Parsec Documentation

Release 1.0.5

Helena Rasche

Oct 03, 2018

Contents

Contents:

CHAPTER 1

Parsec: Galaxy at the Speed of Light

Command-line utilities to assist in working with Galaxy servers.

1.1 Installation

```
$ pip install galaxy-parsec
$ parsec init
```

1.2 Questions?

1.3 Quick Start

This quick start demonstrates using `parsec` commands to manipulate Galaxy histories and datasets. You will want to install jq if you do not have it already.

1.3.1 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 http://wiki.galaxyproject.org/Learn/API .

First initialize parsec:

```
$ parsec init
```

Once initialized, parsec will be usable from the command line. Please note that an admin account is required for a few actions like creation of data libraries, or access to user API keys. Your configuration must allow access to /api without need for a username or password. More infomration can be found at https://galaxyproject.org/admin/config/performance/production-server/

1.3.2 Introduction To Parsec

Parsec is a set of automatically generated wrappers for BioBlend functions. I found myself writing a large number of small / one-off scripts that invoked simple bioblend functions. These scripts were impossible to compose and use in a linux-friendly manner. I copied and pasted code between all of these utility scripts.

Parsec is the answer to all of these problems. It extracts all of the individual functions I was writing as separate CLI commands that can be piped together, run in parallel, etc.

After installation, running `parsec` will present you with a list of sub-commands you can execute.

```
$ parsec
Usage: parsec [OPTIONS] COMMAND [ARGS]...

  Command line wrappers around BioBlend functions. While this sounds
  unexciting, with parsec and jq you can easily build powerful command line
  scripts.

Options:
  --version               Show the version and exit.
  -v, --verbose           Enables verbose mode.
  --galaxy_instance TEXT  name of galaxy instance from ~/.planemo.yml
                          [required]
  --help                  Show this message and exit.

Commands:
  config
  datasets
  datatypes
  folders
  forms
  ...
```

Each of these commands has more commands under it:

```
$ parsec histories
Usage: parsec histories [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  create_dataset_collection       Create a new dataset collection
  create_history                  Create a new history, optionally setting
                                  the...
  create_history_tag              Create history tag
  delete_dataset                  Mark corresponding dataset as deleted.
  delete_dataset_collection       Mark corresponding dataset collection as...
  delete_history                  Delete a history.
  download_dataset                Deprecated method, use...
  download_history                Download a history export archive.
  export_history                  Start a job to create an export archive
```

(continues on next page)

(continued from previous page)

```
                                    for...
  ...
```

1.3.3 Viewing Histories and Datasets

To get information on the Histories currently in your account, call `history get_histories`, and we will pipe this to a `jq` command which selects the first element from the JSON array.

```
$ parsec histories get_histories | jq '.[0]'
```

Parsec will respond with information about your first history

```
{
  "name": "BuildID=Manual-2017.05.02T16:13 WF=PAP_2017_Comparative_(v1.0)_
↪BOOTSTRAPPED Org=CCS Source=Jenkins",
  "url": "/galaxy/api/histories/548c0777ac615645",
  "annotation": null,
  "model_class": "History",
  "id": "548c0777ac615645",
  "tags": [
    "Automated",
    "Annotation",
    "BICH464"
  ],
  "purged": false,
  "published": false,
  "deleted": false
}
```

This may not be all of the information you were expecting about your history. In that case, you might want to call `show_history` which will show you more details about a single history. You can either manually type `parsec histories show_history 548c0777ac615645`, or we can do this in batch:

```
$ parsec histories get_histories | jq '.[0].id' | xargs -n 1 parsec histories show_
↪history
```

Which pulls out the first history, select the `id` attribute, before passing it to `xargs`. If you have not used it before, `xargs` allows us to execute multiple commands for some input data. Here we execute the command `parsec histories show_history` for each line of input (i.e. each ID returned to us from the jq call). `xargs -n 1` ensures that we will only pass a single ID to a single call of `show_history`. If you were to use `jq '.[].id'` instead of `jq '.[0].id'` it would output the IDs for every history you own. You could then pipe this to xargs and run `show_history` on all of your histories!

```
{
  "annotation": null,
  "contents_url": "/galaxy/api/histories/548c0777ac615645/contents",
  "create_time": "2017-05-02T16:18:21.285382",
  "deleted": false,
  "empty": false,
  "genome_build": null,
  "id": "548c0777ac615645",
  "importable": true,
  "model_class": "History",
  "name": "BuildID=Manual-2017.05.02T16:13 WF=PAP_2017_Comparative_(v1.0)_
↪BOOTSTRAPPED Org=CCS Source=Jenkins",
```

(continues on next page)

(continued from previous page)


```
  "published": false,
  "purged": false,
  "size": 34760258,
  "slug": "buildidmanual-20170502t1613-wfpap2017comparativev10bootstrapped-orgccs-
↪sourcejenkins",
  "state": "ok",
  "state_details": {
    "discarded": 0,
    "empty": 0,
    "error": 0,
    "failed_metadata": 0,
    "new": 0,
    "ok": 29,
    "paused": 0,
    "queued": 0,
    "running": 0,
    "setting_metadata": 0,
    "upload": 0
  },
  "state_ids": {
    "discarded": [
      "a6cc986453fae8ba",
      "f2f9b7b017f20578",
      "70eb5af78c588bd1"
    ],
    "empty": [],
    "error": [
      "d643e34e1114cc52",
      "98ae3d35d73f82c9"
    ],
    "failed_metadata": [],
    "new": [],
    "ok": [
      "e510305efbee5f49",
      "0d595b7c2b6e9b93",
      "d04ac6f949ae266c",
      "175f283ddaeca39c",
      "b34432b8a0847c04",
      "ea7ff5323ddebcb8",
      "3e40a393efafc45c",
      "7ce5ec5d51ef85cb",
      "577e4242cdfbe1aa",
      "193d15527d13f45e",
      "4543f9456af7f0df",
      "5e1293df75b4f95b",
      "a57bae35eca5fbfe",
      "6c306b2ed4533f1f",
      "97c5f81b159505f0",
      "64d1d8e46b4554bd",
      "8e9432496d7e2b43",
      "5c8579257c579aae",
      "243ad216fbfa268e",
      "8336d9eb27b27677",
      "a1d4cc61bdba629d",
      "7f93a80890822fa9",
      "c479b351902302e2",
      "36b60fb58ad24a71",
```


(continues on next page)

(continued from previous page)

```
      "041dd3cb6879f1f7",
      "36992e90715c9c77",
      "4bddfe152467e972",
      "2d9f5c0c36d89e10",
      "e53ad6f3133b2816"
    ],
    "paused": [
      "4a8143557292a233",
      "b0f8a75aa6be2c1d"
    ],
    "queued": [],
    "running": [],
    "setting_metadata": [],
    "upload": []
  },
  "tags": [
    "Automated",
    "Annotation",
    "BICH464"
  ],
  "update_time": "2017-05-02T16:49:07.941097",
  "url": "/galaxy/api/histories/548c0777ac615645",
  "user_id": "f570ade6e7840ba0",
  "username_and_slug": "u/helena-rasche/h/buildidmanual-20170502t1613-
↪wfpap2017comparativev10bootstrapped-orgccs-sourcejenkins"
}
```

So much metadata to play with and filter on! Note that many of these commands have additional flags, for example `parsec histories show_history --help` will tell us that we can also pass the –contents option to retrieve a list of datasets in that history, even filtering on their visibility.

```
$ parsec histories show_history --help
Usage: parsec histories show_history [OPTIONS] HISTORY_ID

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

Options:
  --contents      When ``True``, the complete list of datasets in the given
                  history.
  --deleted TEXT  Used when contents=True, includes deleted datasets in
                  history dataset list
  --visible TEXT  Used when contents=True, includes only visible datasets in
                  history dataset list
  --details TEXT  Used when contents=True, includes dataset details. Set to
                  'all' for the most information
```

Thus with a simple query

```
$ parsec histories show_history 548c0777ac615645 --contents --deleted True | jq -S '.
↪[0]'
```

We see the first deleted dataset in the history.

```
{
  "create_time": "2017-05-02T16:18:54.272050",
```

(continues on next page)

(continued from previous page)


```
  "dataset_id": "93c926a0dabafde3",
  "deleted": true,
  "extension": "fasta",
  "hid": 30,
  "history_content_type": "dataset",
  "history_id": "548c0777ac615645",
  "id": "d643e34e1114cc52",
  "name": "Feature Sequence Export Unique on data 27 and data 20",
  "purged": false,
  "state": "error",
  "type": "file",
  "type_id": "dataset-d643e34e1114cc52",
  "update_time": "2017-05-02T16:47:57.807506",
  "url": "/galaxy/api/histories/548c0777ac615645/contents/d643e34e1114cc52",
  "visible": true
}
```


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:


```
$ parsec datasets_show_dataset 10a4b652da44e82a
{
    "accessible": true,
    "annotation": null,
    "api_type": "file",
    "create_time": "2015-02-27T23:46:27.642906",
    "data_type": "galaxy.datatypes.data.Text",
    "dataset_id": "10a4b652da44e82a",
    "deleted": false,
    "display_apps": [],
    "display_types": [],
    "download_url": "/api/histories/f3c2b0f3ecac9f02/contents/10a4b652da44e82a/display
↪",
    "extension": "fastq",
    "file_ext": "fastq",
    "file_path": null,
    "file_size": 16527060,
    "genome_build": "dm3",
    "hda_ldda": "hda",
    "hid": 1,
    "history_content_type": "dataset",
    "history_id": "f3c2b0f3ecac9f02",
    "id": "10a4b652da44e82a",
    "meta_files": [],
    "metadata_data_lines": 4,
    "metadata_dbkey": "dm3",
    "misc_blurb": "15.8 MB",
    "misc_info": "uploaded fastqsanger file",
    "model_class": "HistoryDatasetAssociation",
    "name": "C1_R2_1.chr4.fq",
    "purged": false,
    "resubmitted": false,
    "state": "ok",
    "tags": [],
```


(continues on next page)

(continued from previous page)

```
    "type": "file",
    "update_time": "2015-02-27T23:46:34.659590",
    "url": "/api/histories/f3c2b0f3ecac9f02/contents/10a4b652da44e82a",
    "uuid": "ccad6f3a-f75d-472f-9142-2d4c39ad1a35",
    "visible": true,
    "visualizations": []
}
```

1.4 On JQ

It is worth it to look at some of the things possible with JQ for a moment. The above example may not be so exciting at first blush, but you can do incredible things with the combination of parsec, jq, and xargs. Here are some examples to consider:

- find all histories with a public link, but not published in the shared-histories section, and print out their history name and the shared link.

```
$ parsec histories get_histories | \
   jq '.[].id' | \
   xargs -n 1 parsec histories show_history | \
   jq '. | select(.published == false) | select(.importable == true) | [.
↪published, .importable, .id, .username_and_slug] | @tsv' -r
```

- reset the API keys for 30 users at once.

```
$ parsec users get_users | \
   jq '.[] | \
   select(.username | contains("janedoe")) | .id' | \
   xargs -n 1 parsec users create_user_apikey
```

- download all of the OK datasets in a set of histories

```
$ parsec histories get_histories | \
   jq '.[].id' | \ # Or other, more complex filtering?
   xargs -n 1 parsec histories show_history | \ # Get history details
   jq '.state_ids.ok[]' | \ # Find OK datasets
   xargs -n 1 parsec datasets download_dataset --file_path '.' --use_default_
↪filename # Download
```

1.4.1 View Workflows

Methods for accessing workflows are grouped under `GalaxyInstance.workflows.*`.

To get information on the Workflows currently in your account, use:

```
$ parsec workflows get_workflows
[
    {
        'id': 'e8b85ad72aefca86',
        'name': u"TopHat + cufflinks part 1",
        'url': '/api/workflows/e8b85ad72aefca86'
    },
    {
```

(continues on next page)

(continued from previous page)

```
        'id': 'b0631c44aa74526d',
         'name': 'CuffDiff',
         'url': '/api/workflows/b0631c44aa74526d'
    }
]
```

For example, to further investigate a workflow, we can request:

```
$ parsec workflows show_workflow ded67e5aa1371841 | jq 'del(.steps)'
```

The workflow output is generally quite large as it embeds a full copy of the workflow. In the above JQ command I have removed the `steps` attribute from the output for brevity.


```
{
  "annotation": "",
  "model_class": "StoredWorkflow",
  "latest_workflow_uuid": "94c40212-c4bb-43b7-a43b-eadc1a3b2894",
  "id": "ded67e5aa1371841",
  "url": "/galaxy/api/workflows/ded67e5aa1371841",
  "deleted": false,
  "tags": [],
  "owner": "helena-rasche",
  "name": "PAP 2017 Functional (v8.15)",
  "inputs": {
    "0": {
      "value": "",
      "uuid": "9397916e-afb7-4e48-b89e-d4c99bf202de",
      "label": "Apollo Organism JSON File"
    },
    "2": {
      "value": "",
      "uuid": "eca835c6-328a-4698-a387-d0719b24d19d",
      "label": "Genome Sequence"
    },
    "1": {
      "value": "",
      "uuid": "5511d038-e96b-49b2-998a-d037935f6e06",
      "label": "Annotation Set"
    }
  },
  "published": false
}
```


1.4.2 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:

```
$ parsec users get_users
[
    {
        "username": "test",
        "model_class": "User",
```

(continues on next page)

(continued from previous page)

```
        "email": "test@local.host",
        "id": "f2db41e1fa331b3e"
    },
    ...
]
```

1.4.3 In Depth Example

As a more detailed example, we'll launch a simple workflow.

Step 1. What are the Inputs

```
$ parsec workflows show_workflow ded67e5aa1371841 | jq .inputs > inputs.json
```

In practice this file probably looks similar to this:

```
{
  "0": {
    "value": "",
    "uuid": "9397916e-afb7-4e48-b89e-d4c99bf202de",
    "label": "Apollo Organism JSON File"
  },
  "2": {
    "value": "",
    "uuid": "eca835c6-328a-4698-a387-d0719b24d19d",
    "label": "Genome Sequence"
  },
  "1": {
    "value": "",
    "uuid": "5511d038-e96b-49b2-998a-d037935f6e06",
    "label": "Annotation Set"
  }
}
```

Step 2: Prepare History and Load Datasets

First, we'll create a history to manage all of our work:

```
$ HISTORY_ID=$(parsec histories create_history | jq .id)
$ parsec histories update_history --name 'Parsec test'
```

Next we have to fetch some datasets. You could upload them:

```
$ parsec tools upload_file my-file.gff3 $HISTORY_ID
```

But in my case, I need to run a tool which produces them:

```
JOB_ID=$(parsec tools run_tool $HISTORY_ID edu.tamu.cpt2.webapollo.export \
   '{"org_source|source_select": "direct", "org_source|org_raw": "Miro"}' | \
   jq .id)

$ parsec jobs show_job .outputs $JOB_ID
```

By storing the job ID in a variable, we can make repeated requests to check on it. The second parsec statement fetches the output datasets from this step.


```
{
  "fasta_out": {
    "id": "61513e15ce98c986",
    "src": "hda",
    "uuid": "0de1442b-c410-4a38-b9ca-49cff973d9b8"
  },
  "gff_out": {
    "id": "62ee69adcf74378c",
    "src": "hda",
    "uuid": "887aaf6f-ed07-4ee8-a396-c16612f83d83"
  },
  "json_out": {
    "id": "1f73e96543934ac8",
    "src": "hda",
    "uuid": "3be3d364-83c5-4a23-87fa-ebd8c27f2094"
  }
}
```


Step 3: Invoking the Workflow

Remembering back to the inputs in step 1, we will match them up and create an `inputs.json` file

- 0 / organism json file => json_out
- 1 / genome sequence => gff_out
- 2 / annotation set => fasta_out

This gives us an inputs.json that looks like so:


```
{
  "0": {
    "id": "1f73e96543934ac8",
    "src": "hda"
  },
  "1": {
    "id": "62ee69adcf74378c",
    "src": "hda"
  },
  "2": {
    "id": "61513e15ce98c986",
    "src": "hda"
  }
}
```


We can now invoke our workflow using parsec! Since the inputs is a JSON parameter, it can be supplied many different ways for your convenience. All of the following behave identically.

```
$ cat params.json | parsec jobs search_jobs -; # Stdin
$ parsec jobs search_jobs params.json; # Filename
$ parsec jobs search_jobs $(cat params.json); # String argument
```

Running the invocation:

```
$ parsec workflows invoke_workflow ded67e5aa1371841 --inputs inputs.json --history_id
↪$HISTORY_ID
```

Produces a very succinct workflow launch output:

```
{
    "uuid": "94246003-2f8b-11e7-9427-20474784cc00",
    "state": "new",
    "workflow_id": "3daf5606d767a471",
    "id": "c7f60cfda02f0f46",
    "update_time": "2017-05-02T23:03:39.693288",
    "model_class": "WorkflowInvocation",
    "history_id": "0d17c6f8cd8d49a5"
}
```

We can now use parsec to check on the status of all of the datasets:

```
$ parsec workflows show_invocation 3daf5606d767a471 c7f60cfda02f0f46 | jq '.steps[].
↪state' | sort | uniq -c
   3 "running"
  72 "new"
   3 null
   1 "ok"
```

Or we can use one of the utility scripts to wait on that workflow to finish before continuing on to some other task:

```
$ parsec utils wait_on_invocation 3daf5606d767a471 c7f60cfda02f0f46 && ...
```

1.5 License

1.6 Support

This material is based upon work supported by the National Science Foundation under Grant Number (Award 1565146)

CHAPTER 2

Cookbook

This page will contain more easy "recipes" for using parsec as time goes on. Short tips and tricks that can help you use it more effectively, or short recipes that can document how to do more complex tasks.

2.1 Talking to multiple Galaxies

If you are regularly switching between multiple Galaxy instances, you'll probably want to take advantage of the environment variable for specifying a Galaxy instance. E.g.:

```
$ PARSEC_GALAXY_INSTANCE=uni-admin parsec config get_config | jq .brand
"Internal"
$ PARSEC_GALAXY_INSTANCE=uni-public parsec config get_config | jq .brand
"Public"
```

You could easily set these at the top of a parsec script you've built and all commands from there on would talk to the same Galaxy instance.

2.2 Capturing execution state as XUnit Output

If you find yourself building a pipeline with `parsec` and `jq`, you might find yourself wanting to produce the output in a machine-legible format such as XUnit. `parsec` now ships with a (very alpha) script to help with this. `parsec utils xunit_xargs` provides an `xargs`-like experience, except it produces XUnit formatted output. We'll run through a quick example of this:

```
parsec histories get_histories | \
jq '.[].id' -r | \
head -n 3 | \
parsec utils xunit_xargs parsec histories get_status \| jq .percent_complete \|␣
↪parsec utils cmp eq 100
```

This command will fetch the first three histories, and then attempt to run `parsec histories get_status <history_id> | jq .percent_complete` for each history id passed in.

```
<?xml version="1.0" ?>
<testsuites errors="0" failures="1" tests="3" time="1.6388022899627686">
  <testsuite errors="0" failures="1" name="Parsec XX" skipped="0" tests="3" time="1.
↪6388022899627686">
    <testcase classname="parsec.histories.get_status.769f01a3981796db_|.jq..percent_
↪complete.|.parsec.utils.cmp.eq.100" name="parsec.histories.get_status.
↪769f01a3981796db_" time="0.537762"/>
    <testcase classname="parsec.histories.get_status.83fbc32772cb5fcf_|.jq..percent_
↪complete.|.parsec.utils.cmp.eq.100" name="parsec.histories.get_status.
↪83fbc32772cb5fcf_" time="0.534841"/>
    <testcase classname="parsec.histories.get_status.90c9282cb8718062_|.jq..percent_
↪complete.|.parsec.utils.cmp.eq.100" name="parsec.histories.get_status.
↪90c9282cb8718062_" time="0.566199">
      <failure message="Command 'parsec histories get_status 90c9282cb8718062 | jq .
↪percent_complete | parsec utils cmp eq 100' returned non-zero exit status 1" type=
↪"failure">Traceback (most recent call last):

  File &quot;xunit_xargs.py&quot;, line 95, in cli
    output = check_output(' '.join(built_command), shell=True, stderr=stderr)

  File &quot;/usr/lib/python3.5/subprocess.py&quot;, line 626, in check_output
    **kwargs).stdout

  File &quot;/usr/lib/python3.5/subprocess.py&quot;, line 708, in run
    output=stdout, stderr=stderr)

subprocess.CalledProcessError: Command 'parsec histories get_status 90c9282cb8718062␣
↪| jq .percent_complete | parsec utils cmp eq 100' returned non-zero exit status 1
</failure>
      <system-err>97.82608695652173 != 100.0</system-err>
    </testcase>
  </testsuite>
</testsuites>
```

Here we can see the example output, every history ID that went in came out as a test case. One of them didn't pass a test we cared about and was marked as a failure.

CHAPTER 3

Commands

parsec is a set of wrappers for BioBlend's API. It builds a set of small, useful utilities for talking to Galaxy servers. Each utility is implemented as a subcommand of `parsec`. This section of the documentation describes these commands.

3.1 config

This section is auto-generated from the help text for the parsec command `config`.

3.1.1 `get_config` command

Usage:

```
parsec config get_config [OPTIONS]
```

Help

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

Output

A list of attributes. For example:

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

(continues on next page)

(continued from previous page)

```
u'terms_url': None,
u'user_library_import_dir': None,
u'wiki_url': u'https://galaxyproject.org/'}
```

Options:

```
-h, --help  Show this message and exit.
```

3.1.2 `get_version` command

Usage:

```
parsec config get_version [OPTIONS]
```

Help

Get the current version of the Galaxy instance. This functionality is available since Galaxy `release_15.03`.

Output

Version of the Galaxy instance

For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.2 datasets

This section is auto-generated from the help text for the parsec command `datasets`.

3.2.1 `download_dataset` command

Usage:

```
parsec datasets download_dataset [OPTIONS] DATASET_ID
```

Help

Download a dataset to file or in memory. If the dataset state is not 'ok', a `DatasetStateException` will be thrown.

Output

If a file_path argument is not provided, returns a dict containing the file_content. Otherwise returns nothing.

Options:

```
--file_path TEXT         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   If this argument is True, the exported file will be
                         saved as file_path/%s, where %s is the dataset name.
                         If this argument is False, file_path is assumed to
                         contain the full file path including the filename.
                         [default: True]
--wait_for_completion    This parameter is deprecated and ignored, it will be
                         removed in BioBlend 0.12.  [default: True]
--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.  [default:
                         12000]
-h, --help               Show this message and exit.
```

3.2.2 `show_dataset` command

Usage:

```
parsec datasets show_dataset [OPTIONS] DATASET_ID
```

Help

Get details about a given dataset. This can be a history or a library dataset.

Output

Options:

```
--deleted        Whether to return results for a deleted dataset
--hda_ldda TEXT  Whether to show a history dataset ('hda' - the default) or
                 library dataset ('ldda').  [default: hda]
-h, --help       Show this message and exit.
```

3.3 datatypes

This section is auto-generated from the help text for the parsec command `datatypes`.

3.3.1 `get_datatypes` command

Usage:

```
parsec datatypes get_datatypes [OPTIONS]
```

Help

Get the list of all installed datatypes.

Output

A list of datatype names. For example:

```
[u'snpmatrix',
 u'snptest',
 u'tabular',
 u'taxonomy',
 u'twobit',
 u'txt',
 u'vcf',
 u'wig',
 u'xgmml',
 u'xml']
```

Options:

```
--extension_only TEXT
--upload_only TEXT
-h, --help              Show this message and exit.
```

3.3.2 `get_sniffers` command

Usage:

```
parsec datatypes get_sniffers [OPTIONS]
```

Help

Get the list of all installed sniffers.

Output

A list of sniffer names. For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.4 folders

This section is auto-generated from the help text for the parsec command `folders`.

3.4.1 `create_folder` command

Usage:

```
parsec folders create_folder [OPTIONS] PARENT_FOLDER_ID NAME
```

Help

Create a folder.

Output

details of the updated folder

Options:

```
--description TEXT  folder's description
-h, --help          Show this message and exit.
```

3.4.2 `delete_folder` command

Usage:

```
parsec folders delete_folder [OPTIONS] FOLDER_ID
```

Help

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

Output

Options:

```
--undelete  If set to True, the folder will be undeleted (i.e. the `deleted`
            mark will be removed)
-h, --help  Show this message and exit.
```

3.4.3 `get_permissions` command

Usage:

```
parsec folders get_permissions [OPTIONS] FOLDER_ID SCOPE
```

Help

Get the permissions of a folder.

Output

dictionary including details of the folder

Options:

```
-h, --help  Show this message and exit.
```

3.4.4 `set_permissions` command

Usage:

```
parsec folders set_permissions [OPTIONS] FOLDER_ID
```

Help

Set the permissions of a folder.

Output

dictionary including details of the folder

Options:

```
--action TEXT      action to execute, only "set_permissions" is supported.
                   [default: set_permissions]
--add_ids TEXT     list of role IDs which can add datasets to the folder
--manage_ids TEXT  list of role IDs which can manage datasets in the folder
--modify_ids TEXT  list of role IDs which can modify datasets in the folder
-h, --help         Show this message and exit.
```

3.4.5 `show_folder` command

Usage:

```
parsec folders show_folder [OPTIONS] FOLDER_ID
```

Help

Display information about a folder.

Output

dictionary including details of the folder

Options:

```
--contents  True to get the contents of the folder, rather than just the
            folder details.
-h, --help  Show this message and exit.
```

3.4.6 `update_folder` command

Usage:

```
parsec folders update_folder [OPTIONS] FOLDER_ID NAME
```

Help

Update folder information.

Output

details of the updated folder

Options:

```
--description TEXT  folder's description
-h, --help          Show this message and exit.
```

3.5 forms

This section is auto-generated from the help text for the parsec command `forms`.

3.5.1 `create_form` command

Usage:

```
parsec forms create_form [OPTIONS] FORM_XML_TEXT
```

Help

Create a new form.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.5.2 `get_forms` command

Usage:

```
parsec forms get_forms [OPTIONS]
```

Help

Get the list of all forms.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.5.3 `show_form` command

Usage:

```
parsec forms show_form [OPTIONS] FORM_ID
```

Help

Get details of a given form.

Output

A description of the given form. For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.6 ftpfiles

This section is auto-generated from the help text for the parsec command `ftpfiles`.

3.6.1 `get_ftp_files` command

Usage:

```
parsec ftpfiles get_ftp_files [OPTIONS]
```

Help

Get a list of local files.

Output

A list of dicts with details on individual files on FTP

Options:

```
--deleted TEXT
-h, --help      Show this message and exit.
```

3.7 genomes

This section is auto-generated from the help text for the parsec command `genomes`.

3.7.1 `get_genomes` command

Usage:

```
parsec genomes get_genomes [OPTIONS]
```

Help

Returns a list of installed genomes

Output

Options:

```
-h, --help  Show this message and exit.
```

3.7.2 `install_genome` command

Usage:

```
parsec genomes install_genome [OPTIONS]
```

Help

Download and/or index a genome.

Output

dict(status: 'ok', job: <job ID>) If error: dict(status: 'error', error: <error message>)

Options:

```
--func TEXT           Allowed values: 'download', Download and index; 'index',
                      Index only  [default: download]
--source TEXT         Data source for this build. Can be: UCSC, Ensembl, NCBI,
                      URL
--dbkey TEXT          DB key of the build to download, ignored unless 'UCSC'
                      is specified as the source
--ncbi_name TEXT      NCBI's genome identifier, ignored unless NCBI is
                      specified as the source
--ensembl_dbkey TEXT  Ensembl's genome identifier, ignored unless Ensembl is
                      specified as the source
--url_dbkey TEXT      DB key to use for this build, ignored unless URL is
                      specified as the source
--indexers TEXT       POST array of indexers to run after downloading
                      (indexers[] = first, indexers[] = second, ...)
-h, --help            Show this message and exit.
```

3.7.3 `show_genome` command

Usage:

```
parsec genomes show_genome [OPTIONS] ID
```

Help

Returns information about build <id>

Output

Options:

```
--num TEXT    num
--chrom TEXT  chrom
--low TEXT    low
--high TEXT   high
-h, --help    Show this message and exit.
```

3.8 groups

This section is auto-generated from the help text for the parsec command `groups`.

3.8.1 `add_group_role` command

Usage:

```
parsec groups add_group_role [OPTIONS] GROUP_ID ROLE_ID
```

Help

Add a role to the given group.

Output

Added group role's info

Options:

```
-h, --help  Show this message and exit.
```

3.8.2 `add_group_user` command

Usage:

```
parsec groups add_group_user [OPTIONS] GROUP_ID USER_ID
```

Help

Add a user to the given group.

Output

Added group user's info

Options:

```
-h, --help  Show this message and exit.
```

3.8.3 `create_group` command

Usage:

```
parsec groups create_group [OPTIONS] GROUP_NAME
```

Help

Create a new group.

Output

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

```
[{u'id': u'7c9636938c3e83bf',
  u'model_class': u'Group',
  u'name': u'My Group Name',
  u'url': u'/api/groups/7c9636938c3e83bf'}]
```

Options:

```
--user_ids TEXT  A list of encoded user IDs to add to the new group
--role_ids TEXT  A list of encoded role IDs to add to the new group
-h, --help       Show this message and exit.
```

3.8.4 `delete_group_role` command

Usage:

```
parsec groups delete_group_role [OPTIONS] GROUP_ID ROLE_ID
```

Help

Remove a role from the given group.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.8.5 `delete_group_user` command

Usage:

```
parsec groups delete_group_user [OPTIONS] GROUP_ID USER_ID
```

Help

Remove a user from the given group.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.8.6 `get_group_roles` command

Usage:

```
parsec groups get_group_roles [OPTIONS] GROUP_ID
```

Help

Get the list of roles associated to the given group.

Output

List of group roles' info

Options:

```
-h, --help  Show this message and exit.
```

3.8.7 `get_group_users` command

Usage:

```
parsec groups get_group_users [OPTIONS] GROUP_ID
```

Help

Get the list of users associated to the given group.

Output

List of group users' info

Options:

```
-h, --help  Show this message and exit.
```

3.8.8 `get_groups` command

Usage:

```
parsec groups get_groups [OPTIONS]
```

Help

Get all (not deleted) groups.

Output

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

```
[{'id': '33abac023ff186c2',
  'model_class': 'Group',
  'name': 'Listeria',
  'url': '/api/groups/33abac023ff186c2'},
 {'id': '73187219cd372cf8',
  'model_class': 'Group',
  'name': 'LPN',
  'url': '/api/groups/73187219cd372cf8'}]
```

Options:

```
-h, --help  Show this message and exit.
```

3.8.9 `show_group` command

Usage:

```
parsec groups show_group [OPTIONS] GROUP_ID
```

Help

Get details of a given group.

Output

A description of group For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.8.10 `update_group` command

Usage:

```
parsec groups update_group [OPTIONS] GROUP_ID
```

Help

Update a group.

Output

Options:

```
--group_name TEXT  A new name for the group. If None, the group name is not
                   changed.
--user_ids TEXT    New list of encoded user IDs for the group. It will
                   substitute the previous list of users (with [] if not
                   specified)
--role_ids TEXT    New list of encoded role IDs for the group. It will
                   substitute the previous list of roles (with [] if not
                   specified)
-h, --help         Show this message and exit.
```

3.9 histories

This section is auto-generated from the help text for the parsec command `histories`.

3.9.1 `create_dataset_collection` command

Usage:

```
parsec histories create_dataset_collection [OPTIONS] HISTORY_ID
```

Help

Create a new dataset collection

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.2 `create_history` command

Usage:

```
parsec histories create_history [OPTIONS]
```

Help

Create a new history, optionally setting the `name`.

Output

Dictionary containing information about newly created history

Options:

```
--name TEXT  Optional name for new history
-h, --help   Show this message and exit.
```

3.9.3 `create_history_tag` command

Usage:

```
parsec histories create_history_tag [OPTIONS] HISTORY_ID TAG
```

Help

Create history tag

Output

A dictionary with information regarding the tag. For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.9.4 `delete_dataset` command

Usage:

```
parsec histories delete_dataset [OPTIONS] HISTORY_ID DATASET_ID
```

Help

Mark corresponding dataset as deleted.

Output

Options:

```
--purge     if ``True``, also purge (permanently delete) the dataset
-h, --help  Show this message and exit.
```

3.9.5 `delete_dataset_collection` command

Usage:

```
parsec histories delete_dataset_collection [OPTIONS] HISTORY_ID
```

Help

Mark corresponding dataset collection as deleted.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.6 `delete_history` command

Usage:

```
parsec histories delete_history [OPTIONS] HISTORY_ID
```

Help

Delete a history.

Output

Options:

```
--purge     if ``True``, also purge (permanently delete) the history
-h, --help  Show this message and exit.
```

3.9.7 `download_dataset` command

Usage:

```
parsec histories download_dataset [OPTIONS] HISTORY_ID DATASET_ID
```

Help

Deprecated since version 0.8.0: Use `download_dataset()` instead.

Output

Options:

```
--use_default_filename TEXT  [default: True]
-h, --help                   Show this message and exit.
```

3.9.8 `download_history` command

Usage:

```
parsec histories download_history [OPTIONS] HISTORY_ID JEHA_ID OUTF
```

Help

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

Output

Options:

```
--chunk_size INTEGER  how many bytes at a time should be read into memory
                      [default: 4096]
-h, --help            Show this message and exit.
```

3.9.9 `export_history` command

Usage:

```
parsec histories export_history [OPTIONS] HISTORY_ID
```

Help

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

Output

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

Options:

```
--gzip              create .tar.gz archive if ``True``, else .tar  [default:
                    True]
--include_hidden    whether to include hidden datasets in the export
--include_deleted   whether to include deleted datasets in the export
--wait              if ``True``, block until the export is ready; else, return
                    immediately
-h, --help          Show this message and exit.
```

3.9.10 `get_current_history` command

Usage:

```
parsec histories get_current_history [OPTIONS]
```

Help

Deprecated since version 0.5.2: Use `get_most_recently_used_history()` instead.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.11 `get_histories` command

Usage:

```
parsec histories get_histories [OPTIONS]
```

Help

Get all histories or filter the specific one(s) via the provided `name` or `history_id`. Provide only one argument, `name` or `history_id`, but not both.

Output

Return a list of history element dicts. If more than one history matches the given `name`, return the list of all the histories with the given name

Options:

```
--history_id TEXT  Encoded history ID to filter on
--name TEXT        Name of history to filter on
--deleted TEXT
-h, --help         Show this message and exit.
```

3.9.12 `get_most_recently_used_history` command

Usage:

```
parsec histories get_most_recently_used_history [OPTIONS]
```

Help

Returns the current user's most recently used history (not deleted).

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.13 `get_status` command

Usage:

```
parsec histories get_status [OPTIONS] HISTORY_ID
```

Help

Returns the state of this history

Output

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.

Options:

```
-h, --help  Show this message and exit.
```

3.9.14 `show_dataset` command

Usage:

```
parsec histories show_dataset [OPTIONS] HISTORY_ID DATASET_ID
```

Help

Get details about a given history dataset.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.15 `show_dataset_collection` command

Usage:

```
parsec histories show_dataset_collection [OPTIONS] HISTORY_ID
```

Help

Get details about a given history dataset collection.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.16 `show_dataset_provenance` command

Usage:

```
parsec histories show_dataset_provenance [OPTIONS] HISTORY_ID DATASET_ID
```

Help

Get details related to how dataset was created (`id`, `job_id`, `tool_id`, `stdout`, `stderr`, `parameters`, `inputs`, etc...).

Output

Options:

```
--follow    If ``follow`` is ``True``, recursively fetch dataset provenance
            information for all inputs and their inputs, etc...
-h, --help  Show this message and exit.
```

3.9.17 `show_history` command

Usage:

```
parsec histories show_history [OPTIONS] HISTORY_ID
```

Help

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

Output

details of the given history

Options:

```
--contents      When ``True``, the complete list of datasets in the given
                history.
--deleted TEXT  Used when contents=True, includes deleted datasets in history
                dataset list
--visible TEXT  Used when contents=True, includes only visible datasets in
                history dataset list
--details TEXT  Used when contents=True, includes dataset details. Set to
                'all' for the most information
--types TEXT    ???
-h, --help      Show this message and exit.
```

3.9.18 `show_matching_datasets` command

Usage:

```
parsec histories show_matching_datasets [OPTIONS] HISTORY_ID
```

Help

Get dataset details for matching datasets within a history.

Output

Options:

```
--name_filter TEXT  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
-h, --help          Show this message and exit.
```

3.9.19 `undelete_history` command

Usage:

```
parsec histories undelete_history [OPTIONS] HISTORY_ID
```

Help

Undelete a history

Output

Options:

```
-h, --help  Show this message and exit.
```

3.9.20 `update_dataset` command

Usage:

```
parsec histories update_dataset [OPTIONS] HISTORY_ID DATASET_ID
```

Help

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

Output

details of the updated dataset (for Galaxy release_15.01 and earlier only the updated attributes)

> **Warning:** The return value was changed in BioBlend v0.8.0, previously it was the status code (type int).

Options:

```
--annotation TEXT    Replace history dataset annotation with given string
--deleted            Mark or unmark history dataset as deleted
--genome_build TEXT  Replace history dataset genome build (dbkey)
--name TEXT          Replace history dataset name with the given string
--visible            Mark or unmark history dataset as visible
-h, --help           Show this message and exit.
```

3.9.21 `update_dataset_collection` command

Usage:

```
parsec histories update_dataset_collection [OPTIONS] HISTORY_ID
```

Help

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

Output

the updated dataset collection attributes

> **Warning:** The return value was changed in BioBlend v0.8.0, previously it was the status code (type int).

Options:

```
--deleted    Mark or unmark history dataset collection as deleted
--name TEXT  Replace history dataset collection name with the given string
--visible    Mark or unmark history dataset collection as visible
-h, --help   Show this message and exit.
```

3.9.22 `update_history` command

Usage:

```
parsec histories update_history [OPTIONS] HISTORY_ID
```

Help

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

Output

details of the updated history (for Galaxy release_15.01 and earlier only the updated attributes)

> **Warning:** The return value was changed in BioBlend v0.8.0, previously it was the status code (type int).

Options:

```
--annotation TEXT  Replace history annotation with given string
--deleted          Mark or unmark history as deleted
--importable       Mark or unmark history as importable
--name TEXT        Replace history name with the given string
--published        Mark or unmark history as published
--purged           If True, mark history as purged (permanently deleted).
                   Ignored on Galaxy release_15.01 and earlier
--tags TEXT        Replace history tags with the given list
-h, --help         Show this message and exit.
```

3.9.23 `upload_dataset_from_library` command

Usage:

```
parsec histories upload_dataset_from_library [OPTIONS] HISTORY_ID
```

Help

Upload a dataset into the history from a library. Requires the library dataset ID, which can be obtained from the library contents.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.10 jobs

This section is auto-generated from the help text for the parsec command `jobs`.

3.10.1 `get_jobs` command

Usage:

```
parsec jobs get_jobs [OPTIONS]
```

Help

Get the list of jobs of the current user.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.10.2 `get_state` command

Usage:

```
parsec jobs get_state [OPTIONS] JOB_ID
```

Help

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

Output

> **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.

Options:

```
-h, --help  Show this message and exit.
```

3.10.3 `search_jobs` command

Usage:

```
parsec jobs search_jobs [OPTIONS] JOB_INFO
```

Help

Return jobs for the current user based payload content.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.10.4 `show_job` command

Usage:

```
parsec jobs show_job [OPTIONS] JOB_ID
```

Help

Get details of a given job of the current user.

Output

A description of the given job. For example:

```
{u'create_time': u'2014-03-01T16:17:29.828624',
 u'exit_code': 0,
 u'id': u'a799d38679e985db',
 u'inputs': {u'input': {u'id': u'ebfb8f50c6abde6d',
   u'src': u'hda'}},
 u'model_class': u'Job',
 u'outputs': {u'output': {u'id': u'a799d38679e985db',
   u'src': u'hda'}},
 u'params': {u'chromInfo': u'"/opt/galaxy-central/tool-data/shared/ucsc/
↪chrom/?.len"',
   u'dbkey': u'"?"',
   u'seq_col': u'"2"',
   u'title_col': u'["1"]'},
 u'state': u'ok',
 u'tool_id': u'tab2fasta',
 u'update_time': u'2014-03-01T16:17:31.930728'}
```

Options:

```
--full_details  when ``True``, the complete list of details for the given job.
-h, --help      Show this message and exit.
```

3.11 libraries

This section is auto-generated from the help text for the parsec command `libraries`.

3.11.1 `copy_from_dataset` command

Usage:

```
parsec libraries copy_from_dataset [OPTIONS] LIBRARY_ID DATASET_ID
```

Help

Copy a Galaxy dataset into a library.

Output

Options:

```
--folder_id TEXT  id of the folder where to place the uploaded files. If not
                  provided, the root folder will be used
--message TEXT    message for copying action
-h, --help        Show this message and exit.
```

3.11.2 `create_folder` command

Usage:

```
parsec libraries create_folder [OPTIONS] LIBRARY_ID FOLDER_NAME
```

Help

Create a folder in a library.

Output

Options:

```
--description TEXT     description of the new folder in the data library
--base_folder_id TEXT  id of the folder where to create the new folder. If not
                       provided, the root folder will be used
-h, --help             Show this message and exit.
```

3.11.3 `create_library` command

Usage:

```
parsec libraries create_library [OPTIONS] NAME
```

Help

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

Output

Details of the created library. For example:

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

Options:

```
--description TEXT  Optional data library description
--synopsis TEXT     Optional data library synopsis
-h, --help          Show this message and exit.
```

3.11.4 `delete_library` command

Usage:

```
parsec libraries delete_library [OPTIONS] LIBRARY_ID
```

Help

Delete a data library.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.11.5 `delete_library_dataset` command

Usage:

```
parsec libraries delete_library_dataset [OPTIONS] LIBRARY_ID DATASET_ID
```

Help

Delete a library dataset in a data library.

Output

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

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

Options:

```
--purged    Indicate that the dataset should be purged (permanently deleted)
-h, --help  Show this message and exit.
```

3.11.6 `get_folders` command

Usage:

```
parsec libraries get_folders [OPTIONS] LIBRARY_ID
```

Help

Get all the folders or filter specific one(s) via the provided `name` or `folder_id` in data library with id `library_id`. Provide only one argument: `name` or `folder_id`, but not both.

Output

list of dicts each containing basic information about a folder

Options:

```
--folder_id TEXT  filter for folder by folder id
--name TEXT       filter for folder by name. For ``name`` specify the full
                  path of the folder starting from the library's root folder,
                  e.g. ``/subfolder/subsubfolder``.
-h, --help        Show this message and exit.
```

3.11.7 `get_libraries` command

Usage:

```
parsec libraries get_libraries [OPTIONS]
```

Help

Get all the libraries or filter for specific one(s) via the provided name or ID. Provide only one argument: `name` or `library_id`, but not both.

Output

list of dicts each containing basic information about a library

Options:

```
--library_id TEXT  filter for library by library id
--name TEXT        If ``name`` is set and multiple names match the given name,
                   all the libraries matching the argument will be returned
--deleted          If set to ``True``, return libraries that have been deleted
-h, --help         Show this message and exit.
```

3.11.8 `get_library_permissions` command

Usage:

```
parsec libraries get_library_permissions [OPTIONS] LIBRARY_ID
```

Help

Get the permessions for a library.

Output

dictionary with all applicable permissions' values

Options:

```
-h, --help  Show this message and exit.
```

3.11.9 `set_library_permissions` command

Usage:

```
parsec libraries set_library_permissions [OPTIONS] LIBRARY_ID
```

Help

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

Output

Options:

```
--access_in TEXT  list of role ids
--modify_in TEXT  list of role ids
--add_in TEXT     list of role ids
--manage_in TEXT  list of role ids
-h, --help        Show this message and exit.
```

3.11.10 `show_dataset` command

Usage:

```
parsec libraries show_dataset [OPTIONS] LIBRARY_ID DATASET_ID
```

Help

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

Output

A dictionary containing information about the dataset in the library

Options:

```
-h, --help  Show this message and exit.
```

3.11.11 `show_folder` command

Usage:

```
parsec libraries show_folder [OPTIONS] LIBRARY_ID FOLDER_ID
```

Help

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

Output

Options:

```
-h, --help  Show this message and exit.
```

3.11.12 `show_library` command

Usage:

```
parsec libraries show_library [OPTIONS] LIBRARY_ID
```

Help

Get information about a library.

Output

details of the given library

Options:

```
--contents  True if want to get contents of the library (rather than just the
            library details)
-h, --help  Show this message and exit.
```

3.11.13 `upload_file_contents` command

Usage:

```
parsec libraries upload_file_contents [OPTIONS] LIBRARY_ID PASTED_CONTENT
```

Help

Upload pasted_content to a data library as a new file.

Output

Options:

```
--folder_id TEXT  id of the folder where to place the uploaded file. If not
                  provided, the root folder will be used
--file_type TEXT  Galaxy file format name  [default: auto]
--dbkey TEXT      Dbkey  [default: ?]
-h, --help        Show this message and exit.
```

3.11.14 `upload_file_from_local_path` command

Usage:

```
parsec libraries upload_file_from_local_path [OPTIONS] LIBRARY_ID
```

Help

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

Output

Options:

```
--folder_id TEXT  id of the folder where to place the uploaded file. If not
                  provided, the root folder will be used
--file_type TEXT  Galaxy file format name  [default: auto]
--dbkey TEXT      Dbkey  [default: ?]
-h, --help        Show this message and exit.
```

3.11.15 `upload_file_from_server` command

Usage:

```
parsec libraries upload_file_from_server [OPTIONS] LIBRARY_ID SERVER_DIR
```

Help

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

Output

Options:

```
--folder_id TEXT       id of the folder where to place the uploaded files. If
                       not provided, the root folder will be used
--file_type TEXT       Galaxy file format name  [default: auto]
--dbkey TEXT           Dbkey  [default: ?]
--link_data_only TEXT  either 'copy_files' (default) or 'link_to_files'.
                       Setting to 'link_to_files' symlinks instead of copying
                       the files
--roles TEXT           ???
-h, --help             Show this message and exit.
```

3.11.16 `upload_file_from_url` command

Usage:

```
parsec libraries upload_file_from_url [OPTIONS] LIBRARY_ID FILE_URL
```

Help

Upload a file to a library from a URL.

Output

Options:

```
--folder_id TEXT  id of the folder where to place the uploaded file. If not
                  provided, the root folder will be used
--file_type TEXT  Galaxy file format name  [default: auto]
--dbkey TEXT      Dbkey  [default: ?]
-h, --help        Show this message and exit.
```

3.11.17 `upload_from_galaxy_filesystem` command

Usage:

```
parsec libraries upload_from_galaxy_filesystem [OPTIONS] LIBRARY_ID
```

Help

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

Output

Options:

```
--folder_id TEXT       id of the folder where to place the uploaded files. If
                       not provided, the root folder will be used
--file_type TEXT       Galaxy file format name  [default: auto]
--dbkey TEXT           Dbkey  [default: ?]
--link_data_only TEXT  either 'copy_files' (default) or 'link_to_files'.
                       Setting to 'link_to_files' symlinks instead of copying
                       the files
--roles TEXT           ???
-h, --help             Show this message and exit.
```

3.11.18 `wait_for_dataset` command

Usage:

```
parsec libraries wait_for_dataset [OPTIONS] LIBRARY_ID DATASET_ID
```

Help

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

Output

A dictionary containing information about the dataset in the library

Options:

```
--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.
                  [default: 12000]
--interval FLOAT  Time (in seconds) to wait between 2 consecutive checks.
                  [default: 3]
-h, --help        Show this message and exit.
```

3.12 quotas

This section is auto-generated from the help text for the parsec command `quotas`.

3.12.1 `create_quota` command

Usage:

```
parsec quotas create_quota [OPTIONS] NAME DESCRIPTION AMOUNT OPERATION
```

Help

Create a new quota

Output

A description of quota. For example:

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

Options:

```
--default TEXT    Whether or not this is a default quota. Valid values are
                  ``no``, ``unregistered``, ``registered``. None is equivalent
                  to ``no``.  [default: no]
--in_users TEXT   A list of user IDs or user emails.
--in_groups TEXT  A list of group IDs or names.
-h, --help        Show this message and exit.
```

3.12.2 `delete_quota` command

Usage:

```
parsec quotas delete_quota [OPTIONS] QUOTA_ID
```

Help

Delete a quota

Output

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

```
"Deleted 1 quotas: Testing-B"
```

Options:

```
-h, --help  Show this message and exit.
```

3.12.3 `get_quotas` command

Usage:

```
parsec quotas get_quotas [OPTIONS]
```

Help

Get a list of quotas

Output

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

```
[{u'id': u'0604c8a56abe9a50',
  u'model_class': u'Quota',
  u'name': u'test ',
  u'url': u'/api/quotas/0604c8a56abe9a50'},
 {u'id': u'1ee267091d0190af',
  u'model_class': u'Quota',
  u'name': u'workshop',
  u'url': u'/api/quotas/1ee267091d0190af'}]
```

Options:

```
--deleted   Only return quota(s) that have been deleted
-h, --help  Show this message and exit.
```

3.12.4 `show_quota` command

Usage:

```
parsec quotas show_quota [OPTIONS] QUOTA_ID
```

Help

Display information on a quota

Output

A description of quota. For example:

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

Options:

```
--deleted   Search for quota in list of ones already marked as deleted
-h, --help  Show this message and exit.
```

3.12.5 `undelete_quota` command

Usage:

```
parsec quotas undelete_quota [OPTIONS] QUOTA_ID
```

Help

Undelete a quota

Output

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

```
"Undeleted 1 quotas: Testing-B"
```

Options:

```
-h, --help  Show this message and exit.
```

3.12.6 `update_quota` command

Usage:

```
parsec quotas update_quota [OPTIONS] QUOTA_ID
```

Help

Update an existing quota

Output

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"
```

Options:

```
--name TEXT         Name for the new quota. This must be unique within a
                    Galaxy instance.
--description TEXT  Quota description. If you supply this parameter, but not
                    the name, an error will be thrown.
--amount TEXT       Quota size (E.g. ``10000MB``, ``99 gb``, ``0.2T``,
                    ``unlimited``)
--operation TEXT    One of (``+``, ``-``, ``=``). If you wish to change this
                    value, you must also provide the ``amount``, otherwise it
                    will not take effect.
--default TEXT      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``.  [default: no]
--in_users TEXT     A list of user IDs or user emails.
--in_groups TEXT    A list of group IDs or names.
-h, --help          Show this message and exit.
```

3.13 roles

This section is auto-generated from the help text for the parsec command `roles`.

3.13.1 `create_role` command

Usage:

```
parsec roles create_role [OPTIONS] ROLE_NAME DESCRIPTION
```

Help

Create a new role.

Output

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

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

Options:

```
--user_ids TEXT   A list of encoded user IDs to add to the new role
--group_ids TEXT  A list of encoded group IDs to add to the new role
-h, --help        Show this message and exit.
```

3.13.2 `get_roles` command

Usage:

```
parsec roles get_roles [OPTIONS]
```

Help

Displays a collection (list) of roles.

Output

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"}]
```

Options:

```
-h, --help  Show this message and exit.
```

3.13.3 `show_role` command

Usage:

```
parsec roles show_role [OPTIONS] ROLE_ID
```

Help

Display information on a single role

Output

A description of role For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.14 tool_data

This section is auto-generated from the help text for the parsec command `tool_data`.

3.14.1 `delete_data_table` command

Usage:

```
parsec tool_data delete_data_table [OPTIONS] DATA_TABLE_ID VALUES
```

Help

Delete an item from a data table.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.14.2 `get_data_tables` command

Usage:

```
parsec tool_data get_data_tables [OPTIONS]
```

Help

Get the list of all data tables.

Output

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

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

Options:

```
-h, --help  Show this message and exit.
```

3.14.3 `reload_data_table` command

Usage:

```
parsec tool_data reload_data_table [OPTIONS] DATA_TABLE_ID
```

Help

Reload a data table.

Output

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

```
{"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"}
```

Options:

```
-h, --help  Show this message and exit.
```

3.14.4 `show_data_table` command

Usage:

```
parsec tool_data show_data_table [OPTIONS] DATA_TABLE_ID
```

Help

Get details of a given data table.

Output

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

```
{"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"}
```

Options:

```
-h, --help  Show this message and exit.
```

3.15 tools

This section is auto-generated from the help text for the parsec command `tools`.

3.15.1 `get_tool_panel` command

Usage:

```
parsec tools get_tool_panel [OPTIONS]
```

Help

Get a list of available tool elements in Galaxy's configured toolbox.

Output

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

See also:

bioblend.galaxy.toolshed.get_repositories()

Options:

```
-h, --help  Show this message and exit.
```

3.15.2 `get_tools` command

Usage:

```
parsec tools get_tools [OPTIONS]
```

Help

Get all tools or filter the specific one(s) via the provided `name` or `tool_id`. Provide only one argument, `name` or `tool_id`, but not both.

Output

List of tool descriptions.

See also:

bioblend.galaxy.toolshed.get_repositories()

Options:

```
--tool_id TEXT  id of the requested tool
--name TEXT     name of the requested tool(s)
--trackster     if True, only tools that are compatible with Trackster are
                returned
-h, --help      Show this message and exit.
```

3.15.3 `install_dependencies` command

Usage:

```
parsec tools install_dependencies [OPTIONS] TOOL_ID
```

Help

Install dependencies for a given tool via a resolver. This works only for Conda currently. This functionality is available since Galaxy release_16.10 and is available only to Galaxy admins.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.15.4 `paste_content` command

Usage:

```
parsec tools paste_content [OPTIONS] CONTENT HISTORY_ID
```

Help

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

Output

Options:

```
-h, --help  Show this message and exit.
```

3.15.5 `put_url` command

Usage:

```
parsec tools put_url [OPTIONS] CONTENT HISTORY_ID
```

Help

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

Output

Options:

```
-h, --help  Show this message and exit.
```

3.15.6 `run_tool` command

Usage:

```
parsec tools run_tool [OPTIONS] HISTORY_ID TOOL_ID TOOL_INPUTS
```

Help

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

Output

Options:

```
-h, --help  Show this message and exit.
```

3.15.7 `show_tool` command

Usage:

```
parsec tools show_tool [OPTIONS] TOOL_ID
```

Help

Get details of a given tool.

Output

Options:

```
--io_details    if True, get also input and output details
--link_details  if True, get also link details
-h, --help      Show this message and exit.
```

3.15.8 `upload_file` command

Usage:

```
parsec tools upload_file [OPTIONS] PATH HISTORY_ID
```

Help

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

Output

Options:

```
--dbkey TEXT       (optional) genome dbkey
--file_name TEXT   (optional) name of the new history dataset
--file_type TEXT   Galaxy datatype for the new dataset, default is auto
--space_to_tab     whether to convert spaces to tabs. Default is False.
                   Applicable only if to_posix_lines is True
--to_posix_lines   if True, convert universal line endings to POSIX line
                   endings. Default is True. Set to False if you upload a gzip,
                   bz2 or zip archive containing a binary file
-h, --help         Show this message and exit.
```

3.15.9 `upload_from_ftp` command

Usage:

```
parsec tools upload_from_ftp [OPTIONS] PATH HISTORY_ID
```

Help

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

Output

Options:

```
-h, --help  Show this message and exit.
```

3.16 toolshed

This section is auto-generated from the help text for the parsec command `toolshed`.

3.16.1 `get_repositories` command

Usage:

```
parsec toolshed get_repositories [OPTIONS]
```

Help

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

Options:

```
-h, --help  Show this message and exit.
```

3.16.2 `install_repository_revision` command

Usage:

```
parsec toolshed install_repository_revision [OPTIONS] TOOL_SHED_URL NAME
```

Help

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.

Options:

```
--install_tool_dependencies     Whether or not to automatically handle tool
                                dependencies (see
                                https://galaxyproject.org/toolshed/tool-
                                dependency-recipes/ for more details)
--install_repository_dependencies
                                Whether or not to automatically handle
                                repository dependencies (see
                                https://galaxyproject.org/toolshed/defining-
                                repository-dependencies/ for more details)
--install_resolver_dependencies
                                Whether or not to automatically install
                                resolver dependencies (e.g. conda). This
                                parameter is silently ignored in Galaxy
                                ``release_16.04`` and earlier.
--tool_panel_section_id TEXT    The ID of the Galaxy tool panel section where
                                the tool should be insterted 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.
--new_tool_panel_section_label TEXT
                                The name of a Galaxy tool panel section that
                                should be created and the repository installed
                                into.
-h, --help                      Show this message and exit.
```

3.16.3 `show_repository` command

Usage:

```
parsec toolshed show_repository [OPTIONS] TOOLSHED_ID
```

Help

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

Options:

```
-h, --help  Show this message and exit.
```

3.17 toolshed_categories

This section is auto-generated from the help text for the parsec command `toolshed_categories`.

3.17.1 `get_categories` command

Usage:

```
parsec toolshed_categories get_categories [OPTIONS]
```

Help

Returns a list of dictionaries that contain descriptions of the repository categories found on the given Tool Shed instance.

Output

A list of dictionaries containing information about repository categories present in the Tool Shed. For example:

```
[{u'deleted': False,
  u'description': u'Tools for manipulating data',
  u'id': u'175812cd7caaf439',
  u'model_class': u'Category',
  u'name': u'Text Manipulation',
  u'url': u'/api/categories/175812cd7caaf439'}]
```

New in version 0.5.2.

Options:

```
--deleted   whether to show deleted categories
-h, --help  Show this message and exit.
```

3.17.2 `show_category` command

Usage:

```
parsec toolshed_categories show_category [OPTIONS] CATEGORY_ID
```

Help

Get details of a given category.

Output

details of the given category

Options:

```
-h, --help  Show this message and exit.
```

3.18 toolshed_repositories

This section is auto-generated from the help text for the parsec command `toolshed_repositories`.

3.18.1 `create_repository` command

Usage:

```
parsec [OPTIONS] COMMAND [ARGS]...
```

Help

Create a new repository in a Tool Shed.

3.18.2 `get_ordered_installable_revisions` command

Usage:

```
parsec [OPTIONS] COMMAND [ARGS]...
```

Help

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.

3.18.3 `get_repositories` command

Usage:

```
parsec [OPTIONS] COMMAND [ARGS]...
```

Help

Get a list of all the repositories in a Galaxy Tool Shed.

3.19 toolshed_tools

This section is auto-generated from the help text for the parsec command `toolshed_tools`.

3.19.1 `search_tools` command

Usage:

```
parsec toolshed_tools search_tools [OPTIONS] Q
```

Help

Search for tools in a Galaxy Tool Shed.

Output

dictionary containing search hits as well as metadata for the search. For example:

```
{u'hits': [{u'matched_terms': [],
   u'score': 3.0,
   u'tool': {u'description': u'convert between various FASTQ quality formats',
    u'id': u'69819b84d55f521efda001e0926e7233',
    u'name': u'FASTQ Groomer',
    u'repo_name': None,
    u'repo_owner_username': u'devteam'}},
  {u'matched_terms': [],
   u'score': 3.0,
   u'tool': {u'description': u'converts a bam file to fastq files.',
    u'id': u'521e282770fd94537daff87adad2551b',
    u'name': u'Defuse BamFastq',
    u'repo_name': None,
    u'repo_owner_username': u'jjohnson'}}],
 u'hostname': u'https://testtoolshed.g2.bx.psu.edu/',
 u'page': u'1',
 u'page_size': u'2',
 u'total_results': u'118'}
```

Options:

```
--page INTEGER       page requested  [default: 1]
--page_size INTEGER  page size requested  [default: 10]
-h, --help           Show this message and exit.
```

3.20 users

This section is auto-generated from the help text for the parsec command `users`.

3.20.1 `create_local_user` command

Usage:

```
parsec users create_local_user [OPTIONS] USERNAME USER_EMAIL PASSWORD
```

Help

Create a new Galaxy local user.

Output

a dictionary containing information about the created user

Options:

```
-h, --help  Show this message and exit.
```

3.20.2 `create_remote_user` command

Usage:

```
parsec users create_remote_user [OPTIONS] USER_EMAIL
```

Help

Create a new Galaxy remote user.

Output

a dictionary containing information about the created user

Options:

```
-h, --help  Show this message and exit.
```

3.20.3 `create_user` command

Usage:

```
parsec users [OPTIONS] COMMAND [ARGS]...
```

Help

Deprecated method.

3.20.4 `create_user_apikey` command

Usage:

```
parsec users create_user_apikey [OPTIONS] USER_ID
```

Help

Create a new API key for a given user.

Output

the API key for the user

Options:

```
-h, --help  Show this message and exit.
```

3.20.5 `delete_user` command

Usage:

```
parsec users delete_user [OPTIONS] USER_ID
```

Help

Delete a user.

Output

a dictionary containing information about the deleted user

Options:

```
--purge     if ``True``, also purge (permanently delete) the history
-h, --help  Show this message and exit.
```

3.20.6 `get_current_user` command

Usage:

```
parsec users get_current_user [OPTIONS]
```

Help

Display information about the user associated with this Galaxy connection.

Output

a dictionary containing information about the current user

Options:

```
-h, --help  Show this message and exit.
```

3.20.7 `get_user_apikey` command

Usage:

```
parsec users get_user_apikey [OPTIONS] USER_ID
```

Help

Get the current API key for a given user. This functionality is available since Galaxy `release_17.01`.

Output

the API key for the user

Options:

```
-h, --help  Show this message and exit.
```

3.20.8 `get_users` command

Usage:

```
parsec users get_users [OPTIONS]
```

Help

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

Output

a list of dicts with user details. For example:

```
[{u'email': u'a_user@example.com',
  u'id': u'dda47097d9189f15',
  u'url': u'/api/users/dda47097d9189f15'}]
```

Options:

```
--deleted TEXT
--f_email TEXT  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.ini`` configuration file. This parameter is
                silently ignored for non-admin users in Galaxy
                ``release_15.01`` and earlier.
--f_name TEXT   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.ini``
                configuration file. This parameter is silently ignored in
                Galaxy ``release_15.10`` and earlier.
--f_any TEXT    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.ini`` configuration file. This parameter is
                silently ignored in Galaxy ``release_15.10`` and earlier.
-h, --help      Show this message and exit.
```

3.20.9 `show_user` command

Usage:

```
parsec users show_user [OPTIONS] USER_ID
```

Help

Display information about a user.

Output

a dictionary containing information about the user

Options:

```
--deleted   whether to return results for a deleted user
-h, --help  Show this message and exit.
```

3.21 utils

This section is auto-generated from the help text for the parsec command `utils`.

3.21.1 `cmp` command

Usage:

```
parsec utils cmp [OPTIONS] METHOD CMP_WITH
```

Help

comparison tool. Exits if the value read from stdin does not pass the comparison test with the specified value.

method is the comparison method. One of these (lt, gt, eq, ne) which will trigger a numerical comparison (cast to floats), or one of teq, tneq which will trigger a string comparison.

cmp_with is the value to compare against. E.g. '100' or 'test'

e.g.:

```
echo '5' | parsec utils cmp lt 10 # exit 0

echo '5' | parsec utils cmp lt 1 # exit 1
```

Options:

```
-h, --help  Show this message and exit.
```

3.21.2 `library_recurse` command

Usage:

```
parsec utils library_recurse [OPTIONS] LIBRARY_ID
```

Help

Get all the folders or filter specific one(s) via the provided `name` or `folder_id` in data library with id `library_id`. Provide only one argument: `name` or `folder_id`, but not both.

Output

list of dicts each containing basic information about a folder

Options:

```
--path TEXT  Folder path to filter on (otherwise root of repo)
-h, --help   Show this message and exit.
```

3.21.3 `wait_on_invocation` command

Usage:

```
parsec utils wait_on_invocation [OPTIONS] WORKFLOW_ID INVOCATION_ID
```

Help

Given a workflow and invocation id, wait until that invocation is complete (or one or more steps have errored)

This will exit with the following error codes:

- 0: done successfully
- 1: running (if –exit_early)
- 2: failure
- 3: unknown

Options:

```
--exit_early          Exit immediately after checking status rather than
                      sleeping indefinitely
--backoff_min FLOAT   Minimum time to sleep between checks, in seconds.
--backoff_max FLOAT   Maximum time to sleep between checks, in seconds
-h, --help            Show this message and exit.
```

3.21.4 `xunit_xargs` command

Usage:

```
parsec utils xunit_xargs [OPTIONS] _
```

Help

xargs look-alike that wraps output calls as XUnit XML

e.g.:

```
parsec histories get_histories |             jq '.[].id' -r |             head -n 3 |␣
↪            parsec utils xunit_xargs parsec histories get_status \| jq .percent_
↪complete
```

will fetch the first three histories mentioned, and then pass them to xargs to run `parsec histories get_status [history_id] | jq .percent_complete`. This will in turn produce XUnit XML that can be used in Jenkins or similar systems:

```
<?xml version="1.0" ?>
<testsuites errors="0" failures="0" tests="3" time="1.5944418907165527">
        <testsuite errors="0" failures="0" name="Parsec XX" skipped="0" tests="3"␣
↪time="1.5944418907165527">
                <testcase classname="parsec.histories.get_status.769f01a3981796db_|.
↪jq..percent_complete" name="parsec.histories.get_status.769f01a3981796db_" time="0.
↪604831">
                        <system-out>100</system-out>
                </testcase>
                <testcase classname="parsec.histories.get_status.83fbc32772cb5fcf_|.
↪jq..percent_complete" name="parsec.histories.get_status.83fbc32772cb5fcf_" time="0.
↪483556">
                        <system-out>100</system-out>
                </testcase>
                <testcase classname="parsec.histories.get_status.90c9282cb8718062_|.
↪jq..percent_complete" name="parsec.histories.get_status.90c9282cb8718062_" time="0.
↪506056">
                        <system-out>97.82608695652173</system-out>
                </testcase>
        </testsuite>
</testsuites>
```

Options:

```
-h, --help  Show this message and exit.
```

3.22 visual

This section is auto-generated from the help text for the parsec command `visual`.

3.22.1 `get_visualizations` command

Usage:

```
parsec visual get_visualizations [OPTIONS]
```

Help

Get the list of all visualizations.

Output

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

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

Options:

```
-h, --help  Show this message and exit.
```

3.22.2 `show_visualization` command

Usage:

```
parsec visual show_visualization [OPTIONS] VISUAL_ID
```

Help

Get details of a given visualization.

Output

A description of the given visualization. For example:

```
{u'annotation': None,
 u'dbkey': u'mm9',
 u'id': u'18df9134ea75e49c',
 u'latest_revision': {  ... },
 u'model_class': u'Visualization',
 u'revisions': [u'aa90649bb3ec7dcb', u'20622bc6249c0c71'],
 u'slug': u'visualization-for-grant-1',
 u'title': u'Visualization For Grant',
 u'type': u'trackster',
 u'url': u'/u/azaron/v/visualization-for-grant-1',
 u'user_id': u'21e4aed91386ca8b'}
```

Options:

```
-h, --help  Show this message and exit.
```

3.23 workflows

This section is auto-generated from the help text for the parsec command `workflows`.

3.23.1 `cancel_invocation` command

Usage:

```
parsec workflows cancel_invocation [OPTIONS] WORKFLOW_ID INVOCATION_ID
```

Help

Cancel the scheduling of a workflow.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.23.2 `delete_workflow` command

Usage:

```
parsec workflows delete_workflow [OPTIONS] WORKFLOW_ID
```

Help

Delete a workflow identified by *workflow_id*.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.23.3 `export_workflow_dict` command

Usage:

```
parsec workflows export_workflow_dict [OPTIONS] WORKFLOW_ID
```

Help

Exports a workflow.

Output

Dictionary representing the requested workflow

Options:

```
-h, --help  Show this message and exit.
```

3.23.4 `export_workflow_json` command

Usage:

```
parsec workflows export_workflow_json [OPTIONS] WORKFLOW_ID
```

Help

Deprecated since version 0.9.0: Use `export_workflow_dict()` instead.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.23.5 `export_workflow_to_local_path` command

Usage:

```
parsec workflows export_workflow_to_local_path [OPTIONS] WORKFLOW_ID
```

Help

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

Output

Options:

```
--use_default_filename  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.
                        [default: True]
-h, --help              Show this message and exit.
```

3.23.6 `get_invocations` command

Usage:

```
parsec workflows get_invocations [OPTIONS] WORKFLOW_ID
```

Help

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

Output

A list of workflow invocations. For example:

```
[{u'history_id': u'2f94e8ae9edff68a',
  u'id': u'df7a1f0c02a5b08e',
  u'model_class': u'WorkflowInvocation',
  u'state': u'new',
  u'update_time': u'2015-10-31T22:00:22',
```

(continues on next page)

(continued from previous page)

```
  u'uuid': u'c8aa2b1c-801a-11e5-a9e5-8ca98228593c',
  u'workflow_id': u'03501d7626bd192f'}]
```

Options:

```
-h, --help  Show this message and exit.
```

3.23.7 `get_workflow_inputs` command

Usage:

```
parsec workflows get_workflow_inputs [OPTIONS] WORKFLOW_ID LABEL
```

Help

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

Output

list of workflow inputs matching the label query

Options:

```
-h, --help  Show this message and exit.
```

3.23.8 `get_workflows` command

Usage:

```
parsec workflows get_workflows [OPTIONS]
```

Help

Get all workflows or filter the specific one(s) via the provided `name` or `workflow_id`. Provide only one argument, `name` or `workflow_id`, but not both.

Output

A list of workflow dicts. For example:

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

Options:

```
--workflow_id TEXT  Encoded workflow ID (incompatible with ``name``)
--name TEXT         Filter by name of workflow (incompatible with
                    ``workflow_id``). If multiple names match the given name,
                    all the workflows matching the argument will be returned.
--published         if ``True``, return also published workflows
-h, --help          Show this message and exit.
```

3.23.9 `import_shared_workflow` command

Usage:

```
parsec workflows import_shared_workflow [OPTIONS] WORKFLOW_ID
```

Help

Imports a new workflow from the shared published workflows.

Output

A description of the workflow. For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.23.10 `import_workflow_dict` command

Usage:

```
parsec workflows import_workflow_dict [OPTIONS] WORKFLOW_DICT
```

Help

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

Output

Options:

```
--publish   if ``True`` the uploaded workflow will be published; otherwise it
            will be visible only by the user which uploads it (default)
-h, --help  Show this message and exit.
```

3.23.11 `import_workflow_from_local_path` command

Usage:

```
parsec workflows import_workflow_from_local_path [OPTIONS]
```

Help

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

Output

Options:

```
--publish   if ``True`` the uploaded workflow will be published; otherwise it
            will be visible only by the user which uploads it (default)
-h, --help  Show this message and exit.
```

3.23.12 `import_workflow_json` command

Usage:

```
parsec workflows import_workflow_json [OPTIONS] WORKFLOW_JSON
```

Help

Deprecated since version 0.9.0: Use `import_workflow_dict()` instead.

Output

Options:

```
-h, --help  Show this message and exit.
```

3.23.13 `invoke_workflow` command

Usage:

```
parsec workflows invoke_workflow [OPTIONS] WORKFLOW_ID
```

Help

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

Output

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

```
{u'history_id': u'2f94e8ae9edff68a',
 u'id': u'df7a1f0c02a5b08e',
 u'inputs': {u'0': {u'id': u'a7db2fac67043c7e',
   u'src': u'hda',
   u'uuid': u'7932ffe0-2340-4952-8857-dbaa50f1f46a'}},
 u'model_class': u'WorkflowInvocation',
 u'state': u'ready',
 u'steps': [{u'action': None,
   u'id': u'd413a19dec13d11e',
   u'job_id': None,
   u'model_class': u'WorkflowInvocationStep',
   u'order_index': 0,
   u'state': None,
   u'update_time': u'2015-10-31T22:00:26',
   u'workflow_step_id': u'cbbbf59e8f08c98c',
   u'workflow_step_label': None,
   u'workflow_step_uuid': u'b81250fd-3278-4e6a-b269-56a1f01ef485'}
↪,
  {u'action': None,
   u'id': u'2f94e8ae9edff68a',
```

(continues on next page)

(continued from previous page)

```
    u'job_id': u'e89067bb68bee7a0',
    u'model_class': u'WorkflowInvocationStep',
    u'order_index': 1,
    u'state': u'new',
    u'update_time': u'2015-10-31T22:00:26',
    u'workflow_step_id': u'964b37715ec9bd22',
    u'workflow_step_label': None,
    u'workflow_step_uuid': u'e62440b8-e911-408b-b124-e05435d3125e'}
↪],
 u'update_time': u'2015-10-31T22:00:26',
 u'uuid': u'c8aa2b1c-801a-11e5-a9e5-8ca98228593c',
 u'workflow_id': u'03501d7626bd192f'}
```

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.

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

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

then the following renames the output dataset to 'foo':

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

see also this email thread.

Warning: Historically, the `run_workflow` method 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.

Options:

```
--inputs TEXT                   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``).
--params TEXT                   A mapping of non-datasets tool parameters (see
                                below)
--history_id TEXT               The encoded history ID where to store the
                                workflow output. Alternatively,
                                ``history_name`` may be specified to create a
                                new history.
--history_name TEXT             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      If ``True``, used workflow inputs will be
                                imported into the history. If ``False``, only
                                workflow outputs will be visible in the given
                                history.
--replacement_params TEXT       pattern-based replacements for post-job
                                actions (see below)
--allow_tool_state_corrections  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.
-h, --help                      Show this message and exit.
```

3.23.14 `run_invocation_step_action` command

Usage:

```
parsec workflows run_invocation_step_action [OPTIONS] WORKFLOW_ID
```

Help

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).

Output

Options:

```
-h, --help  Show this message and exit.
```

3.23.15 `run_workflow` command

Usage:

```
parsec workflows run_workflow [OPTIONS] WORKFLOW_ID
```

Help

Run the workflow identified by `workflow_id`.

Output

A dict containing the history ID where the outputs are placed as well as output dataset IDs. For example:

```
{u'history': u'64177123325c9cfd',
 u'outputs': [u'aa4d3084af404259']}
```

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.

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

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

then the following renames the output dataset to 'foo':

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

see also this email thread.

Warning: This method waits for the whole workflow to be scheduled before returning and does not scale to large workflows as a result. This method has therefore been deprecated in favor of `invoke_workflow()`, which also features improved default behavior for dataset input handling.

Options:

```
--dataset_map TEXT              A mapping of workflow inputs to datasets. The
                                datasets source can be a
                                LibraryDatasetDatasetAssociation (``ldda``),
                                LibraryDataset (``ld``), or
```

(continues on next page)

(continued from previous page)

```
                              HistoryDatasetAssociation (``hda``). The map must
                              be in the following format: ``{'<input>': {'id':
                              <encoded dataset ID>, 'src': '[ldda, ld, hda]'}}``
                              (e.g. ``{'23': {'id': '29beef4fadeed09f', 'src':
                              'ld'}}``)
--params TEXT                 A mapping of non-datasets tool parameters (see
                              below)
--history_id TEXT             The encoded history ID where to store the workflow
                              output. Alternatively, ``history_name`` may be
                              specified to create a new history.
--history_name TEXT           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    If ``True``, used workflow inputs will be imported
                              into the history. If ``False``, only workflow
                              outputs will be visible in the given history.
--replacement_params TEXT     pattern-based replacements for post-job actions
                              (see below)
-h, --help                    Show this message and exit.
```

3.23.16 `show_invocation` command

Usage:

```
parsec workflows show_invocation [OPTIONS] WORKFLOW_ID INVOCATION_ID
```

Help

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.

Output

The workflow invocation. For example:


```
{u'history_id': u'2f94e8ae9edff68a',
 u'id': u'df7a1f0c02a5b08e',
 u'inputs': {u'0': {u'id': u'a7db2fac67043c7e',
   u'src': u'hda',
   u'uuid': u'7932ffe0-2340-4952-8857-dbaa50f1f46a'}},
 u'model_class': u'WorkflowInvocation',
 u'state': u'ready',
 u'steps': [{u'action': None,
   u'id': u'd413a19dec13d11e',
   u'job_id': None,
   u'model_class': u'WorkflowInvocationStep',
   u'order_index': 0,
   u'state': None,
   u'update_time': u'2015-10-31T22:00:26',
   u'workflow_step_id': u'cbbbf59e8f08c98c',
   u'workflow_step_label': None,
   u'workflow_step_uuid': u'b81250fd-3278-4e6a-b269-56a1f01ef485'},
  {u'action': None,
   u'id': u'2f94e8ae9edff68a',
```


(continues on next page)

(continued from previous page)

```
    u'job_id': u'e89067bb68bee7a0',
    u'model_class': u'WorkflowInvocationStep',
    u'order_index': 1,
    u'state': u'new',
    u'update_time': u'2015-10-31T22:00:26',
    u'workflow_step_id': u'964b37715ec9bd22',
    u'workflow_step_label': None,
    u'workflow_step_uuid': u'e62440b8-e911-408b-b124-e05435d3125e'}],
 u'update_time': u'2015-10-31T22:00:26',
 u'uuid': u'c8aa2b1c-801a-11e5-a9e5-8ca98228593c',
 u'workflow_id': u'03501d7626bd192f'}
```

Options:

```
-h, --help  Show this message and exit.
```

3.23.17 `show_invocation_step` command

Usage:

```
parsec workflows show_invocation_step [OPTIONS] WORKFLOW_ID INVOCATION_ID
```

Help

See the details of a particular workflow invocation step.

Output

The workflow invocation step. For example:

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

Options:

```
-h, --help  Show this message and exit.
```

3.23.18 `show_workflow` command

Usage:

```
parsec workflows show_workflow [OPTIONS] WORKFLOW_ID
```

Help

Display information needed to run a workflow.

Output

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

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

Options:

```
-h, --help  Show this message and exit.
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search