

---

# **slims-python-api Documentation**

***Release 0.1***

**Genohm**

**Apr 29, 2022**



---

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Installing slims-python-api . . . . .	3
1.2	Some simple examples . . . . .	3
<b>2</b>	<b>API Documentation</b>	<b>5</b>
2.1	slims.slims module . . . . .	5
2.2	slims.criteria module . . . . .	8
2.3	slims.step module . . . . .	13
2.4	slims.flowrun module . . . . .	18
2.5	slims.output module . . . . .	18
2.6	slims.content module . . . . .	18
2.7	slims.util module . . . . .	19
<b>3</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</b>



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](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](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>



## 2.1 slims.slims module

```
class slims.slims.Slims(name: str, url: str, username: str = None, password: str = None,  
                        oauth: bool = False, client_id: str = None, client_secret: str = None,  
                        repo_location: str = None, local_host: str = 'localhost', local_port: int  
                        = 5000, **request_params)
```

Bases: object

Creates a new slims instance to work with

### Parameters

- **name** (*str*) – The name of this slims instance
- **url** (*str*) – The url of the REST API of this slims instance
- **username** (*str*, *optional*) – The username to login with (needed for standard operations)
- **password** (*str*, *optional*) – The password to login with (needed for standard operations)
- **oauth** (*bool*, *optional*) – Whether OAuth authentication is used
- **client\_id** (*str*, *optional*) – The client ID used to authenticate when OAuth is true
- **client\_secret** (*str*, *optional*) – The client secret used to authenticate when OAuth is true
- **repo\_location** (*str*, *optional*) – The location of the file repository (this can be used to access attachments without needing to download them)
- **local\_host** (*str*, *optional*) – The IP on which this python script is running Needed for SLimsGate flows. SLims will contact the python script on this url. Defaults to “localhost”

- **local\_port** (*int, optional*) – The port on which this python script is running Needed for ports. SLims will contact the python script on this ports. Defaults to “5000”
- **request\_params** – Parameters to pass verbatim to requests when calling the REST API, e.g. `verify='path/to/cert'`

**add** (*table: str, values: dict*) → `slims.internal.Record`

Add a new record in slims

#### Parameters

- **table** (*string*) – Table where the element need to be added.
- **values** (*dict*) – The values of the new record

**Returns** The added record

#### Examples

```
>>> slims.add("Content", {
    "cntn_id", "ID",
    "cntn_status", Status.PENDING.value,
    "cntn_fk_contentType", 1
})
```

Adds a content record with id “ID” in status pending with the content type with primary key 1

**add\_flow** (*flow\_id: str, name: str, usage: str, steps: list, testing: bool = False, last\_flow: bool = True*)  
→ `None`

Add a new SLimsGate flow to the slims interface

---

**Note:** Adding a slimsgate flow means your python script will continue executing until you shut it down.

---

#### Parameters

- **flow\_id** (*string*) – Technical identicator of the flow
- **name** (*string*) – Displayed name of the the flow
- **usage** (*string*) – Usage of the slimsgate flow
- **steps** (*list step*) – The steps of the slimsgate flow
- **testing** (*bool*) – Dry run=====
- **last\_flow** (*boolean*) – Defines if this is the last flow you will add (Default True)

#### Examples

```
>>> def hello_world(flow_run):
    print("Hello world")
>>> slims.add_flow(
    flow_id="helloWorld",
    name="Make python say hello",
    usage="CONTENT_MANAGEMENT",
    steps=[
        Step(
            name="The step",
```

(continues on next page)

(continued from previous page)

```
        action=hello_world
    )))
```

**fetch** (*table: str, criteria: slims.criteria.Criterion, sort: list = None, start: int = None, end: int = None*)  
 → List[slims.internal.Record]  
 Fetch data by criteria

The optional start and end parameters can be used to page the returned results.

#### Parameters

- **table** (*str*) – The table to fetch from
- **criteria** (*criteria*) – The criteria to match
- **sort** (*list, optional*) – The fields to sort on
- **start** (*int, optional*) – The first row to return
- **end** (*int, optional*) – The last row to return

**Returns** The list of matched records

#### Examples

```
>>> slims.fetch("Content",
                start_with("cntn_id", "DNA"),
                sort = ["cntn_barCode"],
                start = 10,
                end = 20)
```

Fetches content records that have an id that starts with DNA. The returned list is sorted by cntn\_barCode (ascending). The first returned results is the has the 10th barcode and the last one is the 20th

```
>>> slims.fetch("Content",
                start_with("cntn_id", "DNA"),
                sort = ["-cntn_barCode"])
```

Fetches content records that have an id that starts with DNA. The returned list is sorted by cntn\_barCode (descending).

**fetch\_by\_pk** (*table: str, pk: int*) → Optional[slims.internal.Record]  
 Fetch a record by primary key

#### Parameters

- **table** (*string*) – The table of the record
- **pk** (*int*) – The primary key of the record

**Returns** A single record (or None)

#### Examples

```
>>> slims.fetch_by_pk("Content", 1)
```

**token\_updater** (*token: dict*) → None

## 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)
- **end** – 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 applicable (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", "0005"]))
```

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 a “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 statuses 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

`slims.util.display_field_value` (*record*: *slims.internal.Record*, *fields*: *list*) → None

Prints the results depending on the field.

### Parameters

- **(object** (*record*) – Record): the results to displayed
- **fields** (*list*) – the fields to displayed it needs to be a list of string ([“field”] or [“field1”, “field2”])

`slims.util.display_results` (*records*: *list*, *fields*: *list*, *number*: *int* = None) → None

Prints to display the filtered results as a list of elements with their selected fields.

### Parameters

- **records** (*list*) – List of results to display
- **fields** (*list*) – List of fields(String) to display it needs to be a list of string ([“field”] or [“field1”, “field2”])
- **number** (*int*) – the number of displayed results default value is None which displayed all existing results





## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### S

- `slims.content`, 18
- `slims.criteria`, 8
- `slims.flowrun`, 18
- `slims.output`, 18
- `slims.slims`, 5
- `slims.step`, 13
- `slims.util`, 19



## A

add() (*slims.criteria.Junction method*), 8  
 add() (*slims.slims.Slims method*), 6  
 add\_flow() (*slims.slims.Slims method*), 6

## B

between\_inclusive() (*in module slims.criteria*), 8  
 boolean\_input() (*in module slims.step*), 14

## C

conjunction() (*in module slims.criteria*), 8  
 contains() (*in module slims.criteria*), 8  
 Criterion (*class in slims.criteria*), 8

## D

date\_input() (*in module slims.step*), 14  
 date\_time\_input() (*in module slims.step*), 14  
 disjunction() (*in module slims.criteria*), 9  
 display\_field\_value() (*in module slims.util*), 19  
 display\_results() (*in module slims.util*), 19  
 DONE (*slims.flowrun.Status attribute*), 18

## E

ends\_with() (*in module slims.criteria*), 9  
 equals() (*in module slims.criteria*), 9  
 equals\_ignore\_case() (*in module slims.criteria*), 10  
 execute() (*slims.step.Step method*), 13  
 Expression (*class in slims.criteria*), 8

## F

FAILED (*slims.flowrun.Status attribute*), 18  
 fetch() (*slims.slims.Slims method*), 7  
 fetch\_by\_pk() (*slims.slims.Slims method*), 7  
 file\_input() (*in module slims.step*), 14  
 file\_output() (*in module slims.step*), 14  
 file\_value() (*in module slims.output*), 18  
 float\_input() (*in module slims.step*), 14  
 FlowRun (*class in slims.flowrun*), 18

## G

greater\_than() (*in module slims.criteria*), 10  
 greater\_than\_or\_equal() (*in module slims.criteria*), 10

## I

integer\_input() (*in module slims.step*), 15  
 is\_na() (*in module slims.criteria*), 10  
 is\_not() (*in module slims.criteria*), 11  
 is\_not\_null() (*in module slims.criteria*), 11  
 is\_not\_one\_of() (*in module slims.criteria*), 11  
 is\_null() (*in module slims.criteria*), 11  
 is\_one\_of() (*in module slims.criteria*), 12

## J

Junction (*class in slims.criteria*), 8

## L

less\_than() (*in module slims.criteria*), 12  
 less\_than\_or\_equal() (*in module slims.criteria*), 12  
 log() (*slims.flowrun.FlowRun method*), 18

## M

multiple\_choice\_with\_field\_list\_input() (*in module slims.step*), 15  
 multiple\_choice\_with\_value\_map\_input() (*in module slims.step*), 15

## N

not\_equals() (*in module slims.criteria*), 13

## P

password\_input() (*in module slims.step*), 16

## R

rich\_text\_input() (*in module slims.step*), 16

## S

`single_choice_with_field_list_input()`  
(in module *slims.step*), 16

`single_choice_with_value_map_input()` (in  
module *slims.step*), 16

*Slims* (class in *slims.slims*), 5

*slims.content* (module), 18

*slims.criteria* (module), 8

*slims.flowrun* (module), 18

*slims.output* (module), 18

*slims.slims* (module), 5

*slims.step* (module), 13

*slims.util* (module), 19

`starts_with()` (in module *slims.criteria*), 13

*Status* (class in *slims.flowrun*), 18

*Status()* (in module *slims.content*), 18

*Step* (class in *slims.step*), 13

*StepExecutionException*, 13

## T

`table_input()` (in module *slims.step*), 17

`text_input()` (in module *slims.step*), 17

`time_input()` (in module *slims.step*), 17

`to_dict()` (*slims.criteria.Criterion* method), 8

`to_dict()` (*slims.criteria.Expression* method), 8

`to_dict()` (*slims.criteria.Junction* method), 8

`to_dict()` (*slims.step.Step* method), 13

`token_updater()` (*slims.slims.Slims* method), 7

## V

`value_map_output()` (in module *slims.step*), 17