AIS.py Documentation

Release 0.2.2

Camptocamp

Oct 22, 2018

Contents

AIS.py: a Python interface for the Swisscom All-in Signing Service (aka AIS).

Release v0.2.2.

AIS.py works like this:

```
>>> from AIS import AIS, PDF
>>> client = AIS.AIS('alice', 'a_secret', 'a.crt', 'a.key')
>>> pdf = PDF('source.pdf')
>>> client.sign_one_pdf(pdf)
>>> print(pdf.out_filename)
/tmp/.../T/tmpexkdrlkm.pdf
```

AIS is a webservice for electronic signatures offered by Swisscom. You can check out the corporate page and the reference guide of the service.

To use the webservice you have to send an appropriate digest of the file. The service returns a PKCS#7 detached signature that can be send alongside the original file.

To validate a detached signature, the digest of the original file can be computed again.

In the case of PDF files, the signature is integrated in the PDF itself and it needs to be extracted to be verified.

A complication in that case is that during verification you must be able to compute the same digest that was used to generate the signature with the original file, but the original file is not available anymore, and the signed file has clearly a different digest.

Thus, the procedure is the following:

1. The original PDF file is prepared by adding an empty signature block. This includes a `ByteRange` object.
2. The digest is computed only in the part specified by the `ByteRange`, so it excludes the empty signature.
3. The digest is sent to the AIS webservice.
4. The detached signature is included in the placeholder.

`AIS.py` takes care of all this, delegating point 1 to iText.

CHAPTER 1

Installation

Make sure you have Python 2.7, 3.4, 3.5 or a recent Pypy and Java 7 or later, then:

```
$ pip install AIS.py
```

This will pull Python dependencies, and the Java library is vendored in, so you don't need to install anything other than Python and Java.

CHAPTER 2

Tests

A few tests are found in the `tests/` directory. Integration tests use the real webservice, and HTTP requests/responses are recorded with the vcrpy library as cassettes. This means that you can run all the tests on your machine without real credentials to AIS. The sensible part of the request (i.e. the login and password) is hidden automatically from the cassette file. This also allows the tests to run on Travis CI.

To run the tests locally, enter the directory you cloned and:

```
$ pip install tox
$ tox
```

Tox will automatically create a virtualenv for each Python version, install the package and run the tests.

If you prefer to do this manually for one Python version:

```
$ python -m virtualenv env
$ source env/bin/activate
$ pip install -e .
$ py.test
```

CHAPTER 3

Status

AIS.py is already functional for its main use case, but a few things could be improved:

- Allow to request only a trusted timestamp instead of a signature.
- Allow to choose a different digest algorithm than SHA256.
- Handle second factor authentication in addition to static certificates.
- Implement in Python the generation of an empty signature instead of calling iText through a Java wrapper. Later handling of PDF files is already in Python thanks to the PyPDF2 library that gives a somewhat low level access.
- Fix a few problems with vcrpy that prevent tests from running in Python 3.
- Find a way to check PDF signatures programmatically in the tests.
- Document all parameters and return values in the docstrings (i.e. improve the API reference).

CHAPTER 4

API Reference

This section describes classes and exceptions.

4.1 API

4.1.1 AIS client

class AIS.**AIS**(*customer*, *key_static*, *cert_file*, *cert_key*)

Bases: `object`

Client object holding connection information to the AIS service.

post(*payload*)

Do the post request for this payload and return the signature part of the json response.

Return type dict

sign_batch(*pdfs*)

Sign a batch of files.

sign_one_pdf(*pdf*)

Sign the given pdf file.

4.1.2 PDF file

class AIS.**PDF**(*in_filename*, *prepared=False*)

Bases: `object`

A container for a PDF file to be signed and the signed version.

in_filename = None
Filename of the PDF to be treated.

out_filename = None
Filename of the output, signed PDF.

prepare()
Add an empty signature to self.out_filename.

classmethod prepare_batch(*pdfs*)
Add an empty signature to each of pdfs with only one java call.

prepared = None
Is the PDF prepared with an empty signature?

write_signature(*signature*)
Write the signature in the pdf file

4.1.3 Exceptions

exception AIS.**AISError**
Bases: `exceptions.Exception`

Generic AIS Error.

exception AIS.**AuthenticationFailed**
Bases: `AIS.exceptions.AISError`

Authentication with AIS failed.

This means that AIS returned http://ais.swisscom.ch/1.0/resultminor/AuthenticationFailed

exception AIS.**UnknownAISError**
Bases: `AIS.exceptions.AISError`

Unknown AIS Error.

exception AIS.**AISError**
Bases: `exceptions.Exception`

Generic AIS Error.

exception AIS.**MissingPreparedSignature**
Bases: `AIS.exceptions.AISError`

The PDF file needs to be prepared with an empty signature.

CHAPTER 5

Release History

5.1 0.2.2 (2018-10-22)

- Store the last created request_id on the AIS instance
- Use a proper test matrix on Travis to test various Python releases
- Add Python 3.6 to test matrix

5.2 0.2.1 (2016-06-16)

- Return in batch mode timestamp and revocation information with the signature.
- Fix python3 bugs.
- Refactoring.

5.3 0.2.0 (2016-05-19)

Documentation

- Added sections for introduction, installation, testing, project status, API reference.

5.4 0.1 (2016-05-17)

Initial release. It is possible to start with a batch of pdf files that do not yet have a prepared signature, and sign them.

CHAPTER 6

Contributors

AIS.py is written by:

- Leonardo Pistone (Camptocamp).
- Cyril Gaudin (Camptocamp).

CHAPTER 7

License

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index