Python XMP Toolkit Documentation

Release 2.0.1

European Space Agency (ESA), European Southern Observator

August 17, 2015

Contents

Python XMP Toolkit is a library for working with XMP metadata, as well as reading/writing XMP metadata stored in many different file formats.

Authors:

- Lars Holm Nielsen <lars@hankat.dk>
- John Evans <john.g.evans.ne@gmail.com>
- Federico Caboni <federico.caboni@me.com>
- Amit Kapadia <akapad@gmail.com>

Part I

Documentation

CHAPTER 1

Introduction

Python XMP Toolkit is a library for working with XMP metadata, as well as reading/writing XMP metadata stored in many different file formats.

Python XMP Toolkit is wrapping Exempi (using ctypes), a C/C++ XMP library based on Adobe XMP Toolkit, ensuring that future updates to the XMP standard are easily incorporated into the library with a minimum amount of work.

Python XMP Toolkit has been developed by:

- ESA/Hubble - European Space Agency
- ESO - European Southern Observatory
- CRS4 - Centre for Advanced Studies, Research and Development in Sardinia

1.1 Feature Overview

The XMP features provided are similar to that of Adobe XMP Toolkit, which are:

- Support for parsing, manipulating, and serializing XMP data.
- Support for locating the XMP in a file, adding XMP to a file, or updating the XMP in a file.
- Support for nearly any file format with smart file handlers for JPEG, TIFF, GIF, PNG, PSD, InDesign, MOV, MP3, MPEG2, MPEG4, AVI, FLV, SWF, ASF, PostScript, P2, SonyHDV, AVCHD, UCF, WAV, XDCAM, XDCAMEX.
- An API very similar to Adobe XMP Toolkit.
- Based on Exempi 2.1.1 and Adobe XMP Toolkit 4.4.2

Following important features from Adobe XMP Toolkit are not available in Python XMP Toolkit:

- Localized text support
- Methods for working with XMP structs.
- Methods for working with XMP qualifiers
- Methods for working with XMP Aliases

1.2 What is XMP?

The Adobe Extensible Metadata Platform (XMP) specification describes a widely used method for embedding descriptive metadata within images. XMP tags are stored within the image header of all common image formats (JPEG, TIFF, PNG, GIF, PSD) and can be read by popular image processing and cataloging packages. The XMP standard is also widely used by photographers and the publication industry. Users of consumer and professional digital cameras may already be familiar with Exchangeable Image File Format (EXIF) metadata tags that include

camera and exposure information within the digital photo file as a set of XMP tags. In practice an XMP header is a block of XML text included in the header block of the image file and is only supported in image types with header/comment blocks.

The advantages of embedded image identity metadata are numerous. Including metadata effectively makes the images self-documenting, which is particularly useful when the source URL for an image is lost. This information can now be accessed by multimedia management packages, or indexed by databases designed to read the embedded information. For instance, an online or desktop planetarium program could load an image from the web and extract the appropriate metadata to place it in the proper position in the sky.

CHAPTER 2

Installation

2.1 Requirements

- Python 2.6, 2.7, or 3.3
- Exempi 2.2.0+
- Linux or OS X (see notes below for Windows)

2.2 Python XMP Toolkit

The short version of installation is:

```
python setup.py install
```

or if you use pip:

```
pip install python-xmp-toolkit
```

Note, in case you haven't installed Exempi you will get an `ExempiLoadError` exception once you try to load `libxmp`.

2.3 Exempi

Python XMP Toolkit requires Exempi 2.2.0 or higher which can be downloaded from http://libopenraw.freedesktop.org/wiki/Exempi. It is probably already installed if you are working on linux.

To install Exempi, unpack the distribution and run:

```
./configure
make
sudo make install
```

Versions below 2.2.0 will not work. Note, Exempi may also be available in your systems package manager, e.g.:

```
sudo apt-get install libexempi3  # (Ubuntu/Debian)
brew install exempi  # (Homebrew on OS X)
```

2.4 Mac OS X

Note Exempi requires boost (http://www.boost.org/) to compile, so on OS X you probably need to run configure with one of the following options:

```
./configure --with-boost=/usr/local # (for Homebrew)
./configure --with-darwinports
./configure --with-fink
```

2.5 Windows

The library has not been tested on Windows, and nor has any serious effort been made to test it. Hence, developers wanting to use the library on Windows are encouraged to try it out and let us know if it works.

The library ought to work on Windows, if Exempi can be compiled as a DLL using e.g. Cygwin.

CHAPTER 3

Using Python XMP Toolkit

This little tutorial will show you two different methods for how to read/write XMP documents from files as well as manipulate them metadata once extracted from the file.

The tutorial is meant to be understood without prior knowledge of XMP. However, readers who decides to use the library are strongly encouraged to gain basic knowledge and understanding of:

- XMP Data Model
- XMP Serialization

A basic understanding of these two concepts can save yourself from common misunderstandings of what XMP is and what XMP can do. Good resources are e.g. the wiki page or the XMP Specification Part 1 available from:

- http://en.wikipedia.org/wiki/Extensible_Metadata_Platform
- http://www.adobe.com/devnet/xmp/

3.1 Method 1: Read XMP

One of the most basic uses of the library is:

```
>>> from libxmp.utils import file_to_dict
>>> xmp = file_to_dict( "test/samples/BlueSquare.xmp" )
```

This will read the XMP embedded in the file and return it as a dictionary. The keys in the dictionary are XMP namespaces so to e.g. get all Dublin Core properties use:

```
>>> from libxmp import consts
>>> dc = xmp[consts.XMP_NS_DC]
```

or to be explicit:

```
>>> dc = xmp["http://purl.org/dc/elements/1.1/"]
```

This will give you a list of all Dublin Core properties, where each element in the list is a tuple. The first element is the property name, the second element is the value and the third element is options associated with the element (describing e.g the type of the property):

First tuple element:

```
>>> print(dc[0][0])
dc:format
```

Second tuple element:

```
>>> print(dc[0][1])
application/vnd.adobe.photoshop
```

Third tuple element is a dict with options:

```
>>> dc[0][2]['IS_SCHEMA']
False
```

3.2 Method 2: Read/Write XMP

Example 1 focused on just extracting the XMP from a file an determine the value of a property. If you however want to extract the XMP from a file, update it, *and* write it back again you need to do like the following

Read file:

```
>>> from libxmp import XMPFiles, consts
>>> xmpfile = XMPFiles( file_path="test/samples/BlueSquare.jpg", open_forupdate=True )
```

Get XMP from file:

```
>>> xmp = xmpfile.get_xmp()
```

Print the property `dc:format`:

```
>>> print(xmp.get_property(consts.XMP_NS_DC, 'format' ))
image/jpeg
```

Change the XMP property:

```
>>> xmp.set_property(consts.XMP_NS_DC, u'format', u'application/vnd.adobe.illustrator' )
>>> print(xmp.get_property(consts.XMP_NS_DC, 'format' ))
application/vnd.adobe.illustrator
```

Check if XMP document can be written to file and write it:

```
>>> xmpfile.can_put_xmp(xmp)
True
```

XMP document is not written to the file, before the file is closed:

```
>>> xmpfile.close_file()
```

3.3 Further Examples

Append an array item to the XMP packet.:

Read file:

```
>>> from libxmp import XMPFiles, consts
>>> xmpfile = XMPFiles( file_path="test/samples/BlueSquare.xmp" )
```

Get XMP from file:

```
>>> xmp = xmpfile.get_xmp()
```

Create a new array item and append a value:

```
>>> xmp.append_array_item(consts.XMP_NS_DC, 'creator', 'Your Name Here', {'prop_array_is_ordered'
```

CHAPTER 4

Reference

4.1 XMP Toolkit

4.1.1 Exceptions

XMPError

ExempiLoadError

4.1.2 Core Module

XMPMeta

XMPIterator

4.1.3 Files Module

XMPFiles

4.1.4 Utils Module

4.1.5 Constants

CHAPTER 5

Developers

This section is intended for developers of Python XMP Toolkit.

To obtain a source distribution go to GitHub at https://github.com/python-xmp-toolkit/python-xmp-toolkit and clone the repository.

5.1 Overview of Source Distribution

- `docs/` – Source code for documentation.
- `libxmp/` – Source files for XMP Toolkit
- `setup.py` – Distutils configuration file.
- `MANIFEST.in` – Template for MANIFEST file used by Distutils.
- `test` – Tests

5.2 Documentation

Documentation is prepared using Sphinx Python Documentation Generator (see http://sphinx.pocoo.org/). To make the documentation run the following command in the root directory:

```
pip install sphinx
python setup.py build_sphinx
```

5.3 Packaging a Distribution

To package a distribution run:

```
python setup.py sdist
```

This will prepare the documentation and use distutils to package together a distribution that will be placed in `dist/`.

5.4 Running Tests

Tests are run by issuing the command:

```
python setup.py test
```

For test coverage, run:

```
pip install coverage
source run-coverage.sh
```

To run tests in Python 2.6, 2.7 and 3.3 using tox, run:

```
pip install tox
tox
```

5.5 Distribution Configuration

The file `setup.py` specify how the distribution is packed together. Most important to note is that version in formation is read from `libxmp.version` file, and that the file `MANIFEST.in` specifies which other files to include in the distribution besides the Python source.

5.6 References for Developers

- ctypes
- Sphinx
- Distutils

CHAPTER 6

Appendix

6.1 Known Issues

- The version of libexempi that comes via Macports refuses to load via ctypes. As a workaround, you should compile libexempi from source.

6.2 Resources

- Project website – https://github.com/python-xmp-toolkit/python-xmp-toolkit
- XMP – http://www.adobe.com/products/xmp/
- Exempi – http://libopenraw.freedesktop.org/wiki/Exempi
- Adobe XMP Toolkit – http://www.adobe.com/devnet/xmp/

6.3 Glossary

XMP eXtensible Metadata Platform

6.4 TODO list

6.5 License

6.6 Changes

Release 2.0.0 (January 17, 2013)

- Added support for Python 3.3, dropped support for 2.5.
- All string outputs in 2.6 and 2.7 are unicode objects with UTF-8 encoding.
- Added exempi module, a low-level ctypes interface to exempi library.
- Most XMPMeta methods that formerly returned bool now raise exceptions in case of failure, except for “does_property_exist”, “does_array_item_exist”.
- Added timezone support for datetime routines.
- Added delete_localized_text method to XMPMeta

Release 1.0.2 (June 21, 2011)

- Fixed python 2.5 issue (ctypes.c_bool are not available in 2.5, so it was changed to ctypes.c_int).

Release 1.0.1 (April 11, 2011)

- Fixed issue on 32-bit systems.

Release 1.0 (March 31, 2011)

- Known issue #7 documented - issue with TIFF smart handler.
- Fixed issue #15 - 64-bit issues on Linux and Mac.
- Fixed issue #11 - Typo in does_property_exist.
- Thanks to marialaura.clemente for bug reports and patches.

Release 1.0rc2 (Feburary 16, 2010)

- Fixed issue #4, #5 and related to XMPIterator, file_to_dict and object_to_dict.
- Fixed issue in file_to_dict which didn’t pass parameters to XMPFiles.open_file() properly.
- Fixed issue #9 file_to_dict now raises IOError instead of returning an empty dictionary for non-existing files. (*backward incompatible*)
- Fixed issue #8 - spelling mistake in function call in XMPMeta.append_array_items
- Based on Exempi 2.1.1 and Adobe XMP Toolkit 4.4.2
- Thanks to olsenpk, pitymushroom, rmarianski, cfarrell1980 for bug reports and patches

Release 1.0rc1 (March 6, 2009)

- Backwards incompatible with previous releases.
- Based on Exempi 2.1.0 and Adobe XMP Toolkit 4.4.2
- Initialise and Terminate should no longer be called before usage.

Release 1.0beta1 (July 6, 2008)

- First public release.

Part II

Indices and tables

- genindex
- modindex
- search

X