960de',
                'downloadable': True,
                'has_repository_dependencies': False,
                'id': 'dfe022067783215f',
                'includes_datatypes': True
            }...
        }
    ]...
}
```
module: str = 'categories'

show_category(category_id: str) → Dict[str, Any]

Get details of a given category.

Parameters

category_id (str) – Encoded category ID

Return type
dict

Returns
details of the given category
Repositories

Interaction with a Tool Shed instance repositories

```
class bioblend.toolshedrepositories.ToolShedRepositoryClient(toolshed_instance: ToolShedInstance)

A generic Client interface defining the common fields.

All clients must define the following field (which will be used as part of the URL composition (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...

create_repository(name: str, synopsis: str, description: str | None = None, type: Literal['unrestricted', 'repository_suite_definition', 'tool_dependency_definition'] = 'unrestricted', remote_repository_url: str | None = None, homepage_url: str | None = None, category_ids: List[str] | None = None) → Dict[str, Any]

Create a new repository in a Tool Shed.

Parameters

- **name (str)** – Name of the repository
- **synopsis (str)** – Synopsis of the repository
- **description (str)** – Optional description of the repository
- **type (str)** – type of the repository. One of “unrestricted”, “repository_suite_definition”, or “tool_dependency_definition”
- **remote_repository_url (str)** – Remote URL (e.g. GitHub/Bitbucket repository)
- **homepage_url (str)** – Upstream’s homepage for the project
- **category_ids (list)** – List of encoded category IDs

Return type
dict

Returns

a dictionary containing information about the new repository. For example:

```json
{
    "deleted": false,
    "deprecated": false,
    "description": "new_synopsis",
    "homepage_url": "https://github.com/galaxyproject/",
    "id": "8cf91205f2f737f4",
    "long_description": "this is some repository",
    "model_class": "Repository",
    "name": "new_repo_17",
    "owner": "qqqqqq",
    "private": false,
    "remote_repository_url": "https://github.com/galaxyproject/tools-devteam",
    "times_downloaded": 0,
    "type": "unrestricted",
    "user_id": "adb5f5c93f827949"
}
```
get_ordered_installable_revisions\((name: str, owner: str) \rightarrow List[str]\)

Returns the ordered list of changeset revision hash strings that are associated with installable revisions. As in the changelog, the list is ordered oldest to newest.

**Parameters**

- **name** (str) – the name of the repository
- **owner** (str) – the owner of the repository

**Return type**

list

**Returns**

List of changeset revision hash strings from oldest to newest

get_repositories\((name: str | None = None, owner: str | None = None) \rightarrow List[Dict[str, Any]]\)

Get all repositories in a Galaxy Tool Shed, or select a subset by specifying optional arguments for filtering (e.g. a repository name).

**Parameters**

- **name** (str) – Repository name to filter on.
- **owner** (str) – Repository owner to filter on.

**Return type**

list

**Returns**

Returns a list of dictionaries containing information about repositories present in the Tool Shed. For example:

```python

[[
    'category_ids': ['c1df3132f6334b0e', 'f6d7b0037d901d9b'],
    'create_time': '2020-02-09T16:24:37.098176',
    'deleted': False,
    'deprecated': False,
    'description': 'Order Contigs',
    'homepage_url': '',
    'id': '287bd69f724b99ce',
    'model_class': 'Repository',
    'name': 'best_tool_ever',
    'owner': 'billybob',
    'private': False,
    'remote_repository_url': '',
    'times_downloaded': 0,
    'type': 'unrestricted',
    'user_id': '5cefd48bc04af6d4']]
```

changed in version 0.4.1: Changed method name from get_tools to get_repositories to better align with the Tool Shed concepts.

get_repository_revision_install_info\((name: str, owner: str, changeset_revision: str) \rightarrow List[Dict[str, Any]]\)

Return a list of dictionaries of metadata about a certain changeset revision for a single tool.

**Parameters**

- **name** (str) – the name of the repository
• **owner** *(str)* – the owner of the repository

• **changeset_revision** *(str)* – the changeset_revision of the RepositoryMetadata object associated with the repository

**Return type**
List of dictionaries

**Returns**
Returns a list of the following dictionaries:

1. a dictionary defining the repository
2. a dictionary defining the repository revision (RepositoryMetadata)
3. a dictionary including the additional information required to install the repository

For example:

```json
[{
  'create_time': '2020-08-20T13:17:08.818518',
  'deleted': False,
  'deprecated': False,
  'description': 'Galaxy Freebayes Bayesian genetic variant detector tool',
  'homepage_url': '',
  'id': '491b7a3fddf9366f',
  'long_description': 'Galaxy Freebayes Bayesian genetic variant detector tool originally included in the Galaxy code distribution but migrated to the tool shed.',
  'model_class': 'Repository',
  'name': 'freebayes',
  'owner': 'devteam',
  'private': False,
  'remote_repository_url': '',
  'times_downloaded': 269,
  'type': 'unrestricted',
  'url': '/api/repositories/491b7a3fddf9366f',
  'user_id': '1de29d50c3c44272'},
{
  'changeset_revision': 'd291dc763c4c',
  'do_not_test': False,
  'downloadable': True,
  'has_repository_dependencies': False,
  'id': '504be8aaa652c154',
  'includes_datatypes': False,
  'includes_tool_dependencies': True,
  'includes_tools': True,
  'includes_tools_for_display_in_tool_panel': True,
  'includes_workflows': False,
  'malicious': False,
  'missing_test_components': False,
  'model_class': 'RepositoryMetadata',
  'numeric_revision': 0,
  'repository_id': '491b7a3fddf9366f',
  'url': '/api/repository_revisions/504be8aaa652c154',
  'valid_tools': [{
    'add_to_tool_panel': True,
    'description': '- Bayesian genetic variant detector'}
},
```

(continues on next page)
{'guid': 'testtoolshed.g2.bx.psu.edu/repos/devteam/freebayes/
freebayes/0.0.3',
'id': 'freebayes',
'name': 'FreeBayes',
'requirements': [{
'name': 'freebayes',
'type': 'package',
'version': '0.9.6_9608597d12e127c847ae03aa03440ab63992fedf'},
{name: 'samtools', type: 'package', version: '0.1.18'}],
'tests': [{
'inputs': [{
'reference_source|reference_source_selector':
',
'options_type|options_type_selector': 'basic',
'reference_source|ref_file': 'phiX.fasta',
'reference_source|input_bams_0|input_bam': 'fake_phiX_reads_1.bam'},
'name': 'Test-1',
'outputs': [{
'output_vcf': 'freebayes_out_1.vcf.contains'},
'required_files': ['fake_phiX_reads_1.bam',
'phiX.fasta',
'freebayes_out_1.vcf.contains']},
'tool_config': '/srv/toolshed/test/var/data/repos/000/repo_708/
freebayes.xml',
'tool_type': 'default',
'version': '0.0.3',
'version_string_cmd': None}],
'freebayes': ['Galaxy Freebayes Bayesian genetic variant detector',
tool',
'http://testtoolshed.g2.bx.psu.edu/repos/devteam/
freebayes',
d291dc763c4c',
'9',
'devteam',
{}],
'freebayes/0.9.6_9608597d12e127c847ae03aa03440ab63992fedf': {'changeset_revision':
d291dc763c4c',
{name: 'freebayes',
'repository_name': 'freebayes',
'repository_owner': 'devteam',
'type': 'package',
'version': '0.9.6_9608597d12e127c847ae03aa03440ab63992fedf'}},
samtools/0.1.18': {'changeset_revision':
d291dc763c4c',
'name': 'samtools',
'repository_name': 'freebayes',
'repository_owner': 'devteam',
'}
module: str = 'repositories'

repository_revisions(downloadable: bool | None = None,
    malicious: bool | None = None,
    missing_test_components: bool | None = None,
    includes_tools: bool | None = None) -> List[Dict[str, Any]]

Returns a (possibly filtered) list of dictionaries that include information about all repository revisions. The following parameters can be used to filter the list.

Parameters

- **downloadable** (bool) – Can the tool be downloaded
- **malicious** (bool) –
- **missing_test_components** (bool) –
- **includes_tools** (bool) –

Return type

List of dictionaries

Returns

Returns a (possibly filtered) list of dictionaries that include information about all repository revisions. For example:

```
[[
    'changeset_revision': '6e26c5a48e9a',
    'downloadable': True,
    'has_repository_dependencies': False,
    'id': '92250aff777a169',
    'includes_datatypes': False,
    'includes_tool_dependencies': False,
    'includes_tools': True,
    'includes_tools_for_display_in_tool_panel': True,
    'includes_workflows': False,
    'malicious': False,
    'missing_test_components': False,
    'model_class': 'RepositoryMetadata',
    'numeric_revision': None,
    'repository_id': '78f2604ff5e65707',
    'url': '/api/repository_revisions/92250aff777a169'],
[
    'changeset_revision': '15a54fa11ad7',
    'downloadable': True,
    'has_repository_dependencies': False,
    'id': 'd3823c748ae2205d',
    'includes_datatypes': False,
    'includes_tool_dependencies': False,
    'includes_tools': True,
    'includes_tools_for_display_in_tool_panel': True,
    'includes_workflows': False,
    'malicious': False,
    'missing_test_components': False,
    'model_class': 'RepositoryMetadata',
```
search_repositories(q: str, page: int = 1, page_size: int = 10) → Dict[str, Any]

Search for repositories in a Galaxy Tool Shed.

Parameters

- **q** (str) – query string for searching purposes
- **page** (int) – page requested
- **page_size** (int) – page size requested

Return type
dict

Returns
dictionary containing search hits as well as metadata for the search. For example:

```python
{
    'hits': [ {
        'matched_terms': [],
        'repository': { 'approved': 'no',
            'categories': 'fastq manipulation',
            'description': 'Convert export file to fastq',
            'full_last_updated': '2015-01-18 09:48 AM',
            'homepage_url': '',
            'id': 'bdfa208f0cf6504e',
            'last_updated': 'less than a year',
            'long_description': 'This is a simple tool to convert Solexa Export files to FASTQ files.',
            'name': 'export_to_fastq',
            'remote_repository_url': '',
            'repo_lineage': '[0:c9e926d9d87e, 1:38859774da87]
            repo_owner_username': 'louise',
            'times_downloaded': 164},
        'score': 4.92},
    ]
}
```

(continues on next page)
show_repository(toolShed_id: str) → Dict[ str, Any]

Display information of a repository from Tool Shed

Parameters
  toolShed_id (str) – Encoded Tool Shed ID

Return type
  dict

Returns

Information about the tool. For example:

```python
{
    'category_ids': ['c1df3132f6334b0e', 'f6d7b0037d901d9b'],
    'create_time': '2020-02-22T20:39:15.548491',
    'deleted': False,
    'deprecated': False,
    'description': 'Order Contigs',
    'homepage_url': '',
    'id': '287bd69f724b99ce',
    'long_description': '',
    'model_class': 'Repository',
    'name': 'best_tool_ever',
    'owner': 'billybob',
    'private': False,
    'remote_repository_url': '',
    'times_downloaded': 0,
    'type': 'unrestricted',
    'user_id': '5cefd48bc04af6d4'
}
```

Changed in version 0.4.1: Changed method name from `show_tool` to `show_repository` to better align with the Tool Shed concepts.

show_repository_revision(metadata_id: str) → Dict[str, Any]

Returns a dictionary that includes information about a specified repository revision.

Parameters
  metadata_id (str) – Encoded repository metadata ID

Return type
  dict

Returns

Returns a dictionary that includes information about a specified repository revision. For example:
update_repository(id: str, tar_ball_path: str, commit_message: str | None = None) \rightarrow Dict[str, Any]

Update the contents of a Tool Shed repository with specified tar ball.

**Parameters**

- **id** (str) – Encoded repository ID
- **tar_ball_path** (str) – Path to file containing tar ball to upload.
- **commit_message** (str) – Commit message used for the underlying Mercurial repository backing Tool Shed repository.

**Return type**

dict

**Returns**

Returns a dictionary that includes repository content warnings. Most valid uploads will result in no such warning and an exception will be raised generally if there are problems. For example a successful upload will look like:

```json
{
    'content_alert': '',
    'message': ''
}
```

New in version 0.5.2.

update_repository_metadata(tool_shed_id: str, name: str | None = None, synopsis: str | None = None, description: str | None = None, remote_repository_url: str | None = None, homepage_url: str | None = None, category_ids: List[str] | None = None) \rightarrow Dict[str, Any]

Update metadata of a Tool Shed repository.

**Parameters**

- **name** (str) – ID of the repository to update
- **name** – New name of the repository
- **synopsis** (str) – New synopsis of the repository
- **description** (str) – New description of the repository
- **remote_repository_url** (str) – New remote URL (e.g. GitHub/Bitbucket repository)
• **homepage_url** (str) – New upstream homepage for the project
• **category_ids** (list) – New list of encoded category IDs

**Return type**
dict

**Returns**
a dictionary containing information about the updated repository.

**Tools**

Interaction with a Tool Shed instance tools

```python
class bioblend.toolshed.tools.ToolShedToolClient(toolshed_instance: ToolShedInstance)
    
    A generic Client interface defining the common fields.
    
    All clients must define the following field (which will be used as part of the URL composition
    (e.g., http://<galaxy_instance>/api/libraries): self.module = 'workflows' | 'libraries' | 'histories' | ...
    
    gi: ToolShedInstance
    module: str = 'tools'
```

**search_tools** *(q: str, page: int = 1, page_size: int = 10) → Dict[str, Any]*)

Search for tools in a Galaxy Tool Shed.

**Parameters**

- **q** (str) – query string for searching purposes
- **page** (int) – page requested
- **page_size** (int) – page size requested

**Return type**
dict

**Returns**
dictionary containing search hits as well as metadata for the search. For example:

```python
{'hits': [{'matched_terms': [],
            'score': 3.0,
            'tool': {'description': 'convert between various FASTQ quality formats',
                     'id': '69819b84d55f521efda001e0926e7233',
                     'name': 'FASTQ Groomer',
                     'repo_name': None,
                     'repo_owner_username': 'devteam'},
            },
            {'matched_terms': [],
            'score': 3.0,
            'tool': {'description': 'converts a bam file to fastq files.',
                     'id': '521e282770fd94537daff87adad2551b',
                     'name': 'Defuse BamFastq',
                     'repo_name': None,
                     'repo_owner_username': 'jjohnson'}],
```

(continues on next page)
'hostname': 'https://testtoolshed.g2.bx.psu.edu/',
'page': '1',
'page_size': '2',
'total_results': '118'}
BioBlend allows library-wide configuration to be set in external files. These configuration files can be used to specify access keys, for example.

### 6.1 Configuration documents for BioBlend

#### 6.1.1 BioBlend

**exception** `bioblend.ConnectionError`

```python
message: str, body: bytes | str | None = None, status_code: int | None = None
```

An exception class that is raised when unexpected HTTP responses come back.

Should make it easier to debug when strange HTTP things happen such as a proxy server getting in the way of the request etc. `@see` body attribute to see the content of the http response

**class** `bioblend.NullHandler`

```python
level=0
```

Initializes the instance - basically setting the formatter to `None` and the filter list to empty.

**emit**

```python
record: LogRecord → None
```

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

**exception** `bioblend.TimeoutException`

**bioblend.get_version() → str**

Returns a string with the current version of the library (e.g., “0.2.0”)

**bioblend.init_logging() → None**

Initialize BioBlend’s logging from a configuration file.

**bioblend.set_file_logger**

```python
name: str, filepath: str, level: int | str = 20, format_string: str | None = None) → None
```

**bioblend.set_stream_logger**

```python
name: str, level: int | str = 10, format_string: str | None = None) → None
```
6.1.2 Config

```python
class bioblend.config.Config(path: str | None = None, fp: IO[str] | None = None, do_load: bool = True)
```

BioBlend allows library-wide configuration to be set in external files. These configuration files can be used to specify access keys, for example. By default we use two locations for the BioBlend configurations:

- System wide: `/etc/bioblend.cfg`
- Individual user: `~/.bioblend` (which works on both Windows and Unix)
CHAPTER

SEVEN

TESTING

If you would like to do more than just a mock test, you need to point BioBlend to an instance of Galaxy. Do so by exporting the following two variables:

```
$ export BIOBLEND_GALAXY_URL=http://127.0.0.1:8080
$ export BIOBLEND_GALAXY_API_KEY=<API key>
```

The unit tests, stored in the `tests` folder, can be run using `pytest`. From the project root:

```
$ pytest
```
GETTING HELP

If you have run into issues, found a bug, or can’t seem to find an answer to your question regarding the use and functionality of BioBlend, please use the Github Issues page to ask your question.
CHAPTER NINE

RELATED DOCUMENTATION

Links to other documentation and libraries relevant to this library:

- Galaxy API documentation
- Blend4j: Galaxy API wrapper for Java
- clj-blend: Galaxy API wrapper for Clojure
INDICES AND TABLES

• genindex
• modindex
• search
b
bioblend, 133
bioblend.config, 134
bioblend.galaxy.config, 11
bioblend.galaxy.dataset_collections, 15
bioblend.galaxy.datasets, 12
bioblend.galaxy.datatypes, 17
bioblend.galaxy.folders, 18
bioblend.galaxy.forms, 20
bioblend.galaxy.ftpfiles, 21
bioblend.galaxy.genomes, 21
bioblend.galaxy.groups, 23
bioblend.galaxy.histories, 26
bioblend.galaxy.invocations, 35
bioblend.galaxy.jobs, 41
bioblend.galaxy.libraries, 47
bioblend.galaxy.objects.client, 86
bioblend.galaxy.objects.wrappers, 93
bioblend.galaxy.quotas, 54
bioblend.galaxy.roles, 57
bioblend.galaxy.tool_data, 66
bioblend.galaxy.tool_dependencies, 68
bioblend.galaxy.tools, 58
bioblend.galaxy.toolshed, 69
bioblend.galaxy.users, 72
bioblend.galaxy.visual, 75
bioblend.galaxy.workflows, 76
bioblend.toolshed.categories, 120
bioblend.toolshed.repositories, 123
bioblend.toolshed.tools, 131
Symbols
__init__() (bioblend.galaxy.GalaxyInstance method), 10
__init__() (bioblend.toolshed.ToolShedInstance method), 119

A
add_group_role() (bioblend.galaxy.groups.GroupsClient method), 23
add_group_user() (bioblend.galaxy.groups.GroupsClient method), 23
annotation (bioblend.galaxy.objects.wrappers.History attribute), 97
annotation (bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation attribute), 99

API_MODULE (bioblend.galaxy.objects.wrappers.DatasetCollection attribute), 94
API_MODULE (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
API_MODULE (bioblend.galaxy.objects.wrappers.History attribute), 96
API_MODULE (bioblend.galaxy.objects.wrappers.Library attribute), 101

BASE_ATTRS (bioblend.galaxy.objects.wrappers.Dataset attribute), 93
BASE_ATTRS (bioblend.galaxy.objects.wrappers.DatasetCollection attribute), 94
BASE_ATTRS (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Folder attribute), 96
BASE_ATTRS (bioblend.galaxy.objects.wrappers.History attribute), 97
BASE_ATTRS (bioblend.galaxy.objects.wrappers.HistoryCollection attribute), 99
BASE_ATTRS (bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation attribute), 99
BASE_ATTRS (bioblend.galaxy.objects.wrappers.HistoryDatasetCollectionAssociation attribute), 100

BASE_ATTRS (bioblend.galaxy.objects.wrappers.HistoryPreview attribute), 101
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Job attribute), 101
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Library attribute), 101
BASE_ATTRS (bioblend.galaxy.objects.wrappers.LibraryDatasetDatasetAssociation attribute), 104
BASE_ATTRS (bioblend.galaxy.objects.wrappers.LibraryDatasetFile attribute), 104
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Step attribute), 104
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Tool attribute), 105
BASE_ATTRS (bioblend.galaxy.objects.wrappers.Workflow attribute), 105
BASE_ATTRS (bioblend.galaxy.objects.wrappers.WorkflowPreview attribute), 109
BASE_ATTRS (bioblend.galaxy.objects.wrappersWRAPPER attribute), 109

bioblend module, 133
bioblend.config module, 134
bioblend.galaxy.config module, 11
bioblend.galaxy.dataset_collections module, 15
bioblend.galaxy.datasets module, 12
bioblend.galaxy.datatypes module, 17
bioblend.galaxy.folders module, 18
bioblend.galaxy.forms module, 20
bioblend.galaxy.ftpfiles module, 21
bioblend.galaxy.genomes module, 21
bioblend.galaxy.groups module, 23
bioblend.galaxy.histories module, 26
bioblend.galaxy.invocations
module, 35
bioblend.galaxy.jobs
module, 41
bioblend.galaxy.libraries
module, 47
bioblend.galaxy.objects.client
module, 86
bioblend.galaxy.objects.wrappers
module, 93
bioblend.galaxy.quotas
module, 54
bioblend.galaxy.roles
module, 57
bioblend.galaxy.tool_data
module, 66
bioblend.galaxy.tool_dependencies
module, 68
bioblend.galaxy.tools
module, 58
bioblend.galaxy.toolshed
module, 69
bioblend.galaxy.users
module, 72
bioblend.galaxy.visual
module, 75
bioblend.galaxy.workflows
module, 76
bioblend.toolshed.categories
module, 120
bioblend.toolshed.repositories
module, 123
bioblend.toolshed.tools
module, 131
build() (bioblend.galaxy.tools.ToolClient method), 58

C
cancel_invocation()
(bioblend.galaxy.invocations.InvocationClient method), 35
cancel_invocation()
(bioblend.galaxy.workflows.WorkflowClient method), 76
cancel_job() (bioblend.galaxy.jobs.JobsClient method), 41
collection_type (bioblend.galaxy.objects.wrappers.DatasetCollection attribute), 94
CollectionDescription (class in bioblend.galaxy.dataset_collections), 15
CollectionElement (class in bioblend.galaxy.dataset_collections), 15
Config (class in bioblend.config), 134
ConfigClient (class in bioblend.galaxy.config), 11
ConnectionError, 133
collection (bioblend.galaxy.objects.wrappers.Dataset attribute), 93
collection (bioblend.galaxy.objects.wrappers.DatasetCollection attribute), 94
collection (bioblend.galaxy.objects.wrappers.Folder attribute), 96
CONTAINER_PREVIEW_TYPE
(bioblend.galaxy.objects.client.ObjDatasetContainerClient attribute), 88
CONTAINER_PREVIEW_TYPE
(bioblend.galaxy.objects.client.ObjHistoryClient attribute), 88
CONTAINER_PREVIEW_TYPE
(bioblend.galaxy.objects.client.ObjLibraryClient attribute), 90
CONTAINER_TYPE
(bioblend.galaxy.objects.client.ObjDatasetContainerClient attribute), 88
CONTAINER_TYPE
(bioblend.galaxy.objects.client.ObjHistoryClient attribute), 88
CONTAINER_TYPE
(bioblend.galaxy.objects.client.ObjLibraryClient attribute), 90
CONTENT_INFO_TYPE
(bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
CONTENT_INFO_TYPE
(bioblend.galaxy.objects.wrappers.History attribute), 97
CONTENT_INFO_TYPE
(bioblend.galaxy.objects.wrappers.Library attribute), 101
content_infos (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
copy_content() (bioblend.galaxy.histories.HistoryClient method), 26
copy_dataset() (bioblend.galaxy.histories.HistoryClient method), 26
copy_from_dataset() (bioblend.galaxy.libraries.LibraryClient method), 47
copy_from_dataset() (bioblend.galaxy.objects.wrappers.Library method), 101
copy_from_dataset() (bioblend.galaxy.objects.wrappers.Library method), 88
copy_from_dataset() (bioblend.galaxy.objects.wrappers.Library method), 90
copy_from_dataset() (bioblend.galaxy.objects.wrappers.Library method), 26
create() (bioblend.galaxy.objects.client.ObjHistoryClient method), 97
create() (bioblend.galaxy.objects.client.ObjLibraryClient method), 101
create_dataset_collection() (bioblend.galaxy.histories.HistoryClient method), 26
create_dataset_collection() (bioblend.galaxy.histories.HistoryClient method), 97
create_folder() (bioblend.galaxy.folders.FoldersClient method), 18
create_folder() (bioblend.galaxy.libraries.LibraryClient method), 134
create_folder() (bioblend.galaxy.objects.wrappers.Library method), 47
create_form() (bioblend.galaxy.forms.FormsClient method), 20
create_group() (bioblend.galaxy.groups.GroupsClient method), 23
create_history() (bioblend.galaxy.histories.HistoryClient method), 27
create_history_tag() (bioblend.galaxy.histories.HistoryClient method), 27
create_library() (bioblend.galaxy.libraries.LibraryClient method), 47
create_local_user() (bioblend.galaxy.users.UserClient method), 72
create_quota() (bioblend.galaxy.quotas.QuotaClient method), 54
create_remote_user() (bioblend.galaxy.users.UserClient method), 72
create_repository() (bioblend.toolshed.repositories.ToolShedRepositoryClient method), 123
create_role() (bioblend.galaxy.roles.RolesClient method), 57
create_user_apikey() (bioblend.galaxy.users.UserClient method), 72
delete() (bioblend.galaxy.objects.client.ObjLibraryClient method), 91
delete() (bioblend.galaxy.objects.client.ObjWorkflowClient method), 92
delete() (bioblend.galaxy.objects.wrappers.DatasetCollection method), 94
delete() (bioblend.galaxy.objects.wrappers.DatasetContainer method), 95
delete() (bioblend.galaxy.objects.wrappers.History method), 97
delete() (bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation method), 99
delete() (bioblend.galaxy.objects.wrappers.HistoryDatasetCollectionAssociation method), 100
delete() (bioblend.galaxy.objects.wrappers.Library method), 101
delete() (bioblend.galaxy.objects.wrappers.LibraryDataset method), 103
delete() (bioblend.galaxy.objects.wrappers.Workflow method), 105
delete_data_table() (bioblend.galaxy.tool_data.ToolDataClient method), 66
delete_dataset() (bioblend.galaxy.histories.HistoryClient method), 27
delete_dataset_collection() (bioblend.galaxy.histories.HistoryClient method), 28
delete_folder() (bioblend.galaxy.folders.FoldersClient method), 18
delete_group_role() (bioblend.galaxy.groups.GroupsClient method), 23
delete_group_user() (bioblend.galaxy.groups.GroupsClient method), 24
delete_history() (bioblend.galaxy.histories.HistoryClient method), 28
delete_library() (bioblend.galaxy.libraries.LibraryClient method), 48
delete_library_dataset() (bioblend.galaxy.libraries.LibraryClient method), 48
delete_quota() (bioblend.galaxy.quotas.QuotaClient method), 55
delete_unused_dependency_paths() (bioblend.galaxy.tool_dependencies.ToolDependenciesClient method), 68
delete_user() (bioblend.galaxy.users.UserClient method), 73
delete_workflow() (bioblend.galaxy.workflows.WorkflowClient method), 76
delete() (bioblend.galaxy.objects.wrappers.DatasetCollectionAttribute), 94

Index 147
deleted (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
deleted (bioblend.galaxy.objects.wrappers.HistoryDatasetContainer attribute), 100
deleted (bioblend.galaxy.objects.wrappers.Workflow attribute), 106
deleted (bioblend.galaxy.objects.wrappers.WorkflowPreview attribute), 109
description (bioblend.galaxy.objects.wrappers.Folder attribute), 96
description (bioblend.galaxy.objects.wrappers.Library attribute), 102
download() (bioblend.galaxy.objects.wrappers.Dataset method), 93
download() (bioblend.galaxy.objects.wrappers.History method), 97
download_dataset() (bioblend.galaxy.datasets.DatasetClient method), 12
download_dataset_collection() (bioblend.galaxy.dataset_collections.DatasetCollectionClient method), 16
download_history() (bioblend.galaxy.histories.HistoryClient method), 28
DS_TYPE (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 95
DS_TYPE (bioblend.galaxy.objects.wrappers.History attribute), 97
DS_TYPE (bioblend.galaxy.objects.wrappers.Library attribute), 101
DSC_TYPE (bioblend.galaxy.objects.wrappers.History attribute), 97

E
elements (bioblend.galaxy.objects.wrappers.HistoryDatasetCollectionAssociation attribute), 100
emit() (bioblend.NullHandler method), 133
export() (bioblend.galaxy.objects.wrappers.History method), 97
export() (bioblend.galaxy.objects.wrappers.Workflow method), 106
export_history() (bioblend.galaxy.histories.HistoryClient method), 28
export_workflow_dict() (bioblend.galaxy.workflows.WorkflowClient method), 76
export_workflow_to_local_path() (bioblend.galaxy.workflows.WorkflowClient method), 77
extract_workflow_from_history() (bioblend.galaxy.workflows.WorkflowClient method), 77

F
file_name (bioblend.galaxy.objects.wrappers.LibraryDataset attribute), 103
Folder (class in bioblend.galaxy.objects.wrappers), 96
folder_ids (bioblend.galaxy.objects.wrappers.Library property), 102
FoldersClient (class in bioblend.galaxy.folders), 18
FormsClient (class in bioblend.galaxy.forms), 20
from_json() (bioblend.galaxy.objects.wrappers.Wrapper class method), 109
FTPFilesClient (class in bioblend.galaxy.ftpfiles), 21

G
GalaxyInstance (class in bioblend.galaxy), 9
GalaxyInstance (class in bioblend.galaxy.objects.galaxy_instance), 86
gene_build (bioblend.galaxy.objects.wrappers.Dataset attribute), 93
GenomeClient (class in bioblend.galaxy.genomes), 21
get() (bioblend.galaxy.objects.client.ObjClient method), 86
get() (bioblend.galaxy.objects.client.ObjDatasetClient method), 87
get() (bioblend.galaxy.objects.client.ObjDatasetCollectionClient method), 87
get() (bioblend.galaxy.objects.client.ObjDatasetContainerClient method), 88
get() (bioblend.galaxy.objects.client.ObjInvocationClient method), 89
get() (bioblend.galaxy.objects.client.ObjJobClient method), 90
get() (bioblend.galaxy.objects.client.ObjToolClient method), 91
get() (bioblend.galaxy.objects.client.ObjWorkflowClient method), 92
get_categories() (bioblend.toolshed.categories.ToolShedCategoryClient method), 120
get_citations() (bioblend.galaxy.tools.ToolClient method), 60
get_common_problems() (bioblend.galaxy.jobs.JobsClient method), 41
get_config() (bioblend.galaxy.config.ConfigClient method), 11
get_contents() (bioblend.galaxy.objects.wrappers.Dataset method), 93
get_current_user() (bioblend.galaxy.users.UserClient method), 73
get_data_tables() (bioblend.galaxy.tool_data.ToolDataClient method), 67
get_dataset() (bioblend.galaxy.objects.wrappers.DatasetContainer method), 95
get_dataset_collection() (bioblend.galaxy.objects.wrappers.History method), 97
get_dataset_permissions()
(bioblend.galaxy.libraries.LibraryClient method), 48
get_datasets()
(bioblend.galaxy.datasets.DatasetClient method), 13
get_datasets()
(bioblend.galaxy.objects.wrappers.DatasetContainer method), 95
get_datatypes()
(bioblend.galaxy.datatypes.DatatypesClient method), 17
get_destination_params()
(bioblend.galaxy.jobs.JobsClient method), 41
get_extra_files()
(bioblend.galaxy.histories.HistoryClient method), 29
get_folder()
(bioblend.galaxy.objects.wrappers.Library method), 102
get_folders()
(bioblend.galaxy.libraries.LibraryClient method), 48
get_forms()
(bioblend.galaxy.forms.FormsClient method), 20
get_ftp_files()
(bioblend.galaxy.ftpfiles.FTPFilesClient method), 21
get_genomes()
(bioblend.galaxy.genomes.GenomeClient method), 21
get_group_roles()
(bioblend.galaxy.groups.GroupsClient method), 24
get_group_users()
(bioblend.galaxy.groups.GroupsClient method), 24
get_groups()
(bioblend.galaxy.groups.GroupsClient method), 24
get_histories()
(bioblend.galaxy.histories.HistoryClient method), 29
get_inputs()
(bioblend.galaxy.jobs.JobsClient method), 42
get_invocation_biocompute_object()
(bioblend.galaxy.invocations.InvocationClient method), 36
get_invocation_report()
(bioblend.galaxy.invocations.InvocationClient method), 36
get_invocation_report_pdf()
(bioblend.galaxy.invocations.InvocationClient method), 36
get_invocation_step_jobs_summary()
(bioblend.galaxy.invocations.InvocationClient method), 36
get_invocation_summary()
(bioblend.galaxy.invocations.InvocationClient method), 37
get_invocations()
(bioblend.galaxy.invocations.InvocationClient method), 37
get_invocations()
(bioblend.galaxy.workflows.WorkflowClient method), 77
get_jobs()
(bioblend.galaxy.jobs.JobsClient method), 42
get_library_permissions()
(bioblend.galaxy.libraries.LibraryClient method), 49
get_libraries()
(bioblend.galaxy.libraries.LibraryClient method), 48
get_metrics()
(bioblend.galaxy.jobs.JobsClient method), 43
get_most_recently_used_history()
(bioblend.galaxy.histories.HistoryClient method), 30
get_or_create_user_apikey()
(bioblend.galaxy.users.UserClient method), 73
get_ordered_installable_revisions()
(bioblend.toolshed.repositories.ToolShedRepositoryClient method), 123
get_outputs()
(bioblend.galaxy.jobs.JobsClient method), 123
get_permissions()
(bioblend.galaxy.folders.FoldersClient method), 19
get_previews()
(bioblend.galaxy.objects.client.ObjClient method), 86
get_previews()
(bioblend.galaxy.objects.client.ObjDatasetClient method), 87
get_previews()
(bioblend.galaxy.objects.client.ObjDatasetCollectionClient method), 88
get_previews()
(bioblend.galaxy.objects.client.ObjDatasetContainerClient method), 88
get_previews()
(bioblend.galaxy.objects.client.ObjInvocationClient method), 89
get_previews()
(bioblend.galaxy.objects.client.ObjJobClient method), 90
get_previews()
(bioblend.galaxy.objects.client.ObjDatasetClient method), 91
get_previews()
(bioblend.galaxy.objects.client.ObjToolClient method), 91
get_published_histories()
(bioblend.galaxy.histories.HistoryClient method), 30
get_quotas()
(bioblend.galaxy.quotas.QuotaClient method), 55
get_repositories()
(bioblend.galaxy.toolshed.ToolShedClient method), 69
get_repositories()
(bioblend.galaxy.toolshed.categories.ToolShedCategoryClient method), 121
get_repositories()
(bioblend.galaxy.toolshed.repositories.ToolShedRepositoryClient method), 124
get_repository_revision_install_info()
(bioblend.galaxy.toolshed.repositories.ToolShedRepositoryClient method), 124
get_roles()
(bioblend.galaxy.roles.RolesClient method), 57
get_sniffers()
(bioblend.galaxy.datatypes.DatatypesClient method), 17
get_state() (bioblend.galaxy.jobs.JobsClient method), 43
get_status() (bioblend.galaxy.histories.HistoryClient method), 30
get_stream() (bioblend.galaxy.objects.wrappers.Dataset method), 93
get_stream() (bioblend.galaxy.objects.wrappers.History method), 100
get_tool_panel() (bioblend.galaxy.tools.ToolClient method), 60
get_tools() (bioblend.galaxy.tools.ToolClient method), 61
get_user_apikey() (bioblend.galaxy.users.UserClient method), 73
get_users() (bioblend.galaxy.users.UserClient method), 73
get_version() (in module bioblend), 133
get_visualizations() (bioblend.galaxy.visual.VisualClient method), 75
get_workflow_inputs() (bioblend.galaxy.workflows.WorkflowClient method), 78
get_workflows() (bioblend.galaxy.workflows.WorkflowClient method), 78
gi (bioblend.galaxy.dataset_collections.DatasetCollectionClient attribute), 16
gi (bioblend.galaxy.datasets.DatasetClient attribute), 14
gi (bioblend.galaxy.histories.HistoryClient attribute), 31
gi (bioblend.galaxy.invocations.InvocationClient attribute), 38
gi (bioblend.galaxy.objects.wrappers.Dataset attribute), 94
gi (bioblend.galaxy.objects.wrappers.DatasetCollection attribute), 95
gi (bioblend.galaxy.objects.wrappers.DatasetContainer attribute), 96
gi (bioblend.galaxy.objects.wrappers.Folder attribute), 96
gi (bioblend.galaxy.objects.wrappers.Tool attribute), 105
gi (bioblend.galaxy.objects.wrappers.Wrapper attribute), 109
gi (bioblend.galaxy.tools.ToolClient attribute), 61
gi (bioblend.toolshed.tools.ToolShedToolClient attribute), 131
GroupsClient (class in bioblend.galaxy.groups), 23

H
History (class in bioblend.galaxy.objects.wrappers), 99
HistoryClient (class in bioblend.galaxy.histories), 96
HistoryContentInfo (class in bioblend.galaxy.objects.wrappers), 99
HistoryDatasetAssociation (class in bioblend.galaxy.objects.wrappers), 99
HistoryDatasetCollectionAssociation (class in bioblend.galaxy.objects.wrappers), 100
HistoryDatasetCollectionElement (class in bioblend.galaxy.dataset_collections), 17
HistoryDatasetElement (class in bioblend.galaxy.dataset_collections), 17
HistoryPreview (class in bioblend.galaxy.objects.wrappers), 100
id (bioblend.galaxy.objects.wrappers.Wrapper attribute), 109
import_dataset() (bioblend.galaxy.objects.wrappers.History method), 98
import_history() (bioblend.galaxy.histories.HistoryClient method), 31
import_new() (bioblend.galaxy.objects.client.ObjWorkflowClient method), 92
import_shared() (bioblend.galaxy.objects.client.ObjWorkflowClient method), 92
import_shared_workflow() (bioblend.galaxy.workflows.WorkflowClient method), 78
import_workflow_dict() (bioblend.galaxy.workflows.WorkflowClient method), 79
import_workflow_from_local_path() (bioblend.galaxy.workflows.WorkflowClient method), 79
init_logging() (in module bioblend), 133
input_labels (bioblend.galaxy.objects.wrappers.Workflow property), 106
input_labels_to_ids (bioblend.galaxy.objects.wrappers.Workflow property), 106
input_steps (bioblend.galaxy.objects.wrappers.Step attribute), 104
inputs (bioblend.galaxy.objects.wrappers.Workflow attribute), 106
install_dependencies() (bioblend.galaxy.tools.ToolClient method), 61
install_genome() (bioblend.galaxy.genomes.GenomeClient method), 22
install_repository_revision() (bioblend.galaxy.toolshed.ToolShedClient method), 70
inv_dag (bioblend.galaxy.objects.wrappers.Workflow attribute), 106
InvocationClient (class in bioblend.galaxy.invocations), 35
BioBlend Documentation, Release 1.2.0

invoke() (bioblend.galaxy.objects.wrappers.Workflow method), 106
invoke_workflow() (bioblend.galaxy.workflows.WorkflowClient method), 80
is_mapped (bioblend.galaxy.objects.wrappers_WRAPPER property), 109
is_modified (bioblend.galaxy.objects.wrappers.Wrapper attribute), 110
is_runnable (bioblend.galaxy.objects.wrappers.Workflow property), 108

J
Job (class in bioblend.galaxy.objects.wrappers), 101
JobsClient (class in bioblend.galaxy.jobs), 41

L
Library (class in bioblend.galaxy.objects.wrappers), 101
LibraryClient (class in bioblend.galaxy.libraries), 47
LibraryContentInfo (class in bioblend.galaxy.objects.wrappers), 103
LibraryDataset (class in bioblend.galaxy.objects.wrappers), 103
LibraryDatasetDatasetAssociation (class in bioblend.galaxy.objects.wrappers), 103
LibraryDatasetElement (class in bioblend.galaxy.dataset_collections), 17
LibraryPreview (class in bioblend.galaxy.objects.wrappers), 104
list() (bioblend.galaxy.objects.client.ObjClient method), 86
list() (bioblend.galaxy.objects.client.ObjDatasetClient method), 87
list() (bioblend.galaxy.objects.client.ObjDatasetCollectionClient method), 88
list() (bioblend.galaxy.objects.client.ObjHistoryClient method), 89
list() (bioblend.galaxy.objects.client.ObjInvocationClient method), 90
list() (bioblend.galaxy.objects.client.ObjJobClient method), 91
list() (bioblend.galaxy.objects.client.ObjLibraryClient method), 91
list() (bioblend.galaxy.objects.client.ObjToolClient method), 91
list() (bioblend.galaxy.objects.client.ObjWorkflowClient method), 93

M
misc_info (bioblend.galaxy.objects.wrappers.Dataset attribute), 94
missing_ids (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
module bioblend, 133

Index 151
refactor_workflow()
(refactor_workflow method), 82
refresh() (refresh method), 82
refresh() (refresh method), 94
refresh() (refresh method), 95
refresh() (refresh method), 96
refresh() (refresh method), 96
reload() (reload method), 62
reload_data_table()
(reload_data_table method), 67
reload_toolbox() (reload_toolbox method), 11
report_error() (report_error method), 44
repository_revisions()
(repository_revisions method), 127
requirements() (requirements method), 63
rerun_invocation() (rerun_invocation method), 38
rerun_job() (rerun_job method), 44
resume_job() (resume_job method), 44
RolesClient (RolesClient class), 57
root_folder (root_folder property), 102
run() (run method), 105
run_invocation_step_action()
(run_invocation_step_action method), 39
run_invocation_step_action()
(run_invocation_step_action method), 83
run_tool() (run_tool method), 63
set_dataset_permissions()
(set_dataset_permissions method), 49
set_file_logger() (set_file_logger method), 133
set_library_permissions()
(set_library_permissions method), 50
set_permissions() (set_permissions method), 19
set_stream_logger() (set_stream_logger method), 133
show_category()
(show_category method), 122
show_data_table()
(show_data_table method), 67
show_dataset()
(show_dataset method), 14
show_dataset()
(show_dataset method), 31
show_dataset()
(show_dataset method), 50
show_dataset_collection()
(show_dataset_collection method), 16
show_dataset_collection()
(show_dataset_collection method), 32
show_dataset_provenance()
(show_dataset_provenance method), 32
show_folder()
(show_folder method), 19
show_folder()
(show_folder method), 50
show_form() (show_form method), 20
show_genome() (show_genome method), 22
show_group() (show_group method), 25
show_history() (show_history method), 33
show_in_tool_panel (show_in_tool_panel method), 109
show_invocation() (show_invocation method), 39
show_invocation() (show_invocation method), 83
show_invocation() (show_invocation method), 40
show_invocation() (show_invocation method), 84
show_job() (show_job method), 49
show_job_lock() (bioblend.galaxy.jobs.JobsClient method), 46
show_library() (bioblend.galaxy.libraries.LibraryClient method), 50
show_matching_datasets() (bioblend.galaxy.histories.HistoryClient method), 33
show_quota() (bioblend.galaxy.quotas.QuotaClient method), 56
show_repository() (bioblend.galaxy.toolshed.ToolShedClient method), 71
show_repository() (bioblend.galaxy.toolshed.repositories.ToolShedRepositoryClient method), 129
show_repository_revision() (bioblend.galaxy.toolshed.repositories.ToolShedRepositoryClient method), 129
show_role() (bioblend.galaxy.roles.RolesClient method), 58
show_tool() (bioblend.galaxy.tools.ToolClient method), 65
show_user() (bioblend.galaxy.users.UserClient method), 74
show_versions() (bioblend.galaxy.workflows.WorkflowClient method), 85
show_visualization() (bioblend.galaxy.visual.VisualClient method), 75
show_workflow() (bioblend.galaxy.workflows.WorkflowClient method), 85
sink_ids (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
sorted_step_ids() (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
source_ids (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
SRC (bioblend.galaxy.objects.wrappers.HistoryDatasetAssociation attribute), 99
SRC (bioblend.galaxy.objects.wrappers.HistoryDatasetCollectionAssociation attribute), 100
SRC (bioblend.galaxy.objects.wrappers.LibraryDataset attribute), 103
SRC (bioblend.galaxy.objects.wrappers.LibraryDatasetDatasetAssociation attribute), 104
state (bioblend.galaxy.objects.wrappers.Dataset attribute), 94
Step (class in bioblend.galaxy.objects.wrappers), 104
steps (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
summarize_toolbox() (bioblend.galaxy.tool_dependencies.ToolDependenciesClient method), 68
synopsis (bioblend.galaxy.objects.wrappers.Library attribute), 102
T
tags (bioblend.galaxy.objects.wrappers.History attribute), 98
tags (bioblend.galaxy.objects.wrappers.Workflow attribute), 108
tags (bioblend.galaxy.objects.wrappers.WorkflowPreview attribute), 109
TimeoutException, 133
to_dict() (bioblend.galaxy.dataset_collections.CollectionDescription method), 15
to_dict() (bioblend.galaxy.dataset_collections.CollectionElement method), 15
to_json() (bioblend.galaxy.objects.wrappers.Wrapper method), 110
Tool (class in bioblend.galaxy.objects.wrappers), 104
tool_id (bioblend.galaxy.objects.wrappers.Step attribute), 104
tool_ids (bioblend.galaxy.objects.wrappers.Workflow property), 108
tool_inputs (bioblend.galaxy.objects.wrappers.Step attribute), 104
tool_labels_to_ids (bioblend.galaxy.objects.wrappers.Workflow attribute), 109
tool_version (bioblend.galaxy.objects.wrappers.Step attribute), 104
ToolClient (class in bioblend.galaxy.tools), 58
ToolDataClient (class in bioblend.galaxy.tool_data), 66
ToolDependenciesClient (class in bioblend.galaxy.tool_dependencies), 68
ToolShedCategoryClient (class in bioblend.toolshed.categories), 120
ToolShedClient (class in bioblend.galaxy.toolshed), 69
ToolShedInstance (class in bioblend.toolshed), 119
ToolShedRepositoryClient (class in bioblend.galaxy.toolshed.repositories), 123
ToolShedToolClient (class in bioblend.galaxy.toolshed.tools), 131
touch() (bioblend.galaxy.objects.wrappers.Wrapper method), 110
type (bioblend.galaxy.objects.wrappers.Step attribute), 104
U
undelete_history() (bioblend.galaxy.histories.HistoryClient method), 34
undelete_quota() (bioblend.galaxy.quotas.QuotaClient method), 56
uninstall_dependencies() (bioblend.galaxy.tools.ToolClient method), 65
uninstall_repository_revision() (bioblend.galaxy.toolshed.ToolShedClient method), 71
BioBlend Documentation, Release 1.2.0

unmap()  (bioblend.galaxy.objects.wrappers.Wrapper method), 110

unused_dependency_paths()  (bioblend.galaxy.tool_dependencies.ToolDependenciesClient method), 69

update()  (bioblend.galaxy.objects.wrappers.History method), 98

update()  (bioblend.galaxy.objects.wrappers.LibraryDataset method), 103

update_dataset()  (bioblend.galaxy.histories.HistoryClient method), 34

update_dataset_collection()  (bioblend.galaxy.histories.HistoryClient method), 34

update_folder()  (bioblend.galaxy.folders.FoldersClient method), 19

update_group()  (bioblend.galaxy.groups.GroupsClient method), 25

update_history()  (bioblend.galaxy.histories.HistoryClient method), 35

update_job_lock()  (bioblend.galaxy.jobs.JobsClient method), 46

update_library_dataset()  (bioblend.galaxy.libraries.LibraryClient method), 51

update_library()  (bioblend.galaxy.libraries.LibraryClient method), 51

update_library_metadata()  (bioblend.galaxy.libraries.LibraryClient method), 51

update_permissions()  (bioblend.galaxy.datasets.DatasetClient method), 14

update_quota()  (bioblend.galaxy.quotas.QuotaClient method), 56

update_repository()  (bioblend.toolshed.repositories.ToolShedRepositoryClient method), 130

update_repository_metadata()  (bioblend.toolshed.repositories.ToolShedRepositoryClient method), 130

update_user()  (bioblend.galaxy.users.UserClient method), 74

update_workflow()  (bioblend.galaxy.workflows.WorkflowClient method), 85

upload_data()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_dataset()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_dataset_from_library()  (bioblend.galaxy.histories.HistoryClient method), 35

upload_file()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_file()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_file()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_file_contents()  (bioblend.galaxy.libraries.LibraryClient method), 51

upload_file_from_local_path()  (bioblend.galaxy.libraries.LibraryClient method), 51

upload_file_from_server()  (bioblend.galaxy.libraries.LibraryClient method), 51

upload_file_from_url()  (bioblend.galaxy.libraries.LibraryClient method), 51

upload_from_ftp()  (bioblend.galaxy.objects.wrappers.Library method), 103

upload_from_galaxy_filesystem()  (bioblend.galaxy.libraries.LibraryClient method), 53

upload_from_galaxy_fs()  (bioblend.galaxy.objects.wrappers.Library method), 102

upload_from_local()  (bioblend.galaxy.objects.wrappers.Library method), 103

upload_from_url()  (bioblend.galaxy.objects.wrappers.Library method), 103

UserClient  (class in bioblend.galaxy.users), 72

VisualClient  (class in bioblend.galaxy.visual), 75

wait()  (bioblend.galaxy.objects.wrappers.Dataset method), 94

wait_for_dataset()  (bioblend.galaxy.datasets.DatasetClient method), 15

wait_for_dataset()  (bioblend.galaxy.libraries.LibraryClient method), 54

wait_for_dataset_collection()  (bioblend.galaxy.dataset_collections.DatasetCollectionClient method), 16

wait_for_invocation()  (bioblend.galaxy.invocations.InvocationClient method), 40

wait_for_job()  (bioblend.galaxy.jobs.JobsClient method), 46

whoami()  (bioblend.galaxy.config.ConfigClient method), 11

Workflow  (class in bioblend.galaxy.objects.wrappers), 105

WorkflowClient  (class in bioblend.galaxy.workflows), 76

WorkflowPreview  (class in bioblend.galaxy.objects.wrappers), 109

Index 155
wrapped (bioblend.galaxy.objects.wrappers.Wrapper attribute), 110
Wrapper (class in bioblend.galaxy.objects.wrappers), 109