wer

Release 0.1.0

June 11, 2016

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Python parser for Microsoft Windows Event Reports (WER)

- Free software: BSD license

1.1 Installation

```
pip install wer
```

1.2 Documentation

https://wer.readthedocs.io/

1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Installation

At the command line:

```
pip install wer
```

CHAPTER 3

Usage

To use wer in a project:

```
import wer
```

CHAPTER 4

Reference

4.1 wer

class wer.**Report**(*node=None*, *context=None*, ***kwargs*)
Windows Error Report

id
Computes the signature of the record, a SHA-512 of significant values

Returns SHa-512 Hex string

4.2 wer.helpers

class wer.helpers.**DateField**(*xpath*)
Custom date field

Uses the custom date mapper

class wer.helpers.**DateMapper**(*format=None*, *normalize=False*)
Custom mapper for WER date

Converts XML timestamp to python datetime.datetime

to_python(*node*)
Converts internal Windows timestamp to Python datetime.datetime

Parameters **node** (*basestring*) – XML node value

Returns Python datetime

Return type datetime.datetime

to_xml(*dt*)
Converts Windows timestamp

Parameters **dt** – date and time to convert

Returns Windows timestamp

Return type int

wer.helpers.**unix_to_windows_timestamp**(*unix_timestamp*)
Converts a Windows timestamp to Unix one

Parameters **unix_timestamp** (*int*) – Unix timestamp

Returns Windows timestamp

Return type int

wer.helpers.**windows_to_unix_timestamp**(*windows_timestamp*)
Converts a Windows timestamp to Unix one

Parameters **windows_timestamp** (*int*) – Windows timestamp

Returns Unix timestamp

Return type int

4.3 wer.schema

class wer.schema.**ApplicationInfo**(*node=None*, *context=None*, ***kwargs*)
ApplicationInfo complex type

company = <eulxml.xmlmap.fields.StringField>
Optional application company :type *string*

name = <eulxml.xmlmap.fields.StringField>
Application name :type *string*

path = <eulxml.xmlmap.fields.StringField>
Application executable path :type *string*

class wer.schema.**DictMixin**
Mixin class in order to export :class:‘eulxml.xmlmap.XmlObject values to a Python dict

to_dict()
Recusively exports object values to a dict

Returns ‘dict‘of values

class wer.schema.**EventInfo**(*node=None*, *context=None*, ***kwargs*)
EventInfo complex type

description = <eulxml.xmlmap.fields.StringField>
Event description :type *string*

name = <eulxml.xmlmap.fields.StringField>
Friendly event name :type *string*

report_type = <eulxml.xmlmap.fields.IntegerField>
Report type :type *int*

time = <wer.helpers.DateField>
Event date :type datetime.datetime

type = <eulxml.xmlmap.fields.StringField>
Event type :type *string*

class wer.schema.**File**(*node=None*, *context=None*, ***kwargs*)
File complex type

name = <eulxml.xmlmap.fields.StringField>
File name :type *string*

type = <eulxml.xmlmap.fields.IntegerField>
File type :type *int*

class wer.schema.**LoaderMixin**
Loading XML into object mixin

classmethod from_file(*file_path*, *validate=True*)
Creates a Python object from a XML file

Parameters

- **file_path** – Path to the XML file
- **validate** (*Boolean*) – XML should be validated against the embedded XSD definition

Returns the Python object

classmethod from_string(*xml_string*, *validate=True*)
Creates a Python object from a XML string

Parameters

- **xml_string** – XML string
- **validate** (*Boolean*) – XML should be validated against the embedded XSD definition

Returns the Python object

class wer.schema.**MachineInfo**(*node=None*, *context=None*, ***kwargs*)
MachineInfo complex type

lcid = <eulxml.xmlmap.fields.IntegerField>
Machine language identifier :type *int*

name = <eulxml.xmlmap.fields.StringField>
Machine name :type *string*

oem = <eulxml.xmlmap.fields.StringField>
Optional machine OEM name :type *string*

os = <eulxml.xmlmap.fields.StringField>
Machine operating system version :type *string*

class wer.schema.**Parameter**(*node=None*, *context=None*, ***kwargs*)
Parameter complex type

id = <eulxml.xmlmap.fields.IntegerField>
Parameter ID :type *int*

name = <eulxml.xmlmap.fields.StringField>
Optional paramneter name :type *string*

value = <eulxml.xmlmap.fields.StringField>
Paramneter value :type *string*

class wer.schema.**Report**(*node=None*, *context=None*, ***kwargs*)
Windows Error Report

application = <eulxml.xmlmap.fields.NodeField>
Application informations :type *wer.schema.ApplicationInfo*

event = <eulxml.xmlmap.fields.NodeField>
Event informations :type *wer.schema.EventInfo*

files = <eulxml.xmlmap.fields.NodeListField>
Event attached files :type list of *wer.schema.File*

id
Computes the signature of the record, a SHA-512 of significant values

Returns SHa-512 Hex string

machine = <eulxml.xmlmap.fields.NodeField>
Machine informations :type *wer.schema.MachineInfo*

parameters = <eulxml.xmlmap.fields.NodeListField>
Event parameters :type list of *wer.schema.Parameter*

secondary_parameters = <eulxml.xmlmap.fields.NodeListField>
Event secondary parameters :type list of *wer.schema.SecondaryParameter*

user = <eulxml.xmlmap.fields.StringField>
User informations :type `wer.schema.UserInfo`

class `wer.schema.`**SecondaryParameter**(*node=None*, *context=None*, ***kwargs*)
Secondary parameter complex type

id = <eulxml.xmlmap.fields.IntegerField>
Parameter ID :type *int*

value = <eulxml.xmlmap.fields.StringField>
Paramneter value :type *string*

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

wer could always use more documentation, whether as part of the official wer 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/gcrahay/python-wer/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 code contributions are welcome :)

5.4 Development

To set up *python-wer* for local development:

1. Fork python-wer (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/python-wer.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

- Gaetan Crahay - https://github.com/gcrahay

CHAPTER 7

Changelog

7.1 0.1.0 (2016-06-05)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

W

W