# slims-python-api Documentation

***Release 0.1***

**Genohm**

**Mar 03, 2022**

# Contents

Contents:

CHAPTER 1

# Introduction

The slims-python-api is a project that allows users to interact easily with SLims using python scripting. All communication is done via SLims' REST API so similar approaches could work for other programming languages.

## 1.1 Installing slims-python-api

You install slims-python-api with pip. Make sure to install the version corresponding to your installed slims version.

If you run SLIMS 6.3 you run

```
pip install 'slims-python-api>=6.3.0,<6.4.0'
```

If you run SLIMS 6.2 you run

```
pip install 'slims-python-api>=6.2.0,<6.3.0'
```

## 1.2 Some simple examples

```
from slims.slims import Slims
from slims.criteria import equals

slims = Slims("test", "https://testserver.com/slimsrest/", "admin", "admin")
content_records = slims.fetch("Content", equals("cntn_id", "test"))
for content_record in content_records:
   print(content_record.cntn_barCode.value + " " +
         content_record.cntn_fk_location.displayValue)
```

Here we fetch all the content record with "test" as their id. Then we loop over them and print their barcode and the name of the location they are in.

```
slims = Slims("test", "https://testserver.com/slimsrest/", "admin", "admin")
content_records = slims.fetch("Content", equals("cntn_id", "test"))

for content_record in content_records:
   content_record.update({"cntn_id", "foo"})
```

We fetch all the content records with “test” as their id and then update it to “foo”

These operations are combined in two of our cookbook examples.

- Fetching and displaying data on the command line https://github.com/genohm/slims-python-api/blob/master/cookbook/data-manipulation/fetching_data.py
- Updating, adding and removing records https://github.com/genohm/slims-python-api/blob/master/cookbook/data-manipulation/data_modification.py
- Sample web application (in web.py) that allows users to submit orders in a simple fashion https://github.com/genohm/slims-python-api/blob/master/cookbook/order-submission/main.py
- Sample web application that shows a table of the latest results in slims https://github.com/genohm/slims-python-api/blob/master/cookbook/live-report/main.py
- Fetching data and then plogging them https://github.com/genohm/slims-python-api/blob/master/cookbook/plotting/main.py

CHAPTER 2

# API Documentation

## 2.1 slims.slims module

## 2.2 slims.criteria module

**class** slims.criteria.**Criterion**
Bases: object

**to_dict**() → dict
Serializes criterion to dictionary

**class** slims.criteria.**Expression**(*criterion: dict*)
Bases: *slims.criteria.Criterion*

A simple expression like 'cntn_id' equals 'test'

**to_dict**() → dict
Serializes criterion to dictionary

**class** slims.criteria.**Junction**(*operator: slims.criteria._JunctionType*)
Bases: *slims.criteria.Criterion*

A combination of multiple criteria

**add**(*member: slims.criteria.Criterion*) → slims.criteria.Junction
Adds a member to this junction

**to_dict**() → dict
Serializes criterion to dictionary

slims.criteria.**between_inclusive**(*field: str*, *start: Any*, *end: Any*) → slims.criteria.Expression
Applies a "between" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match

- **start** (*any*) – the value to start with (inclusive)
- **start** – the value to end with (inclusive)

**Returns** A between criterion

### Examples

```
>>> slims.fetch("Content", between_inclusive("cntn_barCode", "00001", "00010"))
```

Will fetch all the content records that have a barcode between 00001 and 00010

slims.criteria.**conjunction**() → slims.criteria.Junction
Combines multiple criteria in a conjunctive way (and)

**Returns** A conjunction

### Examples

```
>>> slims.fetch("Content", conjunction()
            .add(start_with("cntn_id", "DNA"))
            .add(greater_than("cntn_quantity", 5)))
```

Will fetch all the content records for which their id starts with "DNA" and their quantity is bigger than 5.

slims.criteria.**contains**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies a "contains" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to contain

**Returns** A contains criterion

### Examples

```
>>> slims.fetch("Content", contains("cntn_id", "test"))
```

Will fetch all the content records that have an id that contains "test"

slims.criteria.**disjunction**() → slims.criteria.Junction
Combines multiple criteria in a disjunctive way (or)

**Returns** A disjunction

### Examples

```
>>> slims.fetch("Content", disjunction()
            .add(start_with("cntn_id", "DNA"))
            .add(greater_than("cntn_quantity", 5)))
```

Will fetch all the content records for which their id starts with "DNA" or their quantity is bigger than 5.

slims.criteria.**ends_with**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "ends with" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to end with

**Returns** An ends with criterion

### Examples

```
>>> slims.fetch("Content", ends_with("cntn_id", "001"))
```

Will fetch all the content records that have an id that ends with "001"

slims.criteria.**equals**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "equals" constraint to the specified field

This is case-sensitive depending on the used database.

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to match

**Returns** An equals criterion

### Examples

```
>>> slims.fetch("Content", equals("cntn_id", "dna0001"))
```

This will fetch all the content records that have "dna0001" as their id

slims.criteria.**equals_ignore_case**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "equals" constraint to the specified field

This is always case-insensitive

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to match

**Returns** An equals criterion

### Examples

```
>>> slims.fetch("Content", equals_ignore_case("cntn_id", "dna0001"))
```

Will fetch all the content records that have "dna0001" as their id

slims.criteria.**greater_than**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "greater than" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to match

**Returns** A greater than criterion

### Examples

```
>>> slims.fetch("Content", greater_than("cntn_quantity", "5"))
```

Will fetch all the content records that have a quantity greater than 5

slims.criteria.**greater_than_or_equal**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "greater than or equal" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to match

**Returns** A less than or equal criterion

### Examples

```
>>> slims.fetch("Content", greater_than_or_equal("cntn_quantity", "5"))
```

Will fetch all the content records that have a quantity greater than or equal to 5

slims.criteria.**is_na**(*field: str*) → slims.criteria.Expression
Applies a "is not applicable" constraint to the specified field (this is an option on custom fields

**Parameters** **field** (*string*) – the field that should not be applicable

**Returns** A not applicable criterion

### Examples

```
>>> slims.fetch("Content", is_na("cntn_cf_numberOfSigarettes"))
```

Will fetch all the content records for which the number of sigarrettes is not applible (for example for non smokers)

slims.criteria.**is_not**(*criterion: slims.criteria.Criterion*) → slims.criteria.Junction
Inverts a criterion

**Parameters** **criterion** (*criterion*) – The criterion to invert

**Returns** A criterion

### Examples

```
>>> slims.fetch("Content", is_not(start_with("cntn_id", "DNA")))
```

Will fetch all the content records for which their id does not starts with "DNA"

slims.criteria.**is_not_null**(*field: str*) → slims.criteria.Expression
Applies an "is not null" constraint to the specified field

**Parameters** **field** (*string*) – the field that shouldn't be null

**Returns** A not null criterion

### Examples

```
>>> slims.fetch("Content", is_not_null("cntn_fk_location"))
```

Will fetch all the content records that are in a location

slims.criteria.**is_not_one_of**(*field: str*, *value: list*) → slims.criteria.Expression
Applies an "is not one of" constraint to the specified field

**Parameters**

- **field** (*string*) –
- **value** (*list*) –

### Examples

```
>>> slims.fetch("Content", is_not_one_of("cntn_barCode", ["0001", "0002", "0004
↪"]))
```

Will fetch all the content records that have a barcode that is not 0001, 0002 or 0004.

slims.criteria.**is_null**(*field: str*) → slims.criteria.Expression
Applies an "is null" constraint to the specified field

**Parameters** **field** (*string*) – the field that should be null

**Returns** An is null criterion

### Examples

```
>>> slims.fetch("Content", is_null("cntn_fk_location"))
```

Will fetch all the content records that are not in a location

slims.criteria.**is_one_of**(*field: str*, *value: list*) → slims.criteria.Expression
Applies an "is one of" constraint to the specified field

**Parameters**

- **field** (*string*) –
- **value** (*list*) –

### Examples

```
>>> slims.fetch("Content", is_one_of("cntn_barCode", ["0001", "0002", "0004"]))
```

Will fetch all the content records that have a barcode that is either 0001, 0002 or 0004.

slims.criteria.**less_than**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "less than" constraint to the specified field

**Parameters**

- **field** (*string*) – the field to match
- **value** (*any*) – the value to match

**Returns** A less than criterion

### Examples

```
>>> slims.fetch("Content", less_than("cntn_quantity", "5"))
```

Will fetch all the content records that have a quantity smaller than 5

`slims.criteria.`**`less_than_or_equal`**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "less than or equal" constraint to the specified field

**Parameters**

- **`field`** (*`string`*) – the field to match
- **`value`** (*`any`*) – the value to match

**Returns** A less than or equal criterion

### Examples

```
>>> slims.fetch("Content", less_than_or_equal("cntn_quantity", "5"))
```

Will fetch all the content records that have a quantity less than or equal to 5

`slims.criteria.`**`not_equals`**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies an "not equals" constraint to the specified field

**Parameters**

- **`field`** (*`string`*) – the field to match
- **`value`** (*`any`*) – the value not to match

**Returns** A not equals criterion

### Examples

```
>>> slims.fetch("Content", not_equals("cntn_id", "dna0001"))
```

Will fetch all the content records that do not have "dna0001" as their id

`slims.criteria.`**`starts_with`**(*field: str*, *value: Any*) → slims.criteria.Expression
Applies a "starts with" constraint to the specified field

**Parameters**

- **`field`** (*`string`*) – the field to match
- **`value`** (*`any`*) – the value to start with

**Returns** A starts with criterion

### Examples

```
>>> slims.fetch("Content", start_with("cntn_id", "dna"))
```

Will fetch all the content records that have an id that starts with "dna"

## 2.3 slims.step module

**class** slims.step.**Step**(*name: str*, *action: Callable*, *asynchronous: bool = False*, *hidden: bool = False*, *input: list = []*, *output: list = []*, ***kwargs*)

Bases: object

The step class defines the step properties of a SLimsGate flow.

**execute**(*flow_run: slims.flowrun.FlowRun*) → Any

takes id of the flow run and it executes the flow. :param flow_run: flow to run :type flow_run: object

**Returns** if synchronous returns the value of action

**to_dict**(*route_id: str*) → dict

Construct and return a dict with all (except action) class attribute and the route id passed by argument. :param route_id: id of the route :type route_id: string

**Returns** (dic) of the attributes (except action) and route id

**exception** slims.step.**StepExecutionException**

Bases: Exception

slims.step.**boolean_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have a yes or no choice input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**date_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have a date input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**date_time_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have a date and time input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**file_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have a file input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**file_output**() → dict
Allows to have a file output for SLimsGate.

Returns: file output to download in client side

slims.step.**float_input**(*name: str*, *label: str*, ***kwargs*) → dict
Allows to have float input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**integer_input**(*name: str*, *label: str*, ***kwargs*) → dict
Allows to have integer input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**multiple_choice_with_field_list_input**(*name: str*, *label: str*, *fieldelements: list*, *fieldtype: list = None*, ***kwargs*) → dict
Allows to have a multiple choice out of a list input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **fieldelements** (*list*) – the list of elements in which the choice needs to be made, usually strings
- **fieldtype** (*list*) – the type of the field elements its default value is None
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**multiple_choice_with_value_map_input**(*name: str*, *label: str*, *table: str = None*, *filtered: Any = None*, *reference: str = None*, *fixed_choice_custom_field: str = None*, ***kwargs*) → dict

Allows to have a multiple choice out of a list input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **table** (*String*) – the table name of the possible choices to display its default value is None
- **filtered** (*Criteria object*) – the filter applied on the list of displayed choice its default value is None
- **reference** (*String*) – the name of the valueMap of the output of the previous step_dicts its default value is None
- **fixed_choice_custom_field** (*String*) – the name of a custom field its default value is None
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**password_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have a password input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**rich_text_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have rich text input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**single_choice_with_field_list_input**(*name: str*, *label: str*, *fieldelements: list*, *fieldtype: list = None*, ***kwargs*) → dict

Allows to have a single choice out of a list input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input

- **fieldelements** (*list*) – the list of elements in which the choice needs to be made, usually strings
- **fieldtype** (*list*) – the type of the field elements its default value is None
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**single_choice_with_value_map_input**(*name: str*, *label: str*, *table: str = None*, *filtered: Any = None*, *reference: str = None*, *fixed_choice_custom_field: str = None*, ***kwargs*) → dict

Allows to have a single choice out of a list input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **table** (*String*) – the table name of the possible choices to display its default value is None
- **filtered** (*Criteria object*) – the filter applied on the list of displayed choice its default value is None
- **reference** (*String*) – the name of the valueMap of the output of the previous step_dicts its default value is None
- **fixed_choice_custom_field** (*String*) – the name of a custom field its default value is None
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**table_input**(*name: str*, *label: str*, *subparameters: list*, ***kwargs*) → dict

Allows to have a table input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **subparameters** (*list*) – the list of parameters that need to be in the table
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**text_input**(*name: str*, *label: str*, ***kwargs*) → dict

Allows to have short text input for SLimsGate by return a dictionary.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**time_input**(*name: str*, *label: str*, ***kwargs*) → dict
Allows to have a time input for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the input
- **label** (*String*) – the label of the input
- **-- every additional and optional parameter** (***kwargs*) – it needs to be of the form defaultValue="it is a default value"

Returns: (dict) a dictionary containing all these elements

slims.step.**value_map_output**(*name: str*, *datatype: str*) → dict
Allows to have a value map output for SLimsGate.

**Parameters**

- **name** (*String*) – the name of the output
- **datatype** (*String*) – the label of the output

Returns: (dict) a dictionary containing all these elements

## 2.4 slims.flowrun module

**class** slims.flowrun.**FlowRun**(*slims_api: slims.internal._SlimsApi*, *index: str*, *data: dict*)
Bases: object

**log**(*message: str*) → None
Logs a message to Slims

**Parameters** **message** (*string*) – the message to log

### Example

```
>>> def step_action(flow_run):
        flow_run.log("Hello from python")
```

**class** slims.flowrun.**Status**
Bases: enum.Enum

Status of a flow run step

**DONE = 1**

**FAILED = 2**

## 2.5 slims.output module

slims.output.**file_value**(*file_name: str*) → dict
Opens the file with file_name and returns its content as a string.

**Parameters** **file_name** (*string*) – Name of the file to read.

**Returns** value(String) and "fileName":value(String)

**Return type** (dic) with “bytes”

## 2.6 slims.content module

slims.content.**Status**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*)

List of content statusses in SLims

Can be used to fetch or update content

### Examples

```
>>> slims.fetch("Content",
        equals("cntn_status", Status.PENDING.value))
```

Deprecated since version Enum-value: statuses are deprecated since SLIMS 6.4. Unless your SLIMS system still uses them (see Lab Settings), you should use the Status table and cntn_fk_status for status queries.

## 2.7 slims.util module

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## S

# Index

## T



## V