limeclient Documentation

Release 0.0.1

Lundalogik AB

February 02, 2015

Contents

1 Importing Files **3**
- 1.1 File Recommendations . . . 3
- 1.2 Concepts . . . 3
- 1.3 Importing a file . . . 3
- 1.4 Authenticating . . . 4
- 1.5 Uploading a file . . . 4
- 1.6 Retrieving the LIME data type . . . 4
- 1.7 Creating an import configuration . . . 4
- 1.8 Behavior . . . 5
- 1.9 Mapping . . . 5
- 1.10 Adding a simple field mapping . . . 5
- 1.11 Adding a mapping to an option field . . . 6
- 1.12 Mapping relations . . . 6
- 1.13 Starting an import job . . . 7

2 Import API **9**
- 2.1 LimeClient . . . 9
- 2.2 LimeTypes . . . 9
- 2.3 ImportFiles . . . 11
- 2.4 Import Configuration . . . 12
- 2.5 Import Jobs . . . 13

3 Indices and tables **15**

Python Module Index **17**

Contents:

CHAPTER 1

Importing Files

The LIME Pro API supports importing simple text files.

1.1 File Recommendations

The first line of the file should contain names for the columns in the rest of the file.

The file should preferably be encoded using UTF-8.

1.2 Concepts

In order to start an import, we need to define three major components, an import file, an import configuration, and an import job.

The import file contains not only the contents of the file itself, but also some additional information, such as what delimiter is used to separate the columns.

The import configuration contains information about what file to import, what type of LIME objects we want to import the file to, and how to map what's in the file to fields on the fields and relation of that LIME type.

Finally, when **the import job** is created it will start the actual import as soon as it can. The import job will then contain the status of the import, and any errors that might have occured during the process.

1.3 Importing a file

Let's say we have the following file containing information about people we want to add to LIME:

```
name;e-mail;ship;rank
Ripley;ellen@nostromo.com;Nostromo;Warrant Officer
Dallas;dallas@weyland.com;Nostromo;Captain
Ash;ash@weyland.com;Nostromo;Science Officer
...
```

We have a LIME database that contains information about employees and ships, and we've been tasked with importing this file in to LIME.

We install the latest version of limeclient, create a module called weyland.py.

1.4 Authenticating

First of all, we need to authenticate ourselves with the LIME Pro server. If we're using the API-client this done by creating an instance of `LimeClient` and logging in:

```
from limeclient import LimeClient

 client = LimeClient('https://myserver:2134', 'weyland_db')
 with client.login('user', 'pass') as c:
     # Do stuff here...
```

This will start a new session in LIME that will automatically be closed when program is finished.

If you're importing data to a hosted LIME installation, you should not pass the database name argument.

1.5 Uploading a file

Once we've authenticated ourselves, it's time for us to tell LIME about the file we want to import.

```
with client.login('user', 'pass') as c:
    with open('nostromo_crew.txt') as content:
        f = ImportFiles(c).create(filename='nostromo_crew.txt',
                                  content=content)
        f.delimiter = ';'
        f.save()
```

Here we uploaded a file to LIME, which returns with a file object populated with default values. We then told LIME a little bit more about how it should interpret the file, and saved the new information on the server.

1.6 Retrieving the LIME data type

The second piece of information we need before we can start configuring how to interpret the content of the file is an object that represents the data type in LIME.

If the information in the file contains information about people we might want to load the 'crew' `LimeTypes`:

```
with client.login('user', 'pass') as c:
    with open('nostromo_crew.txt') as content:
        f = ImportFiles(c).create(filename='nostromo_crew.txt',
                                  content=content)
        f.delimiter(';')
        f.save()

    crew = LimeTypes(c).get_by_name('crew')
```

1.7 Creating an import configuration

With that we have enough information to start configuring our import:

```
with client.login('user', 'pass') as c:
    with open('nostromo_crew.txt') as content:
        f = ImportFiles(c).create(filename='nostromo_crew.txt',
```

```
                                content=content)
        f.delimiter(';')
        f.save()

    crew = LimeTypes(c).get_by_name('crew')

    config = ImportConfigs(c).create(lime_type=crew, importfile=f)
```

1.8 Behavior

We can tell LIME what it should do for each row it finds in our import file.

config.behavior = ImportConfig.CreateAndUpdate Update existing objects if they match what's in the file, and create new objects if nothing matches. This is the default value for a new `ImportConfig`

config.behavior = ImportConfig.OnlyUpdate Only update objects that match what's in the import file. Don't create any new objects.

config.behavior = ImportConfig.OnlyCreate Create a new object for each row in the import file. Don't try to match against existing objects in LIME.

1.9 Mapping

For each row in the file to import, LIME needs to know what to do with the data. We tell LIME how to accomplish this by telling it how to map each column to something a field or relation of the data type we're import to.

The LIME API supports three types of mappings: mappings of simple types, such as strings and numbers, mappings option fields where the value can be one of several predefined values, and finally relations to other types of entities in LIME, such as persons being related to companies.

1.10 Adding a simple field mapping

In our example, the name and e-mail of the crew members are simple types, so we add simple field mappings for those. We also mark the name field as a key field, meaning that we will use this when determining if this row matches an existing object in LIME.

```
with client.login('user', 'pass') as c:
    # ...

    crew = LimeTypes(c).get_by_name('crew')

    config = ImportConfigs(c).create(lime_type=crew, importfile=f)

    name = SimpleFieldMapping(field=crew.fields['name'],
                              column='name',
                              key=False)
    config.add_mapping(name)

    email = SimpleFieldMapping(field=crew.fields['email'],
                              column='e-mail',
                              key=True)
    config.add_mapping(email)
```

We can specify 'key=True' for multiple mappings. In that case all values must match for the import to consider updating a person in LIME instead of adding a new.

1.11 Adding a mapping to an option field

You can map a column in the import file to an option field in LIME by adding an `OptionFieldMapping` to your import configuration.

Within the `OptionFieldMapping`, you specify how a value in a column translates to one of the possible values of an option field in LIME.

```
with client.login('user', 'pass') as c:
    # ...

    field = crew.fields['rank']
    position = OptionFieldMapping(field=field, column='rank')
    position.default = field.option_by_key('Engineer')
    position.map_option(column_val='Captain',
                        option=field.option_by_key('Captain'))
    position.map_option(column_val='Warrant Officer',
                        option=field.option_by_key('Warrant Officer'))
    config.add_mapping(position)
```

In the example above we first say that any values for the 'rank' column that haven't been explicitly mapped, we should assume that the crew member is engineer.

We then proceed to explicitly map the values for captain and warrant officer.

1.12 Mapping relations

Finally, we need to import the ship of each crew member in the file. 'Ship' is a separate table in the LIME database and we need to tell the import about this:

```
with client.login('user', 'pass') as c:
    # ...

    crew = LimeTypes(c).get_by_name('crew')

    # ...

    relation = crew.relations['ship']
    ship = relation.related
    relation_mapping = RelationMapping(column='ship', relation=relation,
                                       key_field=ship.fields['name'])
    config.add_mapping(relation_mapping)

    config.save()
```

We ask the lime type for the relation to the ship type, we use that to get a hold of the actual ship type. We then tell the importer that the 'ship' column contains names of ships.

Now, we can save the import configuration and are ready to start the import.

1.13 Starting an import job

We can now start the import job:

```
with client.login(user=args.user, password=args.password) as c:
    # ...

    job = ImportJobs(c).create(config)

    for i in range(10):
       time.sleep(1)
       job = job.refresh()
       print('Current job status: {}'.format(job.status))
       if job.has_errors:
           print('Oh noes! Errors!')
           print(job.errors.errors[:10])
       if job.status != 'pending' and job.status != 'running':
           break
```

This tells LIME to put the import job on a queue. We then proceed to poll the status of the job. If something goes wrong, the ten first errors will be printed to the console.

CHAPTER 2

Import API

This part of the documentation covers the different classes and methods available in limeclient.

2.1 LimeClient

class `limeclient.`**`LimeClient`**(*host*, *database=None*, *debug=False*, *verify_ssl_cert=True*)

Handles all communication with LIME's API

Parameters

- **host** – name of host to connect to
- **database** – name of database to logon to. Should not be specified when logging on to a hosted LIME solution.
- **debug** – if *True*, print traffic to stdout. Defaults to *False*
- **verify_ssl_cert** – if *False*, ignore SSL certificate verification. Defaults to *True*.

`login`(*user=None*, *password=None*)

Log in to LIME.

`LimeClient` should be used as a context manager. That way logging out and closing a session will be done automatically, even if an error is encountered.

```
client = LimeClient('localhost', 'mydatabase')
with client.login('user', 'pass') as c:
    # do stuff
```

2.2 LimeTypes

class `limeclient.`**`LimeTypes`**(*lime_client*)

Retrieve type information about entities in LIME Pro.

Parameters **lime_client** – a logged in `LimeClient` instance

`get_by_name`(*name*)

Retrieve a `LimeType` given its name in LIME Pro.

Parameters **name** – name in LIME Pro (e.g. 'company')

`get_by_url`(*url*)

Retrieve a `LimeType` given its url.

Parameters **url** – this is the url that uniquely identifies an lime type.

class `limeclient.`**`LimeType`**(*hal*, *lime_client*)
Represents a type of object in LIME Pro.

`name`

Name of lime type.

`is_system`

True if this is a system type.

`fields`
Retrieve all fields for this lime type.

`relations`
Retrieve all relations (`Relation`) for this lime type.

2.2.1 Field Types

class `limeclient.`**`SimpleField`**(*hal*, *lime_client*)
Represents a simple field type.

`label`

`length`

`localname`

`name`

`readonly`

`required`

`type`

class `limeclient.`**`OptionField`**(*hal*, *lime_client*)
Represents an option field type.

`label`

`localname`

`name`

`readonly`

`required`

`type`

`option_by_id`(*id*)
Retrieve an `Option` value given its id.

`option_by_key`(*key*)
Retrieve an `Option` value given its key.

`option_by_localname`(*localname*)
Retrieve an `Option` value given its local name.

`options`
All possible `Option` values for this field.

class `limeclient.`**`Option`**(*raw*)
Represents a possible value for an `OptionField`.

`id`
Id of the option value

`key`
Key of the option value

`localname`
Local name of the option value

2.2.2 Relations

class `limeclient.`**`Relation`**(*hal*, *lime_client*)
Represents a relation to another lime type in LIME Pro.

`localname`

`name`

`related`
The related `LimeType`

2.3 ImportFiles

class `limeclient.`**`ImportFiles`**(*lime_client*)
Handles uploading of import files to LIME

Parameters **lime_client** – a logged in `LimeClient` instance

`create`(*filename*, *content*)
Upload an import file to LIME. Returns an `ImportFile` instance

Parameters

- **filename** – name of uploaded file
- **content** – a file object containing the data to import

class `limeclient.`**`ImportFile`**(*hal*, *lime_client*)
Represents a file to import to LIME.

Parameters

- **hal** – representation of an import file as returned from LIME.
- **hal** – lime_client `LimeClient` to use for communication with LIME

`delimiter`

Use this to set the delimiter used in the file so LIME knows how to parse it.

`filename`

The name of the file.

`headers`

Returns the headers (`ImportFileHeaders`) of the file.

`save`()

Save the file information in LIME.

class `limeclient.`**`ImportFileHeaders`**(*hal*, *lime_client*)
Contains the headers of a parsed import file.

Parameters

- **hal** – representation of an import file as returned from LIME.
- **hal** – lime_client `LimeClient` to use for communication with LIME

`headers`

A list of the header names in the file

2.4 Import Configuration

class `limeclient.`**`ImportConfigs`**(*lime_client*)
Manages creation of `ImportConfig` instances in LIME

Parameters **lime_client** – a logged in `LimeClient` instance

`create`(*lime_type*, *importfile*)
Create a new `ImportConfig` instance in LIME.

Parameters

- **lime_type** – `LimeType` instance that references what type of data to import
- **importfile** – `ImportFile` instance to import

class `limeclient.`**`ImportConfig`**(*hal*, *lime_client*)
Used for configuring an import.

Parameters

- **hal** – representation of an import file as returned from LIME.
- **hal** – lime_client `LimeClient` to use for communication with LIME

`behavior`

Determines how the import handles existing objects in LIME Pro. Can be one of the following:

•*ImportConfig.CreateAndUpdate*

•*ImportConfig.UpdateOnly*

•*ImportConfig.CreateOnly*

`add_mapping`(*mapping*)
Add information about how to map a column in the import file to data in LIME.

Parameters **mapping** – One of `SimpleFieldMapping`, `OptionFieldMapping`, or `RelationMapping`.

static `create`(*lime_client*, *lime_type*, *importfile*)
Create a new instance of `ImportConfig`

Parameters

- **lime_client** – a logged in `LimeClient` instance
- **lime_type** – `LimeType` instance that references what type of data to import
- **importfile** – `ImportFile` instance to import

save()
Save the import configuration in LIME.

validate()

Ask LIME to validate the import configuration. Returns an `ImportConfigStatus` instance.

2.4.1 Mapping

The following types can be passed to `ImportConfig.add_mapping()` to define how columns in the import file should be mapped to fields in LIME Pro:

class limeclient.**SimpleFieldMapping**(*column*, *field*, *key=False*)
Maps a column to a simple field on the object we want to import to.

Parameters

- **column** – Name of column in import file
- **field** – the field we want to map to
- **key** – if *True*, the value of this column will be used to find existing objects in LIME Pro.

class limeclient.**OptionFieldMapping**(*column*, *field*)
Maps a column to a simple field on the object we want to import to.

Parameters

- **column** – Name of column in import file
- **field** – the field we want to map to

default
The value to give the field if none of the mappings apply for the value in the column.

map_option(*column_val*, *option*)
Map a value for a column to an option for a field.

Parameters

- **column_val** – a value of the column in the import file
- **option** – a `Option` instance. The option value to map to.

class limeclient.**RelationMapping**(*column*, *relation*, *key_field*)
Use the value in a column to find a related object in LIME Pro.

Parameters

- **column** – column that we want to map
- **relation** – the `Relation` that we want to map.
- **key_field** – the field of the related type that we will match against to find related objects.

2.5 Import Jobs

class limeclient.**ImportJobs**(*lime_client*)
Handles the creation of a new import job.

Parameters **lime_client** – a logged in `LimeClient` instance

create(*import_config*)

Create a new `ImportJob`. This indicates to the server that it can start executing the job as soon as possible.

Parameters import_config – a ready `ImportConfig` instance.

get(*url*)

Retrieve an existing `ImportJob` from the server.

Parameters url – the url that identifies the job on the server.

class `limeclient.`**ImportJob**(*hal*, *lime_client*)

Represents an import job on the server.

Parameters lime_client – a logged in `LimeClient` instance

created_time

Time when the job was created.

started_time

Time when job was actually started.

completed_time

Time when job was completed.

created_count

Number of new items created.

updated_count

Number of already existing items that where updated.

errors_count

Number of errors encountered.

status

Current status of the job. Can be one of 'pending', 'running', 'done', 'done_with_errors', or 'failed'

errors

Retrieve a `ImportJobErrors` that contains all errors for this job

has_errors

Determine if this job has encountered any errors.

refresh()

Retrieve a fresh version of the import job from the server.

class `limeclient.`**ImportJobErrors**(*hal*, *lime_client*)

count

The number of errors encountered during the job.

errors

A collection of all errors, grouped by rows in the imported file.

Example:

```
[{'row': 4, 'errors': ["Value of field 'name' is longer than the allowed 32 characters"]}]
```

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index

l

limeclient, 9

A

add_mapping() (limeclient.ImportConfig method), 12

B

behavior (limeclient.ImportConfig attribute), 12

C

completed_time (limeclient.ImportJob attribute), 14
count (limeclient.ImportJobErrors attribute), 14
create() (limeclient.ImportConfig static method), 12
create() (limeclient.ImportConfigs method), 12
create() (limeclient.ImportFiles method), 11
create() (limeclient.ImportJobs method), 13
created_count (limeclient.ImportJob attribute), 14
created_time (limeclient.ImportJob attribute), 14

D

default (limeclient.OptionFieldMapping attribute), 13
delimiter (limeclient.ImportFile attribute), 11

E

errors (limeclient.ImportJob attribute), 14
errors (limeclient.ImportJobErrors attribute), 14
errors_count (limeclient.ImportJob attribute), 14

F

fields (limeclient.LimeType attribute), 10
filename (limeclient.ImportFile attribute), 11

G

get() (limeclient.ImportJobs method), 14
get_by_name() (limeclient.LimeTypes method), 9
get_by_url() (limeclient.LimeTypes method), 9

H

has_errors (limeclient.ImportJob attribute), 14
headers (limeclient.ImportFile attribute), 11
headers (limeclient.ImportFileHeaders attribute), 12

I

id (limeclient.Option attribute), 11
ImportConfig (class in limeclient), 12
ImportConfigs (class in limeclient), 12
ImportFile (class in limeclient), 11
ImportFileHeaders (class in limeclient), 12
ImportFiles (class in limeclient), 11
ImportJob (class in limeclient), 14
ImportJobErrors (class in limeclient), 14
ImportJobs (class in limeclient), 13
is_system (limeclient.LimeType attribute), 10

K

key (limeclient.Option attribute), 11

L

label (limeclient.OptionField attribute), 10
label (limeclient.SimpleField attribute), 10
length (limeclient.SimpleField attribute), 10
LimeClient (class in limeclient), 9
limeclient (module), 9
LimeType (class in limeclient), 10
LimeTypes (class in limeclient), 9
localname (limeclient.Option attribute), 11
localname (limeclient.OptionField attribute), 10
localname (limeclient.Relation attribute), 11
localname (limeclient.SimpleField attribute), 10
login() (limeclient.LimeClient method), 9

M

map_option() (limeclient.OptionFieldMapping method), 13

N

name (limeclient.LimeType attribute), 10
name (limeclient.OptionField attribute), 10
name (limeclient.Relation attribute), 11
name (limeclient.SimpleField attribute), 10

O

Option (class in limeclient), 10

option_by_id() (limeclient.OptionField method), 10
option_by_key() (limeclient.OptionField method), 10
option_by_localname() (limeclient.OptionField method), 10
OptionField (class in limeclient), 10
OptionFieldMapping (class in limeclient), 13
options (limeclient.OptionField attribute), 10

R

readonly (limeclient.OptionField attribute), 10
readonly (limeclient.SimpleField attribute), 10
refresh() (limeclient.ImportJob method), 14
related (limeclient.Relation attribute), 11
Relation (class in limeclient), 11
RelationMapping (class in limeclient), 13
relations (limeclient.LimeType attribute), 10
required (limeclient.OptionField attribute), 10
required (limeclient.SimpleField attribute), 10

S

save() (limeclient.ImportConfig method), 12
save() (limeclient.ImportFile method), 11
SimpleField (class in limeclient), 10
SimpleFieldMapping (class in limeclient), 13
started_time (limeclient.ImportJob attribute), 14
status (limeclient.ImportJob attribute), 14

T

type (limeclient.OptionField attribute), 10
type (limeclient.SimpleField attribute), 10

U

updated_count (limeclient.ImportJob attribute), 14

V

validate() (limeclient.ImportConfig method), 13