onixcheck

Release 0.9.5

August 19, 2016

Contents

1 Overview **3**
- 1.1 Onixcheck - Book Trade Metadata Validation . . . 3
- 1.2 Introduction . . . 3
- 1.3 Installation . . . 3
- 1.4 Quickstart . . . 3
- 1.5 Documentation . . . 4
- 1.6 Development . . . 4

2 Installation **5**

3 Usage **7**

4 Reference **9**
- 4.1 onixcheck . . . 9

5 Contributing **11**
- 5.1 Bug reports . . . 11
- 5.2 Documentation improvements . . . 11
- 5.3 Feature requests and feedback . . . 11
- 5.4 Development . . . 11

6 Authors **13**

7 Changelog **15**
- 7.1 0.9.5 (2016-08-19) . . . 15
- 7.2 0.9.4 (2016-07-15) . . . 15
- 7.3 0.9.3 (2016-05-10) . . . 15
- 7.4 0.9.2 (2016-04-11) . . . 15
- 7.5 0.9.1 (2016-04-11) . . . 15
- 7.6 0.9.0 (2016-03-27) . . . 16
- 7.7 0.8.1 (2015-07-23) . . . 16
- 7.8 0.8.0 (2015-07-23) . . . 16
- 7.9 0.4.0 (2015-07-18) . . . 16

8 Indices and tables **17**

Contents:

CHAPTER 1

Overview

1.1 Onixcheck - Book Trade Metadata Validation

1.2 Introduction

ONIX for Books is an international XML standard for the publishing and book trade industry.

`onixcheck` is a Python library and command line tool for validating ONIX metadata. It allows you to validate ONIX versions 2.1 and 3.0 against the official XML Schema.

- Free software: BSD license

1.3 Installation

On Windows you can download the standalone binary command line tool: onixcheck-0.9.4_win.zip

If you have Python or PyPy installed on your system you can do the usual:

```
pip install onixcheck
```

1.4 Quickstart

1.4.1 Command line usage examples

Validate all .xml, .onx, .onix files in current directory:

```
onixcheck
```

Validate a single onix file:

```
onixcheck myonixfile.xml
```

Validate all .xml files in /onixdata and its subdirectories:

```
onixcheck --path /onixdata --ext xml --recursive
```

Show help:

```
onixcheck -h
```

1.4.2 Using onixcheck as a python lib

Simple usage with *onixcheck.validate*:

```
>>> import onixcheck
>>> errors = onixcheck.validate('src/onixcheck/data/invalid_onix3_ref.xml')
>>> print(errors[0].short)
ERROR - SCHEMASV - invalid_onix3_ref.xml:4:0 - Element 'SentDateTime': This element is not expected.
```

errors is either a list of *Message* objects (INVALID file) or an empty list (VALID file)

1.5 Documentation

https://onixcheck.readthedocs.org/

1.6 Development

To run the all tests run:

```
tox
```

Contributions/suggestions are welcome.

CHAPTER 2

Installation

At the command line:

```
pip install onixcheck
```

CHAPTER 3

Usage

To use onixcheck in a project:

```
import onixcheck
```

CHAPTER 4

Reference

4.1 onixcheck

`onixcheck.validate`(*infile*, *schemas=(u'xsd',)*)

Validate an ONIX file.

Parameters

- **infile** (*file or str*) – File or path to file
- **schemas** (*collections.Iterable[str]*) – Iterable with paths to custom valdation profiles

Returns List of *Message* objects (invalid ONIX) or empty list (valid ONIX)

Return type list[Message]

class `onixcheck.models.Message`

A Validation message representing a single error condition.

Parameters

- **level** (*str*) – Error level
- **validator** (*str*) – The validator that raised the error
- **location** (*str*) – Location of error (filename:line:column)
- **message** (*str*) – Description of the error condiction
- **error_type** (*str*) – Type of error

classmethod `from_exception`(*exc*, *filename=u''*)

Parameters

- **exc** (*Exception*) –
- **filename** (*str*) – Optional filename to prefix error location

Return Message

classmethod `from_logentry`(*logentry*, *filename=u''*)

Instanciate Message from lxml LogEntry object

Parameters

- **logentry** (*_LogEntry*) – Validatation error from LXML
- **filename** (*str*) – Optional filename to prefix error location

Return Message

short
Short string representation of message

class onixcheck.models.**OnixFile**(*infile*)
Convenience file object wrapper.

Parameters **infile** (*file or str*) – File or path to file

get_validator(*schema_type=u'xsd'*)
Create a matching validator for the ONIX file.

Return etree._Validator

xml_tree()
Parse the infile with lxml and add the proper namespace if required.

Return etree.ElementTree An lxml ElementTree with proper namespace

class onixcheck.models.**OnixMeta**
Read and detect minimal ONIX file properties needed for validation.

Onix XML files may or may not have *release* and *xmlns* attributes on their root element. OnixMeta.from_file(infile) will detect Onix Version and Style and also patch the root element with the appropriate namespace needed for validation.

Parameters

- **xml_version** (*str*) – XML Version as str ("1.0").
- **xml_encoding** (*str*) – XML Encoding as str ("utf-8").
- **onix_version** (*str*) – Onix Version as string ("2.1" or "3.0")
- **onix_style** (*str*) – Onix Style as str ("short" or "reference")
- **namespaces** (*dict*) – dict of str with namspaces from the root element

classmethod from_file(*infile*)
Contruct OnixMeta from an infile.

Parameters **infile** (*file or str*) – File or Path to file

Return OnixMeta Initialized OnixMeta instance

classmethod from_tree(*tree*)
Construct OnixMeta from an ElementTree.

Parameters **etree.ElementTree** (*tree*) – LXML Parsed ONIX data

Return OnixMeta Initialized OnixMeta instance

onixcheck.utils.**iter_files**(*root*, *exts=None*, *recursive=False*)
Iterate over file paths within root filtered by specified extensions.

Parameters

- **root** (*str*) – Root folder to start collecting files
- **exts** (*iterable*) – Restrict results to given file extensions
- **recursive** (*bool*) – Wether to walk the complete directory tree

Rtype collections.Iterable[str] absolute file paths with given extensions

CHAPTER 5

Contributing

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

5.1 Bug reports

When 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.

5.2 Documentation improvements

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

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/titusz/onixcheck/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 :)

5.4 Development

To set up *onixcheck* for local development:

1. Fork onixcheck on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/onixcheck.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

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

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

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`) [1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- Titusz Pan - http://py7.de

CHAPTER 7

Changelog

7.1 0.9.5 (2016-08-19)

- Update Onix 3 to Code List issue 34
- Remove build artifacts from source distribution

7.2 0.9.4 (2016-07-15)

- Fix issue with windows console output encoding
- Update windows build to pyinstaller 3.1

7.3 0.9.3 (2016-05-10)

- Update ONIX 3.0 Schemas to Revision 3
- Add support for RELAX NG and custom schema validations
- Added validator name to short message output

7.4 0.9.2 (2016-04-11)

- Fix ExtentType in custom validation profile
- Fix false alarm with multiple Price elements in custom profile

7.5 0.9.1 (2016-04-11)

- Inital support for custom validation profiles
- Custom Google Play Books onix 3.0 validation profile

7.6 0.9.0 (2016-03-27)

- Added Python 3.5 testing / support
- Update to ONIX to Code List Issue 32 / 2016-01-24

7.7 0.8.1 (2015-07-23)

- More extensive documentation

7.8 0.8.0 (2015-07-23)

- Added CLI-support for fast directory traversal validation
- Secured XML-Parsing via defusedxml
- Catch basic XML syntax errors
- Windows standalone binary builds

7.9 0.4.0 (2015-07-18)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

F

from_exception() (onixcheck.models.Message class method), 9
from_file() (onixcheck.models.OnixMeta class method), 10
from_logentry() (onixcheck.models.Message class method), 9
from_tree() (onixcheck.models.OnixMeta class method), 10

G

get_validator() (onixcheck.models.OnixFile method), 10

I

iter_files() (in module onixcheck.utils), 10

M

Message (class in onixcheck.models), 9

O

OnixFile (class in onixcheck.models), 10
OnixMeta (class in onixcheck.models), 10

S

short (onixcheck.models.Message attribute), 10

V

validate() (in module onixcheck), 9

X

xml_tree() (onixcheck.models.OnixFile method), 10