edeposit.amqp.ltp

Release 1.0.3

September 14, 2015

Contents

This project provides AMQP bindings for LTP (Long Time Preservation) system used in Czech National Library.

The LTP is basically archive for digital documents for long time periods (hundred of years).

Access to this archive is restricted, so if you wish to use this module for yourself, you will need to negotiate access for yourself.

CHAPTER 1

API

1.1 __init__.py

This module contains bindings to AMQP.

1.1.1 API

`ltp.`**`_instanceof`**(*instance*, *cls*)
Check type of *instance* by matching `.__name__` with *cls.__name__*.

`ltp.`**`reactToAMQPMessage`**(*message*, *send_back*)
React to given (AMQP) message. *message* is expected to be `collections.namedtuple()` structure from `structures` filled with all necessary data.

Parameters

- **message** (*object*) – One of the request objects defined in `structures`.
- **send_back** (*fn reference*) – Reference to function for responding. This is useful for progress monitoring for example. Function takes one parameter, which may be response structure/namedtuple, or string or whatever would be normally returned.

Returns Response class from `structures`.

Return type object

Raises `ValueError` – if bad type of *message* structure is given.

1.1.2 Submodules

ltp.py

This module contains functions to create SIP package for the LTP system.

API

`ltp.ltp.`**`_get_package_name`**(*prefix='/tmp'*, *book_id=None*)
Return package path. Use uuid to generate package's directory name.

Parameters

- **book_id** (*str, default None*) – UUID of the book.
- **prefix** (*str, default settings.TEMP_DIR*) – Where the package will be stored. Default `settings.TEMP_DIR`.

Returns Path to the root directory.

Return type str

`ltp.ltp.`**`_create_package_hierarchy`**(*prefix='/tmp'*, *book_id=None*)

Create hierarchy of directories, at it is required in specification.

root_dir is root of the package generated using `settings.TEMP_DIR` and `_get_package_name()`.

orig_dir is path to the directory, where the data files are stored.

metadata_dir is path to the directory with MODS metadata.

Parameters

- **book_id** (*str, default None*) – UUID of the book.
- **prefix** (*str, default settings.TEMP_DIR*) – Where the package will be stored. Default `settings.TEMP_DIR`.

Warning: If the *root_dir* exists, it is REMOVED!

Returns root_dir, orig_dir, metadata_dir

Return type list of str

`ltp.ltp.`**`create_ltp_package`**(*aleph_record*, *book_id*, *ebook_fn*, *data*, *url*, *urn_nbn=None*)

Create LTP package as it is specified in specification v1.0 as I understand it.

Parameters

- **aleph_record** (*str*) – XML containing full aleph record.
- **book_id** (*str*) – UUID of the book.
- **ebook_fn** (*str*) – Original filename of the ebook.
- **data** (*str/bytes*) – Ebook's content.
- **url** (*str*) – URL of the publication used when the URL can't be found in *aleph_record*.
- **urn_nbn** (*str, default None*) – URN:NBN.

Returns Name of the package's directory in `/tmp`.

Return type str

checksum_generator submodule

This submodule is used to generate MD5 checksums for data and metadata files in SIP package.

It also used to create *hash file*, which holds all checksums with paths to the files from root of the package. For example path `/home/xex/packageroot/somedir/somefile.txt` will be stored as `/packageroot/somedir/somefile.txt`.

API

Checksum generator in format specified in LTP specification.

ltp.checksum_generator.**_get_required_fn**(*fn*, *root_path*)
Definition of the MD5 file requires, that all paths will be absolute for the package directory, not for the filesystem.

This function converts filesystem-absolute paths to package-absolute paths.

Parameters

- **fn** (*str*) – Local/absolute path to the file.
- **root_path** (*str*) – Local/absolute path to the package directory.

Returns Package-absolute path to the file.

Return type str

Raises `ValueError` – When *fn* is absolute and *root_path* relative or conversely.

ltp.checksum_generator.**generate_checksums**(*directory*, *blacklist=set(['info.xml'])*)
Compute checksum for each file in *directory*, with exception of files specified in *blacklist*.

Parameters

- **directory** (*str*) – Absolute or relative path to the directory.
- **blacklist** (*list/set/tuple*) – List of blacklisted filenames. Only filenames are checked, not paths!

Returns Dict in format `{fn: md5_hash}`.

Return type dict

Note: File paths are returned as absolute paths from package root.

Raises `UserWarning` – When *directory* doesn't exists.

ltp.checksum_generator.**generate_hashfile**(*directory*, *blacklist=set(['info.xml'])*)
Compute checksum for each file in *directory*, with exception of files specified in *blacklist*.

Parameters

- **directory** (*str*) – Absolute or relative path to the directory.
- **blacklist** (*list/set/tuple*) – List of blacklisted filenames. Only filenames are checked, not paths!

Returns Content of hashfile as it is specified in ABNF specification for project.

Return type str

fn_composers submodule

This module holds few functions used to dynamically construct filenames for files in SIP package.

API

Filenames are generated dynamically. Here is set of composers of filanames.

ltp.fn_composers.**_get_suffix**(*path*)
Return suffix from *path*.

`/home/xex/somefile.txt` -> `txt`.

Parameters **path** (*str*) – Full file path.

Returns Suffix.

Return type str

Raises `UserWarning` – When `/` is detected in suffix.

ltp.fn_composers.**original_fn**(*book_id*, *ebook_fn*)
Construct original filename from *book_id* and *ebook_fn*.

Parameters

- **book_id** (*int/str*) – ID of the book, without special characters.
- **ebook_fn** (*str*) – Original name of the ebook. Used to get suffix.

Returns Filename in format `oc_nk-BOOKID.suffix`.

Return type str

ltp.fn_composers.**metadata_fn**(*book_id*)
Construct filename for metadata file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `meds_nk-BOOKID.xml`.

Return type str

ltp.fn_composers.**volume_fn**(*cnt*)
Construct filename for 'volume' metadata file.

Parameters **cnt** (*int*) – Number of the MODS record.

Returns Filename in format `mods_volume.xml` or `mods_volume_cnt.xml`.

Return type str

ltp.fn_composers.**checksum_fn**(*book_id*)
Construct filename for checksum file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `MD5_BOOKID.md5`.

Return type str

ltp.fn_composers.**info_fn**(*book_id*)
Construct filename for info.xml file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `info_BOOKID.xml`.

Return type str

settings submodule

Module is containing all necessary global variables for the package.

Module also has the ability to read user-defined data from two paths:

- `$HOME/_SETTINGS_PATH`
- `/etc/_SETTINGS_PATH`

See `_SETTINGS_PATH` for details.

Note: If the first path is found, other is ignored.

Example of the configuration file (`$HOME/edeposit/ltp.json`):

```
{
    "EXPORT_DIR": "/somedir/somewhere"
}
```

Attributes

`ltp.settings.`**`TEMP_DIR`** **= '/tmp'**
Path to the temporary directory, where the packages are built.

`ltp.settings.`**`EXPORT_DIR`** **= '/home/ltp/edep2ltp'**
Path to the directory for LTP export.

`ltp.settings.`**`IMPORT_DIR`** **= '/home/ltp/ltp2edep'**
Path to the directory for LTP import.

structures submodule

1.2 ltp.py

This module contains functions to create SIP package for the LTP system.

1.2.1 API

`ltp.ltp.`**`_get_package_name`**(*prefix='/tmp'*, *book_id=None*)
Return package path. Use uuid to generate package's directory name.

Parameters

- **book_id** (*str, default None*) – UUID of the book.
- **prefix** (*str, default settings.TEMP_DIR*) – Where the package will be stored. Default `settings.TEMP_DIR`.

Returns Path to the root directory.

Return type str

`ltp.ltp.`**`_create_package_hierarchy`**(*prefix='/tmp'*, *book_id=None*)
Create hierarchy of directories, at it is required in specification.

root_dir is root of the package generated using `settings.TEMP_DIR` and `_get_package_name()`.

orig_dir is path to the directory, where the data files are stored.

metadata_dir is path to the directory with MODS metadata.

Parameters

- **book_id** (*str, default None*) – UUID of the book.
- **prefix** (*str, default settings.TEMP_DIR*) – Where the package will be stored. Default `settings.TEMP_DIR`.

Warning: If the *root_dir* exists, it is REMOVED!

Returns root_dir, orig_dir, metadata_dir

Return type list of str

`ltp.ltp.`**`create_ltp_package`**(*aleph_record*, *book_id*, *ebook_fn*, *data*, *url*, *urn_nbn=None*)

Create LTP package as it is specified in specification v1.0 as I understand it.

Parameters

- **aleph_record** (*str*) – XML containing full aleph record.
- **book_id** (*str*) – UUID of the book.
- **ebook_fn** (*str*) – Original filename of the ebook.
- **data** (*str/bytes*) – Ebook's content.
- **url** (*str*) – URL of the publication used when the URL can't be found in *aleph_record*.
- **urn_nbn** (*str, default None*) – URN:NBN.

Returns Name of the package's directory in `/tmp`.

Return type str

1.3 fn_composers submodule

This module holds few functions used to dynamically construct filenames for files in SIP package.

1.3.1 API

Filenames are generated dynamically. Here is set of composers of filanames.

`ltp.fn_composers.`**`_get_suffix`**(*path*)

Return suffix from *path*.

`/home/xex/somefile.txt` -> `txt`.

Parameters **path** (*str*) – Full file path.

Returns Suffix.

Return type str

Raises `UserWarning` – When `/` is detected in suffix.

`ltp.fn_composers.`**`original_fn`**(*book_id*, *ebook_fn*)

Construct original filename from *book_id* and *ebook_fn*.

Parameters

- **book_id** (*int/str*) – ID of the book, without special characters.
- **ebook_fn** (*str*) – Original name of the ebook. Used to get suffix.

Returns Filename in format `oc_nk-BOOKID.suffix`.

Return type str

`ltp.fn_composers.`**`metadata_fn`**(*book_id*)
Construct filename for metadata file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `meds_nk-BOOKID.xml`.

Return type str

`ltp.fn_composers.`**`volume_fn`**(*cnt*)
Construct filename for 'volume' metadata file.

Parameters **cnt** (*int*) – Number of the MODS record.

Returns Filename in format `mods_volume.xml` or `mods_volume_cnt.xml`.

Return type str

`ltp.fn_composers.`**`checksum_fn`**(*book_id*)
Construct filename for checksum file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `MD5_BOOKID.md5`.

Return type str

`ltp.fn_composers.`**`info_fn`**(*book_id*)
Construct filename for info.xml file.

Parameters **book_id** (*int/str*) – ID of the book, without special characters.

Returns Filename in format `info_BOOKID.xml`.

Return type str

1.4 checksum_generator submodule

This submodule is used to generate MD5 checksums for data and metadata files in SIP package.

It also used to create *hash file*, which holds all checksums with paths to the files from root of the package. For example path `/home/xex/packageroot/somedir/somefile.txt` will be stored as `/packageroot/somedir/somefile.txt`.

1.4.1 API

Checksum generator in format specified in LTP specification.

`ltp.checksum_generator.`**`_get_required_fn`**(*fn*, *root_path*)
Definition of the MD5 file requires, that all paths will be absolute for the package directory, not for the filesystem.

This function converts filesystem-absolute paths to package-absolute paths.

Parameters

- **fn** (*str*) – Local/absolute path to the file.
- **root_path** (*str*) – Local/absolute path to the package directory.

Returns Package-absolute path to the file.

Return type str

Raises `ValueError` – When *fn* is absolute and *root_path* relative or conversely.

`ltp.checksum_generator.`**`generate_checksums`**(*directory*, *blacklist=set(['info.xml'])*)
Compute checksum for each file in *directory*, with exception of files specified in *blacklist*.

Parameters

- **directory** (*str*) – Absolute or relative path to the directory.
- **blacklist** (*list/set/tuple*) – List of blacklisted filenames. Only filenames are checked, not paths!

Returns Dict in format `{fn: md5_hash}`.

Return type dict

Note: File paths are returned as absolute paths from package root.

Raises `UserWarning` – When *directory* doesn't exists.

`ltp.checksum_generator.`**`generate_hashfile`**(*directory*, *blacklist=set(['info.xml'])*)
Compute checksum for each file in *directory*, with exception of files specified in *blacklist*.

Parameters

- **directory** (*str*) – Absolute or relative path to the directory.
- **blacklist** (*list/set/tuple*) – List of blacklisted filenames. Only filenames are checked, not paths!

Returns Content of hashfile as it is specified in ABNF specification for project.

Return type str

1.5 structures submodule

1.6 settings submodule

Module is containing all necessary global variables for the package.

Module also has the ability to read user-defined data from two paths:

- `$HOME/_SETTINGS_PATH`
- `/etc/_SETTINGS_PATH`

See `_SETTINGS_PATH` for details.

Note: If the first path is found, other is ignored.

Example of the configuration file (`$HOME/edeposit/ltp.json`):

```
{
    "EXPORT_DIR": "/somedir/somewhere"
}
```

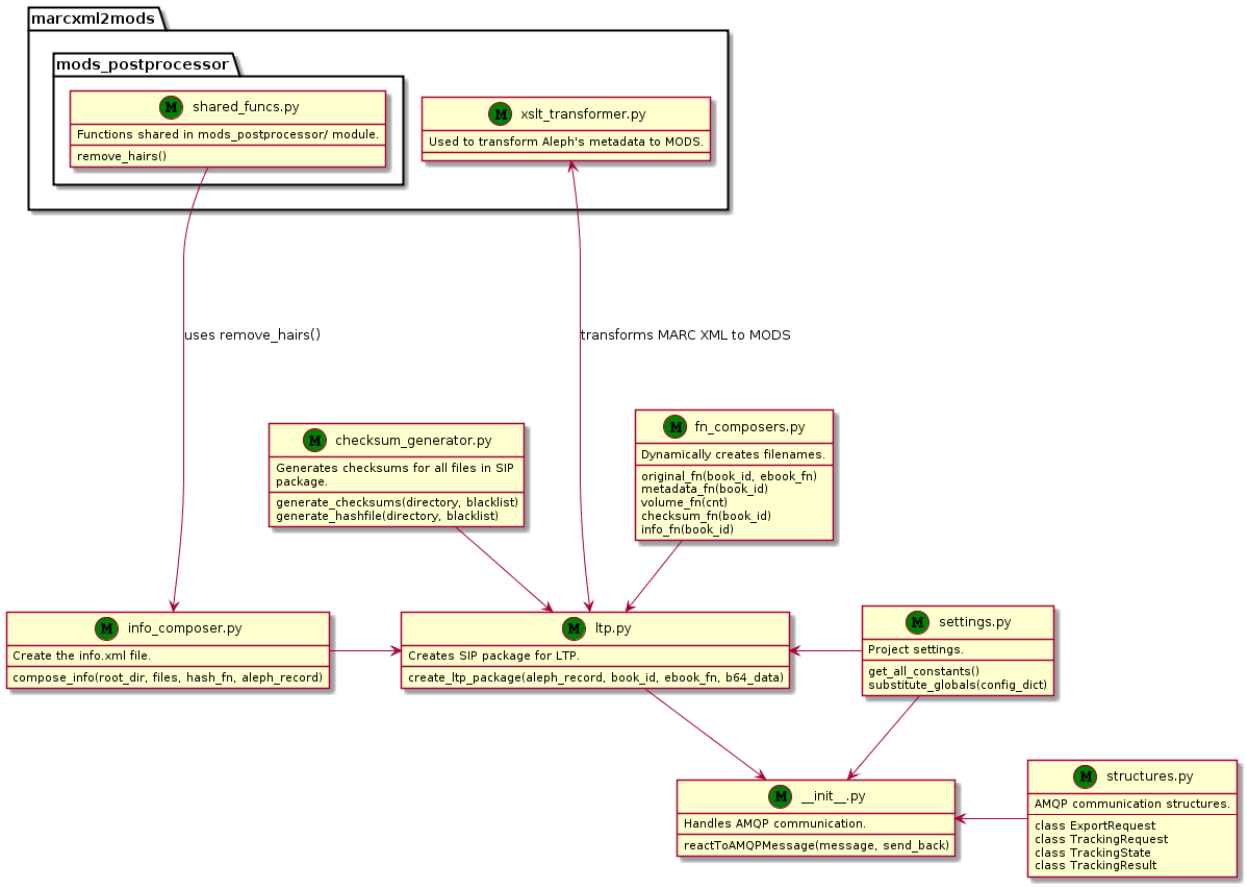
1.6.1 Attributes

ltp.settings.**TEMP_DIR = '/tmp'**
Path to the temporary directory, where the packages are built.

ltp.settings.**EXPORT_DIR = '/home/ltp/edep2ltp'**
Path to the directory for LTP export.

ltp.settings.**IMPORT_DIR = '/home/ltp/ltp2edep'**
Path to the directory for LTP import.

API relations graph



CHAPTER 3

AMQP connection

AMQP communication is handled by the edeposit.amqp module, specifically by the `edeposit_amqp_ltpd.py` script. Bindings to this project are handled by `reactToAMQPMessage()`.

CHAPTER 4

Source code

This project is released as opensource (GPL) and source codes can be found at GitHub:

- https://github.com/edeposit/edeposit.amqp.ltp

4.1 Installation

Module is hosted at PYPI, and can be easily installed using PIP:

```
sudo pip install edeposit.amqp.ltp
```

CHAPTER 5

Testing

Almost every feature of the project is tested in unit/integration tests. You can run this tests using provided `run_tests.sh` script, which can be found in the root of the project.

5.1 Requirements

This script expects that pytest is installed. In case you don't have it yet, it can be easily installed using following command:

```
pip install --user pytest
```

or for all users:

```
sudo pip install pytest
```

CHAPTER 6

Indices and tables

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

l

Symbols

C

E

G

I

L

M

O

R

T

V