pyclarity-lims Documentation

Edinburgh Genomics

Aug 06, 2020

Table of Contents:

Pyclarity-lims is a fork of genologics that we have extended and modified. Most of the initial logic still applies and the *genologics* module still exists as an alias for backward compatibility. However there are a few backward incompatible changes that have had to be made.

CHAPTER 1

Installation

```
pip install pyclarity-lims
```

CHAPTER 2

Getting started

pyclarity-lims is a module that will help you access your Basespace-clarity REST API by parsing the xml the API returns into Python objects.

2.1 Lims connection

To create a Lims connection you'll need to create a *Lims* object.

```
from pyclarity_lims.lims import Lims

l = Lims('https://claritylims.example.com', 'username' , 'Pa55w0rd')
```

The *Lims* instance is the main object that will interact with the REST API and manage all communications. There are two way of accessing information stored in the LIMS:

2.2 Searching the Lims

The most common way of accessing data from the LIMS is to first perform searches. For example, retrieving all samples from project1 would be:

```
samples = l.get_samples(projectname='project1')
```

This will return a list of all *Sample* objects that belong to project1.

The functions from pyclarity_lims closely match the API search function from Basespace-clarity REST API. For example *get_samples* has similar parameters as the samples end point from Basespace-clarity

2.3 Retrieving object with their id

In some cases you will know the id or uri of the instance you want to retrieve. In this case you can create the object directly.

Note that you will still need the *Lims* instance as an argument.

For Example:

```
from pyclarity_lims.entities import Sample
sample = Sample(l, id='sample_luid')
print(sample.name)
```

2.4 Lazy loading and caching

All entities such as *Sample*, *Artifact* or *Step* are loaded lazily which mean that no query will be sent to the REST API until an attribute of the object is accessed or a method is run. In the code above:

```
from pyclarity_lims.entities import Sample
sample = Sample(l, id='sample_luid')
# the Sample object has been created but no query have been sent yet
print(sample.name)
# accessing the name of the sample triggers the query
```

To avoid sending too many queries, all Entities that have been retrieved are also cached which means that once the Entity is retrieved it won't be queried again unless forced. This makes pyclarity_lims more efficient but also not very well suited for long running process during which the state of the LIMS is likely to change. You can bypass the cache as shown in *Make sure to have the up-to-date program status*.

2.5 Looking beyond

You can look at some *Practical Examples* There are many other search methods available in the *Lims* and you can also look at all the classes defined in *Entities*

CHAPTER 3

Practical Examples

3.1 Change value of a UDF of all artifacts of a Step in progress

The goal of this example is to show how you can change the value of a UDF named udfname in all input artifacts. This example assumes you have a *Lims* and a process id.

```
# Create a process entity from an existing process in the LIMS
p = Process(l, id=process_id)
# Retreive  each input artifacts and iterate over them
for artifact in p.all_inputs():
    # change the value of the udf
    artifact.udf['udfname'] = 'udfvalue'
    # upload the artifact back to the Lims
    artifact.put()
```

In some cases we want to optimise the number of queries sent to the LIMS and make use of the batched query the API offers.

```
p = Process(l, id=process_id)
# This time we create all the Artifact entities and use the batch query to retrieve␣
↪the content
# then iterate over them
for artifact in p.all_inputs(resolve=True):
    artifact.udf['udfname'] = 'udfvalue'
# Upload all the artifacts in one batch query
l.batch_put(p.all_inputs())
```

Note: A batch query is usually faster than the equivalent number of individual queries. However the gain seems very variable and is not as high as one might expect.

3.2 Find all the samples that went through a Step with a specific udf value

This is a typical search that is performed when searching for sample that went through a specific sequencing run.

```
# there should only be one such process
processes = l.get_processes(type='Sequencing', udf={'RunId': run_id})
samples = set()
for a in processes[0].all_inputs(resolve=True):
    samples.update(a.samples)
```

3.3 Make sure to have the up-to-date program status

Because all the entities are cached, sometime the Entities get out of date especially when the data in the LIMS is changing rapidly, like the status of a running program.

```
s = Step(l, id=step_id)
s.program_status.status  # returns RUNNING
sleep(10)
s.program_status.status  # returns RUNNING because it is still cached
s.program_status.get(force=True)
s.program_status.status  # returns COMPLETE
```

The function *get* is most of the time used implicitly but can be used explicitly with the force option to bypass the cache and retrieve an up-to-date version of the instance.

3.4 Create sample with a Specific udfs

So far we have only retrieved entities from the LIMS and in some case modified them before uploading them back. We can also create some of these entities and upload them to the LIMS. Here is how to create a sample with a specific udf.

```
Sample.create(l, container=c, position='H:3', project=p, name='sampletest', udf={
↪'testudf':'testudf_value'})
```

3.5 Start and complete a new Step from submitted samples

Creating a step, filling in the placements and the next actions, then completing the step can be very convenient when you want to automate the execution of part of your workflow. Here is an example with one sample placed into a tube.

```
# Retrieve samples/artifact/workflow stage
samples = l.get_samples(projectname='project1')
art = samples[0].artifact
# Find workflow 'workflowname' and take its first stage
stage = l.get_workflows(name='workflowname')[0].stages[0]

# Queue the artifacts to the stage
l.route_artifacts([art], stage_uri=stage.uri)
```

(continues on next page)

(continued from previous page)

```
# Create a new step from that queued artifact
s = Step.create(l, protocol_step=stage.step, inputs=[art], container_type_name='Tube')

# Create the output container
ct = l.get_container_types(name='Tube')[0]
c = Container.create(l, type=ct)

# Retrieve the output artifact that was generated automatically from the input/output␣
↪map
output_art = s.details.input_output_maps[0][1]['uri']

# Place the output artifact in the container
# Note that the placements is a list of tuples ( A, ( B, C ) ), where A is the output␣
↪artifact,
# B is the output Container and C is the location on this container
output_placement_list=[(output_art, (c, '1:1'))]
# set_placements creates the placement entity and "put"s it
s.set_placements([c], output_placement_list)

# Move from "Record detail" window to the "Next Step"
s.advance()

# Set the next step
actions = s.actions.next_actions[0]['action'] = 'complete'
s.actions.put()

# Complete the step
s.advance()
```

3.6 Mix samples in a pool using the api

Some step will allow you to mix multiple input *artifacts* into a pool also represented by an *artifact*. This can be performed using the *StepPools* entities.

Because the pool *artifact* needs to be created in the LIMS, we only need to provide the pool name and we need to provide *None* in place of the pool

```
# Assuming a Step in the pooling stage
s = Step(l, id='122-12345')
# This provides a list of all the artifacts available to pool
s.pools.available_inputs
# The pooled_inputs is a dict where the key is the name of the pool
# the value is a Tuple with first element is the pool artifact and the second if the␣
↪pooled input
# here we're not specifying the pool and will let the LIMS create it.
s.pools.pooled_inputs['Pool1'] = (None, tuple(s.pools.available_inputs))
# then upload
s.pools.put()
# There no more input artifacts available
assert s.pools.available_inputs == []
# There is a pool artifact created
assert type(s.pools.pooled_inputs['Pool1'][0]).__name__ == 'Artifact'

# Now we can advance the step
s.advance()
```

3.7 Creating large number of Samples with create_batch

We have already seen that you can create sample in *Create sample with a Specific udfs*. But when you need to create a large number of samples, this method can be quite slow. The function *create_batch* can create multiple samples (or containers) in a single query. You'll need to specify the Entity you wish to create and the parameters you would have passed to the create method as one dictionary for each entity to create. The function returns the list of created entity in the same order as the list of dictionary provided.

```
# Assuming the Container c and the Project p exists.
samples = l.create_batch(
    Sample,
    [
        {'container': c, 'project': p, 'name': 'sampletest1', 'position': 'H:1', 'udf
↪':{'testudf': 'testudf_value1'}},
        {'container': c, 'project': p, 'name': 'sampletest2', 'position': 'H:2', 'udf
↪':{'testudf': 'testudf_value2'}},
        {'container': c, 'project': p, 'name': 'sampletest3', 'position': 'H:3', 'udf
↪':{'testudf': 'testudf_value3'}},
        {'container': c, 'project': p, 'name': 'sampletest4', 'position': 'H:4', 'udf
↪':{'testudf': 'testudf_value4'}},
        {'container': c, 'project': p, 'name': 'sampletest5', 'position': 'H:5', 'udf
↪':{'testudf': 'testudf_value5'}}
    ]
)
```

Warning: The create_batch function returns entities already created with all attributes specified during the creation populated. However it does not include attributes created on the LIMS side such as the artifact of samples. These have to be retrieved manually using *sample.get(force=True)* or *lims.get_batch(samples, force=True)*

```
# After creation of the samples above
samples[0].artifact           # returns None
samples[0].get(force=True)    # retrieve the attribute as they are on the LIMS
samples[0].artifact           # returns Artifact(uri=...)
```

CHAPTER 4

Lims object

class pyclarity_lims.lims.**Lims**(*baseuri*, *username*, *password*, *version='v2'*)

Bases: object

LIMS interface through which all searches can be performed and *Entity* instances are retrieved.

Parameters

- **baseuri** – Base URI for the GenoLogics server, excluding the 'api' or version parts!
- **username** – The account name of the user to login as.
- **password** – The password for the user account to login as.
- **version** – The optional LIMS API version, by default 'v2'

Example:

```
Lims('https://claritylims.example.com', 'username' , 'Pa55w0rd')
```

VERSION = 'v2'

check_version()

Raise ValueError if the version for this interface does not match any of the versions given for the API.

create_batch(*klass*, *list_kwargs*)

Create using the batch create endpoint. It is only available for Sample and Container entities.

Parameters

- **klass** – The class to use when creating the entity (*Sample* or *Container*)
- **list_kwargs** – A list of dictionary where each dictionary will be used to create a instance of the klass. Elements of the dictionary should match the keyword argument in the create method of *Sample* or *Container*

Returns A list of the created entities in the same order as the list of kwargs.

get(*uri*, *params={}*)

GET data from the URI. It checks the status and return the text of response as an ElementTree.

Parameters

- **uri** – the uri to query
- **params** – dict containing the query parameters

Returns the text of the response as an ElementTree

get_artifacts(*name=None*, *type=None*, *process_type=None*, *artifact_flag_name=None*, *working_flag=None*, *qc_flag=None*, *sample_name=None*, *samplelimsid=None*, *artifactgroup=None*, *containername=None*, *containerlimsid=None*, *reagent_label=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*, *resolve=False*)

Get a list of artifacts, filtered by keyword arguments.

Parameters

- **name** – Artifact name, or list of names.
- **type** – Artifact type, or list of types.
- **process_type** – Produced by the process type, or list of types.
- **artifact_flag_name** – Tagged with the genealogy flag, or list of flags.
- **working_flag** – Having the given working flag; boolean.
- **qc_flag** – Having the given QC flag: UNKNOWN, PASSED, FAILED.
- **sample_name** – Related to the given sample name.
- **samplelimsid** – Related to the given sample id.
- **artifactgroup** – Belonging to the artifact group (experiment in client).
- **containername** – Residing in given container, by name, or list.
- **containerlimsid** – Residing in given container, by LIMS id, or list.
- **reagent_label** – having attached reagent labels.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **resolve** – Send a batch query to the lims to get the content of all artifacts retrieved

get_batch(*instances*, *force=False*)

Get the content of a set of instances using the efficient batch call.

Returns the list of requested instances in arbitrary order, with duplicates removed (duplicates=entities occurring more than once in the instances argument).

For Artifacts it is possible to have multiple instances with the same LIMSID but different URI, differing by a query parameter ?state=XX. If state is not given for an input URI, a state is added in the data returned by the batch API. In this case, the URI of the Entity object is not updated by this function (this is similar to how Entity.get() works). This may help with caching.

The batch request API call collapses all requested Artifacts with different state into a single result with state equal to the state of the Artifact occurring at the last position in the list.

Parameters

- **instances** – List of instances children of Entity
- **force** – optional argument to force the download of already cached instances

get_container_types(*name=None*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of container types, filtered by keyword arguments.

Parameters

- **name** – name of the container type or list of names.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_containers(*name=None*, *type=None*, *state=None*, *last_modified=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of containers, filtered by keyword arguments.

Parameters

- **name** – Containers name, or list of names.
- **type** – Container type, or list of types.
- **state** – Container state: Empty, Populated, Discarded, Reagent-Only.
- **last_modified** – Since the given ISO format datetime.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_file_contents(*id=None*, *uri=None*, *encoding=None*, *crlf=False*, *binary=False*)
Download and returns the contents of the file of <ID> or <uri>.

Parameters

- **id** – the id of the file to retrieve.
- **uri** – the uri of the file to retrieve.
- **encoding** – When retrieve text file, this option can specify the encoding of the file.
- **crlf** – When set to True the text file will be replace \r\n by \n.
- **binary** – When set to True the file content is returned as a binary stream.

Returns The file content in the format specify by the parameters.

get_labs(*name=None*, *last_modified=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of labs, filtered by keyword arguments.

Parameters

- **name** – Lab name, or list of names.
- **last_modified** – Since the given ISO format datetime.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_process_types(*displayname=None*, *add_info=False*)
Get a list of process types with the specified name.

Parameters

- **displayname** – The name the process type
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_processes(*last_modified=None*, *type=None*, *inputartifactlimsid=None*, *techfirstname=None*, *techlastname=None*, *projectname=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*)
Get a list of processes, filtered by keyword arguments.

Parameters

- **last_modified** – Since the given ISO format datetime.
- **type** – Process type, or list of types.
- **inputartifactlimsid** – Input artifact LIMS id, or list of.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **techfirstname** – First name of researcher, or list of.
- **techlastname** – Last name of researcher, or list of.
- **projectname** – Name of project, or list of.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.

get_projects(*name=None*, *open_date=None*, *last_modified=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of projects, filtered by keyword arguments.

Parameters

- **name** – Project name, or list of names.
- **open_date** – Since the given ISO format date.
- **last_modified** – Since the given ISO format datetime.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_protocols(*name=None*, *add_info=False*)
Get a list of existing protocols on the system.

Parameters

- **name** – The name the protocol
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_reagent_kits(*name=None*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of reagent kits, filtered by keyword arguments.

Parameters

- **name** – reagent kit name, or list of names.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_reagent_lots(*name=None*, *kitname=None*, *number=None*, *start_index=None*, *nb_pages=-1*)
Get a list of reagent lots, filtered by keyword arguments.

Parameters

- **name** – reagent kit name, or list of names.
- **kitname** – name of the kit this lots belong to
- **number** – lot number or list of lot number
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.

get_reagent_types(*name=None*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of reagent types, filtered by keyword arguments.

Parameters

- **name** – Reagent type name, or list of names.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_researchers(*firstname=None*, *lastname=None*, *username=None*, *last_modified=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of researchers, filtered by keyword arguments.

Parameters

- **firstname** – Researcher first name, or list of names.
- **lastname** – Researcher last name, or list of names.
- **username** – Researcher account name, or list of names.
- **last_modified** – Since the given ISO format datetime.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_sample_number(*name=None*, *projectname=None*, *projectlimsid=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*)
Gets the number of samples matching the query without fetching every sample, so it should be faster than len(get_samples())

get_samples(*name=None*, *projectname=None*, *projectlimsid=None*, *udf={}*, *udtname=None*, *udt={}*, *start_index=None*, *nb_pages=-1*)
Get a list of samples, filtered by keyword arguments.

Parameters

- **name** – Sample name, or list of names.
- **projectlimsid** – Samples for the project of the given LIMS id.
- **projectname** – Samples for the project of the name.
- **udf** – dictionary of UDFs with 'UDFNAME[OPERATOR]' as keys.
- **udtname** – UDT name, or list of names.
- **udt** – dictionary of UDT UDFs with 'UDTNAME.UDFNAME[OPERATOR]' as keys and a string or list of strings as value.

- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.

get_udfs(*name=None*, *attach_to_name=None*, *attach_to_category=None*, *start_index=None*, *nb_pages=-1*, *add_info=False*)
Get a list of udfs, filtered by keyword arguments.

Parameters

- **name** – name of udf
- **attach_to_name** – item in the system, to wich the udf is attached, such as Sample, Project, Container, or the name of a process.
- **attach_to_category** – If 'attach_to_name' is the name of a process, such as 'CaliperGX QC (DNA)', then you need to set attach_to_category='ProcessType'. Must not be provided otherwise.
- **start_index** – first element to retrieve; start at first element if None.
- **nb_pages** – number of page to iterate over. The page size is 500 by default unless configured otherwise in your LIMS. 0 or negative numbers returns all pages.
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

get_uri(**segments*, ***query*)
Return the full URI given the path segments and optional query.

Parameters

- **segments** – arguments creating the uri
- **query** – kwargs creating the query

get_workflows(*name=None*, *add_info=False*)
Get a list of existing workflows on the system.

Parameters

- **name** – The name of the workflow you're looking for
- **add_info** – Change the return type to a tuple where the first element is normal return and the second is a dict of additional information provided in the query.

parse_response(*response, accept_status_codes=[200]*)
Parse the XML returned in the response. Raise an HTTP error if the response status is not 200.

post(*uri*, *data*, *params={}*)
POST the serialized XML to the given URI. Return the response XML as an ElementTree.

put(*uri*, *data*, *params={}*)
PUT the serialized XML to the given URI. Return the response XML as an ElementTree.

put_batch(*instances*)
Update multiple instances using a single batch request.

Parameters **instances** – List of instances children of Entity

route_artifacts(*artifact_list*, *workflow_uri=None*, *stage_uri=None*, *unassign=False*)
Take a list of artifacts and queue them to the stage specified by the stage uri. If a workflow uri is specified, the artifacts will be queued to the first stage of the workflow.

Parameters

- **artifact_list** – list of Artifacts.
- **workflow_uri** – The uri of the workflow.
- **stage_uri** – The uri of the stage.
- **unassign** – If True, then the artifact will be removed from the queue instead of added.

tostring(*etree*)
Return the ElementTree contents as a UTF-8 encoded XML string.

upload_new_file(*entity*, *file_to_upload*)
Upload a file and attach it to the provided entity.

validate_response(*response, accept_status_codes=[200]*)
Parse the XML returned in the response. Raise an HTTP error if the response status is not one of the specified accepted status codes.

write(*outfile*, *etree*)
Write the ElementTree contents as UTF-8 encoded XML to the open file.

CHAPTER 5

Entities

class pyclarity_lims.entities.**Artifact**(*lims*, *uri=None*, *id=None*, *_create_new=False*)

Bases: *pyclarity_lims.entities.Entity*

Any process input or output; analyte or file.

concentration

–

container

The container where the artifact is located, or None

files

List of *files* associated with the artifact.

get_state()

Parse out the state value from the URI.

input_artifact_list()

Returns the input artifact ids of the parent process.

location

The Artifact's location in a container.

name

The name of the artifact.

output_type

The output-type of the Artifact

parent_process

The *parent process* that generated this artfact.

qc_flag

The qc-flag applied to the Artifact.

reagent_labels

List of *Reagent labels* associated with the artifact.

samples
List of *Samples* associated with this artifact.

state
Parse out the state value from the URI.

stateless
Return the artifact independently of its state

type
The type of the artifact: Analyte, ResultFile or SharedResultFile.

udf
Dictionary of UDFs associated with the artifact.

volume
–

workflow_stages
List of workflow stage *Steps* that this artifact ran through.

workflow_stages_and_statuses
List of tuples containing three elements (A, B, C) where:

- A is a *Step* this artifact has run through.
- B is the status of said Step.
- C the name of the Step.

working_flag
The working-flag of the Artifact.

class pyclarity_lims.entities.**Container**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Container for analyte artifacts.

get_placements()
Get the dictionary of locations and artifacts using the more efficient batch call.

name
Name of the container

occupied_wells
Number of wells occupied in the container.

placements
Dictionary of placements in a Container. The key is the location such as "A:1" and the value is the artifact in that well/tube.

state
State of the container. e.g. Populated

type
Type of the container.

udf
Dictionary of UDFs associated with the container.

udt
Dictionary of UDTs associated with the container.

class pyclarity_lims.entities.**Containertype**(*lims*, *uri=None*, *id=None*, *_create_new=False*)

Bases: *pyclarity_lims.entities.Entity*

Type of container for analyte artifacts.

calibrant_wells
If there are any wells on this container that are use for calibration. They would be defined here.

name
Name of the type of container (Tube, 96 well plates, ...)

unavailable_wells
If there are any well on this container that should not be used. They would be defined here.

x_dimension
Number of position on the x axis

y_dimension
Number of position on the y axis

class pyclarity_lims.entities.**Entity**(*lims*, *uri=None*, *id=None*, *_create_new=False*)

Bases: object

Base abstract class for every entity in the LIMS database. An Entity corresponds to an XML document and as such it should have at least a uri or an id.

classmethod create(*lims*, ***kwargs*)
Create an instance from attributes then post it to the LIMS

get(*force=False*)
Get the XML data for this instance.

id
Return the LIMS id; obtained from the URI.

post()
Save this instance with POST

put()
Save this instance by doing PUT of its serialized XML.

uri

class pyclarity_lims.entities.**File**(*lims*, *uri=None*, *id=None*, *_create_new=False*)

Bases: *pyclarity_lims.entities.Entity*

File attached to a project or a sample.

attached_to
The uri of the Entity this file is attached to

content_location
The location of the file on the server

is_published
Whether the file is published or not

original_location
The original location of the file when it was uploaded

class pyclarity_lims.entities.**Lab**(*lims*, *uri=None*, *id=None*, *_create_new=False*)

Bases: *pyclarity_lims.entities.Entity*

A lab is a list of researchers.

billing_address
Billing address of the lab

externalids
List of external identifiers associated with the lab

name
Name of the lab

shipping_address
Shipping address of the lab

udf
Dictionary of UDFs associated with the Lab

udt
Dictionary of UDTs associated with the Lab

website
URL to the lab website

class pyclarity_lims.entities.**Note**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Note attached to a project or a sample.

content
The content of the note

class pyclarity_lims.entities.**Process**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Process (instance of Processtype) executed producing ouputs from inputs.

all_inputs(*unique=True*, *resolve=False*)
Retrieving all input artifacts from input_output_maps. If unique is true, no duplicates are returned.

Parameters

- **unique** – boolean specifying if the list of artifacts should be uniqued
- **resolve** – boolean specifying if the artifacts entities should be resolved through a batch query.

Returns list of input artifacts.

all_outputs(*unique=True*, *resolve=False*)
Retrieving all output artifacts from input_output_maps. If unique is true, no duplicates are returned.

Parameters

- **unique** – boolean specifying if the list of artifacts should be uniqued
- **resolve** – boolean specifying if the artifact entities should be resolved through a batch query.

Returns list of output artifacts.

analytes()
Retrieving the output Analytes of the process, if existing. If the process is not producing any output analytes, the input analytes are returned. Input/Output is returned as an information string. Makes aggregate processes and normal processes look the same.

date_run
The date at which the process was finished in format Year-Month-Day i.e. 2016-12-05.

files
List of *files* associated with the sample.

input_output_maps
List of tuples (input, output) where input and output item are dictionaries representing the input/output. Keys of the dict can be:

- for the input:
  - post-process-uri: input *Artifact*
  - uri: input *Artifact*
  - limsid: lims id of the input artifact
  - parent-process: *Process* that generated this input
- for the output:
  - uri: output *Artifact*
  - limsid: id of the Artifact generated
  - output-generation-type: type of output generation (example: PerInput)
  - output-type: type of artifact generated (Analyte, or ResultFile)

input_per_sample(*sample*)
Getting all the input artifacts derived from the specified sample

Parameters **sample** – the sample name to check against

Returns list of input artifacts matching the sample name

output_containers()
Retrieve all unique output containers

outputs_per_input(*inart*, *ResultFile=False*, *SharedResultFile=False*, *Analyte=False*)
Getting all the output artifacts related to a particular input artifact

Parameters

- **inart** – input artifact id or artifact entity use to select the output
- **ResultFile** – boolean specifying to only return ResultFiles.
- **SharedResultFile** – boolean specifying to only return SharedResultFiles.
- **Analyte** – boolean specifying to only return Analytes.

Returns output artifact corresponding to the input artifact provided

parent_processes()
Retrieving all parent processes through the input artifacts

process_parameter
Parameter for the process

protocol_name
The name of the protocol

result_files(*output_generation_type=None*)
Retrieve all output artifacts where output-type is ResultFile.

Parameters **output_generation_type** – string specifying the output-generation-type (PerAllInputs or PerInput)

Returns list of output artifacts.

shared_result_files(*output_generation_type=None*)
Retrieve all output artifacts where output-type is SharedResultFile.

Parameters **output_generation_type** – string specifying the output-generation-type (PerAllInputs or PerInput)

Returns list of output artifacts.

step
Retrieve the Step corresponding to this process. They share the same id

technician
The *researcher* that started the step.

type
The *type* of the process

udf
Dictionary of UDFs associated with the process.

Note that the UDFs cannot be modify in Process. Use *Step details* to modify UDFs instead. You can access them with process.step.details.udf

udt
Dictionary of UDTs associated with the process.

class pyclarity_lims.entities.**Processtype**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

name
Name of the process type.

class pyclarity_lims.entities.**Project**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Project concerning a number of samples; associated with a researcher.

close_date
The date at which the project was closed in format Year-Month-Day i.e. 2016-12-05.

externalids
List of external identifiers associated with the project

files
List of files attached to the project

invoice_date
The date at which the project was invoiced in format Year-Month-Day i.e. 2016-12-05.

name
The name of the project.

open_date
The date at which the project was opened in format Year-Month-Day i.e. 2016-12-05.

researcher
The researcher associated with the project.

udf
Dictionary of UDFs associated with the project

udt
Dictionary of UDTs associated with the project

class pyclarity_lims.entities.**Protocol**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Protocol, holding ProtocolSteps and protocol-properties

properties
List of dicts describing the protocol's property.

steps
List of *steps*

class pyclarity_lims.entities.**ProtocolStep**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Steps key in the Protocol object

epp_triggers
List of dicts describing the EPP trigger attached to this step.

name
Name of the step

permitted_containers
List of names for the permitted container type in that step.

queue
The queue associated with this protocol step. The link is possible because they share the same id.

queue_fields
List of dicts describing the fields available in that step's queue.

sample_fields
List of dicts describing the field available in that step's sample view.

step_fields
List of dicts describing the fields available in that step's UDF.

step_properties
List of dicts describing the properties of this step.

type
Processtype associated with this step.

class pyclarity_lims.entities.**Queue**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Queue of a given workflow stage

artifacts
List of *artifacts* associated with this workflow stage.

queued_artifacts
List of *artifacts* associated with this workflow stage alongside the time they've been added to that queue and the container they're in. The list contains tuples organised in the form (A, B, (C, D)), where:

- A is an *artifact*
- B is a `datetime` object,
- C is a *container*
- D is a string specifying the location such as "1:1"

class pyclarity_lims.entities.**ReagentKit**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Type of Reagent with information about the provider

archived
Whether the reagent kit is archived or not

name
Name of the reagent kit

supplier
Supplier for the reagent kit

website
Website associated with the reagent kit

class pyclarity_lims.entities.**ReagentLot**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Information about a particular regaent lot used in a step

created_by
Researcher that created that lot.

created_date
The date at which the lot was created in format Year-Month-Day i.e. 2016-12-05.

expiry_date
The date at which the lot expires in format Year-Month-Day i.e. 2016-12-05.

last_modified_by
Researcher that last modified this lot.

last_modified_date
The date at which the lot was last modified in format Year-Month-Day i.e. 2016-12-05.

lot_number
Lot number

name
Name of the reagent lot

reagent_kit
Reagent kit associated with this lot.

status
Status of the lot.

usage_count
Number of times the lot was used.

class pyclarity_lims.entities.**ReagentType**(*lims*, *uri=None*, *id=None*)
Bases: *pyclarity_lims.entities.Entity*

Reagent Type, usually indexes for sequencing

category
Reagent category associated with the type

class pyclarity_lims.entities.**Reagent_label**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Reagent label element

reagent_label
The reagent label

class pyclarity_lims.entities.**Researcher**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Person; client scientist or lab personnel. Associated with a lab.

email
Email of the researcher

externalids
List of external identifiers associated with the researcher

fax
Fax number of the researcher

first_name
First name of the researcher

initials
Initials of the researcher

lab
Lab associated with the researcher

last_name
Last name of the researcher

name
Complete name of the researcher

phone
Phone number of the researcher

udf
Dictionary of UDFs associated with the researcher

udt
Dictionary of UDTs associated with the researcher

class pyclarity_lims.entities.**Sample**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Customer's sample to be analyzed; associated with a project.

artifact
Initial *Artifact* associated with the sample.

classmethod create(*lims*, *container*, *position*, ***kwargs*)
Create an instance of Sample from attributes then post it to the LIMS

date_completed
The date at which the sample was completed in format Year-Month-Day i.e. 2016-12-05.

date_received
The date at which the sample was received in format Year-Month-Day i.e. 2016-12-05.

externalids
List of external identifiers associated with the sample

files
List of files associated with the sample.

name
Name of the sample.

notes
List of notes associated with the sample.

project
The project associated with that sample.

submitter
The researcher who submitted this sample.

udf
Dictionary of UDFs associated with the sample.

udt
Dictionary of UDTs associated with the sample.

class pyclarity_lims.entities.**Stage**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Holds Protocol/Workflow

index
Position of the stage in the protocol.

name
Name of the stage.

protocol
Protocol associated with this stage.

step
Step associated with this stage.

workflow
Workflow associated with the stage.

class pyclarity_lims.entities.**Step**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Step, as defined by the genologics API.

actions
Link to the *StepActions* entity

advance()
Send a post query to advance the step to the next step

available_programs
List of available programs to trigger. Each element is a tuple with the name and the trigger uri

configuration
Step configuration associated with the step.

classmethod create(*lims*, *protocol_step*, *inputs*, *container_type_name=None*, *reagent_category=None*, *replicates=1*, ***kwargs*)
Create a new instance of a Step. This method will start a step from queued artifacts.

Parameters

- **lims** – Lims connection object
- **protocol_step** – the *ProtocolStep* specifying the step to start.

- **inputs** – A list of *artifacts* as input to the step. These need to be queued for that step for the query to be successful.
- **container_type_name** – optional name of the type of container that this step use for its output. if omitted it uses the required type from the ProtocolStep if there is only one.
- **reagent_category** – optional reagent_category.
- **replicates** – int or list of ints specifying the number of replicates for each inputs.

current_state
The current state of the step.

date_completed
The date at which the step completed in format Year-Month-DayTHour:Min:Sec, e.g. 2016-11-22T10:43:32.857+00:00

date_started
The date at which the step started in format Year-Month-DayTHour:Min:Sec, e.g. 2016-11-22T10:43:32.857+00:00

details
Link to the *StepDetails* entity

placements
Link to the *StepPlacements* entity

pools
Link to the *StepPools* entity

process
Retrieve the Process corresponding to this Step. They share the same id

program_names
List of available program names.

program_status
Link to the *StepProgramStatus* entity

reagent_lots
List of reagent lots

set_placements(*output_containers*, *output_placement_list*)
Create a new placement for a new step. This method also modifies the selected containers with the provided output container. It is meant to be used with a newly created step that does not have a placement yet.

Parameters

- **output_containers** – List of *Containers* used to store the output artifacts.
- **output_placement_list** – List of tuples (A, (B, C)) where:
  - A is an *artifact*,
  - B is a *container*,
  - C is a string specifying the location in the container such as "1:1"

trigger_program(*name*)
Trigger a program of the provided name.

Parameters **name** – the name of the program.

Returns the program status.

Raises **ValueError** – if the program does not exist.

class pyclarity_lims.entities.**StepActions**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Actions associated with the end of the step

escalation

next_actions

List of dicts that represent an action for an artifact. They keys of the dict are:

- artifact: The *artifact* associated with this Action
- step: The next *step* associated with this action
- rework-step: The *step* associated with this action when the Artifact needs to be requeued
- **action: The type of action to perform.**
  - leave: Leave the sample in the QC protocol.
  - repeat: Repeat this step.
  - remove: Remove from workflow.
  - review: Request manager review.
  - complete: Mark protocol as complete.
  - store: Store for later.
  - nextstep: Continue to the next step.
  - rework: Rework from an earlier step.
  - completerepeat: Complete and Repeat
  - unknown: The action is unknown.

step
Step associated with the actions.

class pyclarity_lims.entities.**StepDetails**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Details associated with a step

input_output_maps
List of tuples (input, output) where input and output item are dictionaries representing the input/output. Keys of the dict can be:

- **for the input:**
  - post-process-uri: input *Artifact*
  - uri: input *Artifact*
  - limsid: lims id of the input artifact
  - parent-process: *Process* that generated this input
- **for the output:**
  - uri: output *Artifact*
  - limsid: id of the Artifact generated
  - output-generation-type: type of output generation (example: PerInput)
  - output-type: type of artifact generated (Analyte, or ResultFile)

udf
Dictionary of UDFs associated with the step

udt
Dictionary of UDTs associated with the step

class pyclarity_lims.entities.**StepPlacements**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Placements from within a step. Supports POST

get_placement_list()

get_selected_containers()

placement_list
List of tuples (A, (B, C)) where:

- A is an *artifact*
- B is a *container*
- C is a string specifying the location in the container such as "1:1"

selected_containers
List of *containers*

set_placement_list(*value*)

class pyclarity_lims.entities.**StepPools**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

available_inputs
List of artifact available for pooling.

Note that adding artifacts to a pool will not remove them from this list until put() is run.

pooled_inputs
Dictionary where the keys are the pool names and the values are tuples (pool, inputs) representing a pool. Each tuple has two elements:

- an output Artifact containing the pool.
- a tuple containing the input artifacts for that pool.

put()
Save this instance by doing PUT of its serialized XML.

class pyclarity_lims.entities.**StepProgramStatus**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Status displayed in the step

message
Message returned by the program

status
Status of the program

class pyclarity_lims.entities.**StepReagentLots**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

reagent_lots
List of *ReagentLots*

class pyclarity_lims.entities.**Udfconfig**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Instance of field type (cnf namespace).

allow_non_preset_values
Whether the UDF allows presets.

attach_to_category
–

attach_to_name
Name of entity type the UDF is attached to.

first_preset_is_default_value
Whether the first preset of the UDF is the default value.

is_controlled_vocabulary
Whether the UDF has a controled vocabulary.

is_deviation
Whether the UDF is a deviation.

is_editable
Whether the UDF is editable.

name
Name of the UDF.

presets
List of presets.

show_in_lablink
Whether this UDF will be shown in lablink.

show_in_tables
Whether the UDF can be shown in a table.

class pyclarity_lims.entities.**Workflow**(*lims*, *uri=None*, *id=None*, *_create_new=False*)
Bases: *pyclarity_lims.entities.Entity*

Workflow, introduced in 3.5

name
Name of the workflow.

protocols
List of *protocols* associated with this workflow.

stages
List of *stages* associated with this workflow.

status
Status of the workflow.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

M

N

O

P

Q

R

S

T

U

V

W

X

Y