XMLStarlet CFFI Documentation

Release 1.6.8

Mikhail Grushinskiy

Feb 07, 2023

Contents:

CHAPTER 1

XMLStarlet CFFI

XMLStarlet Toolkit: Python CFFI bindings

- Free software: MIT license
- Documentation (this package): https://xmlstarlet.readthedocs.io.
- Original XMLStarlet Documentation: http://xmlstar.sourceforge.net/doc/UG/

1.1 Features

Supports all XMLStarlet commands from Python, just *import xmlstarlet*:

- *edit(*args)*: Edit/Update XML document(s)
- *select(*args)*: Select data or query XML document(s) (XPATH, etc)
- *transform(*args)*: Transform XML document(s) using XSLT
- *validate(*args)*: Validate XML document(s) (well-formed/DTD/XSD/RelaxNG)
- *format(*args)*: Format XML document(s)
- *elements(*args)*: Display element structure of XML document
- *canonicalize(*args)*: XML canonicalization
- *listdir(*args)*: List directory as XML (**NOT** supported on Windows)
- *escape(*args)*: Escape special XML characters
- *unescape(*args)*: Unescape special XML characters
- *pyx(*args)*: Convert XML into PYX format (based on ESIS - ISO 8879)

- *depyx(*args)*: Convert PYX into XML

For some examples, have a look at *tests/test_xmlstarlet.py*.

1.2 Credits

Kudos to XMLStarlet and its maintainers and users (original sources on SourceForge)!

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

Binary wheels built via GitHub Actions by cibuildwheel

CHAPTER 2

Installation

2.1 Stable release

To install XMLStarlet CFFI, run this command in your terminal:

```
$ pip install xmlstarlet
```

This is the preferred method to install XMLStarlet CFFI, as it will always install the most recent stable release from https://pypi.org.

Binary wheels are automatically built and published for all major OS platforms (Linux, MacOS, and Windows), as well as source packages on every tagged release.

Supported and tested on (64-bit) Python versions from 3.6+.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for XMLStarlet CFFI can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dimitern/xmlstarlet
```

Or download the tarball:

```
$ curl -OL https://github.com/dimitern/xmlstarlet/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

Ideally, you would do this inside a *virtualenv*.

CHAPTER 3

Usage

Let's see some usage examples and description of each command.

3.1 xml CLI Help

The original `xml` CLI command has the following help description:

```
XMLStarlet Toolkit: Command line utilities for XML
Usage: ./xml [<options>] <command> [<cmd-options>]
where <command> is one of:
  ed    (or edit)      - Edit/Update XML document(s)
  sel   (or select)    - Select data or query XML document(s) (XPATH, etc)
  tr    (or transform) - Transform XML document(s) using XSLT
  val   (or validate)  - Validate XML document(s) (well-formed/DTD/XSD/RelaxNG)
  fo    (or format)    - Format XML document(s)
  el    (or elements)  - Display element structure of XML document
  c14n  (or canonic)   - XML canonicalization
  ls    (or list)      - List directory as XML
  esc   (or escape)    - Escape special XML characters
  unesc (or unescape)  - Unescape special XML characters
  pyx   (or xmln)      - Convert XML into PYX format (based on ESIS - ISO 8879)
  p2x   (or depyx)     - Convert PYX into XML
<options> are:
  -q or --quiet        - no error output
  --doc-namespace      - extract namespace bindings from input doc (default)
  --no-doc-namespace   - don't extract namespace bindings from input doc
  --version            - show version
  --help               - show help
Wherever file name mentioned in command help it is assumed
that URL can be used instead as well.

Type: .xml <command> --help <ENTER> for command help

```

(continues on next page)

(continued from previous page)

```
XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see http://xmlstar.sourceforge.net/)
```

3.1.1 From Python

To use XMLStarlet CFFI in a project:

```
import xmlstarlet
```

Each command takes the same string arguments as the C version of `xmlstarlet`, and returns an integer exit code (0 means success).

Some examples for supported commands can be seen below.

3.2 xmlstarlet.edit()

Original `xml` CLI help text for `edit`:

```
XMLStarlet Toolkit: Edit XML document(s)
Usage: ./xml ed <global-options> {<action>} [ <xml-file-or-uri> ... ]
where
  <global-options>  - global options for editing
  <xml-file-or-uri> - input XML document file name/uri (stdin otherwise)

<global-options> are:
  -P, or -S           - preserve whitespace nodes.
     (or --pf, --ps)    Note that space between attributes is not preserved
  -O (or --omit-decl) - omit XML declaration (<?xml ...?>)
  -L (or --inplace)   - edit file inplace
  -N <name>=<value>   - predefine namespaces (name without 'xmlns:')
                        ex: xsql=urn:oracle-xsql
                        Multiple -N options are allowed.
                        -N options must be last global options.
  --net               - allow network access
  --help or -h        - display help

where <action>
  -d or --delete <xpath>
  --var <name> <xpath>
  -i or --insert <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -a or --append <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -s or --subnode <xpath> -t (--type) elem|text|attr -n <name> [-v (--value) <value>]
  -m or --move <xpath1> <xpath2>
  -r or --rename <xpath1> -v <new-name>
  -u or --update <xpath> -v (--value) <value>
                         -x (--expr) <xpath>

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see http://xmlstar.sourceforge.net/)
```

Let's assume you have this `test.xml` file you want to modify:

```
<h:html xmlns:h="urn:local:html">
        <h:body>
                <h:p>
                        <h:a h:href="#">
                                Link
                        </h:a>
                </h:p>
        </h:body>
</h:html>
```

The modification is to find the first hyperlink, extract its text content, and add it as the value of a new attribute `text=` on the root (`html`) element, like so:

```
<?xml version="1.0"?>
<h:html xmlns:h="urn:local:html" text="Link">
        <h:body>
                <h:p>
                        <h:a h:href="#">
                                Link
                        </h:a>
                </h:p>
        </h:body>
</h:html>
```

Here's how you can use the *edit* command to achieve this:

```
result = xmlstarlet.edit(
     "-S",
     "-N", "_=urn:local:html",
     "--var", "foo", "translate(//_:a[1]/text(), ' \n', '')",
     "-s", "/_:html", "-t", "attr", "-n", "text", "-v", "X",
     "-u", "$prev", "-x", "$foo",
     "./test.xml",
     "./test2.xml",
)
if result != 0:
   print("Cannot update the XML")
```

This demonstrates a number of options and techniques:

`-S` preserve whitespaces in the input (do not trim).

`-N _=urn:local:html` define namespaces present in the input (usable in expressions), here we define _ as the namespace prefix for `urn:local:html`.

`--var foo translate(//_:a[1]/text(), ' \n', '')` assign the result of an XPath expression (in this case, a function call removing spaces and new-lines from the text content of the first `a` element), to a named variable `foo`.

`-s /_:html -t attr -n text -v X` create a subnode (in this case, attribute), named `text`, with value `X` (temporarily), as a child of the root `h:html` element.

`-u $prev -x $foo` update the node at the given XPath with the result of another XPath expression. In this case, the special variable `$prev` contains the last matched XPath (`/_:html`), and the variable `$foo` contains `"Link"`.

`./test.xml` the input XML file to operate on.

`./test2.xml` the output XML file (will be **overwritten**).

Tip: More examples can be found in the original `xmlstarlet` edit documentation.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/dimitern/xmlstarlet/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

XMLStarlet CFFI could always use more documentation, whether as part of the official XMLStarlet CFFI docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dimitern/xmlstarlet/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *xmlstarlet* for local development.

1. Fork the *xmlstarlet* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/xmlstarlet.git
```

3. Install your local copy into a virtualenv. Assuming you have Python 3 installed, this is how you set up your fork for local development:

```
$ cd xmlstarlet/
$ python3 -m venv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
```

The following one-liner command goes through all steps: cleans all build artifacts (if any), uninstalls the package (if installed), runs the linters (asserting scores haven't gone down and no new issues are found), the formatter (checking formatting won't change any of the files), builds a source distribution, then a binary wheel, running all tests, producing a coverage HTML report, and finally building the sphinx HTML, displayed in a browser on completion:

```
$ invoke clean --uninstall lint format --check dist --wheel test coverage docs --
↪browser
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ invoke format lint test  # optional; tox runs those as well
$ tox
```

Both *invoke* and *tox* are already installed from *requirements.txt*. To re-create all the *tox* environments and run all matrix combinations:

```
$ tox -r -e ALL  # equivalent to `invoke clean-tests --tox`
```

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push -u origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6 and later (currently, up to 3.10). Check https://github.com/dimitern/xmlstarlet/pulls and make sure all checks pass OK. Binary wheels are built automatically for each PR, or *git push* to a branch.

4.4 Tips

To run a subset of tests:

```
$ pytest tests.test_xmlstarlet
```

(*python setup.py test* will also work as alias of *pytest*).

4.5 Deploying

A reminder for the maintainers on how to deploy.

Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ invoke release --dry-run
```

This runs *tox*, and then displays how the new version will look like, without pushing anything.

If it goes OK, make the actual release with:

```
$ invoke release
```

CHAPTER 5

Credits

5.1 XMLStarlet Developers

- Mikhail Grushinskiy <mgrouch@users.sourceforge.net>
- Dagobert Michelsen <dmichelsen@users.sourceforge.net>
- Noam Postavsky <npostavs@users.sourceforge.net>

5.2 XMLStarlet CFFI Python Bindings Maintainer

- Dimiter Naydenov <dimitern@users.noreply.github.com>

5.3 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 1.6.9 (unreleased)

- Now using latest *cibuildwheel* 2.8.1, which supports more architectures and builds, including PyPy 3.7 - 3.10 (on Linux x86 and x64, and MacOS x64).
- Updated versions in *requirements.txt*.
- Fixed security warnings (e.g. CVE-611) around xml2 and xslt libraries, pinning their versions to at least libxml2 2.9.1 and libxslt 1.1.28.

6.2 1.6.8 (2022-04-30)

- Added Python 3.10 support.
- Fixed issue #199 (pending confirmation) - upgraded libxml2 and libxslt versions to fix CVEs
- Upgraded development and build-time dependencies.
- Now using latest *cibuildwheel* 2.5.0, which supports more architectures and builds.
- Started to improve the documentation - added better usage examples.
- Formatting and linting fixes

6.3 1.6.7 (2020-12-24)

- Fixed MacOS binary wheel builds

6.4 1.6.6 (2020-10-04)

- Simplified and automated building source and binary wheels for Linux, MacOS, and Windows via GitHub actions + *cibuildwheel*.
- Improved documentation and local development workflow.
- Fixes issue #51 (previously closed as “hard to fix”, but now reopened).
- Completely rewritten native Windows build process, based on libxslt.
- Windows port does not support *ls* (and conversely *listdir()*).

6.5 1.6.5 (2020-09-29)

- No changes from previous release except up-to-date dependencies and some build fixes.
- Fixes issue #118 (awaiting confirmation).

6.6 1.6.3 (2019-10-29)

- First working release on PyPI, based on xmlstarlet-1.6.1 source tarball.

6.7 1.6.2 (2019-10-28)

- Second (failed) release on PyPI, based on XMLStarlet master branch.

6.8 1.6.1 (2019-10-23)

- First (incomplete) release on PyPI, based on XMLStarlet master branch.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search