pydocusign Documentation

Release 0.13

Benoît Bryon

March 16, 2015

Contents

1 Project status **3**

2 Resources **5**

3 Contents **7**

3.1 Install 7
3.2 Demo 8
3.3 Testing 16
3.4 About pydocusign 17
3.5 Contributing 21

4 Indices and tables **25**

pydocusign is a Python client for DocuSign signature SAAS platform.

CHAPTER 1

Project status

pydocusign has just been created to provide Python bindings for DocuSign's API. The project is not mature yet, but authors already use it! It means that, while API and implementation may change (improve!) a bit, authors do care of the changes.

Also, help is welcome! Feel free to report issues, request features or refactoring!

CHAPTER 2

Resources

- Documentation: https://pydocusign.readthedocs.org
- PyPI page: https://pypi.python.org/pypi/pydocusign
- Bugtracker: https://github.com/novafloss/pydocusign/issues
- Changelog: https://pydocusign.readthedocs.org/en/latest/about/changelog.html
- Roadmap: https://github.com/novafloss/pydocusign/milestones
- Code repository: https://github.com/novafloss/pydocusign
- Continuous integration: https://travis-ci.org/novapost/pydocusign

CHAPTER 3

Contents

3.1 Install

pydocusign is open-source software, published under BSD license. See *License* for details.

If you want to install a development environment, you should go to *Contributing* documentation.

3.1.1 Prerequisites

- Python [1] 2.7. Other versions may be ok but are not part of the test suite at the moment.

3.1.2 As a library

In most cases, you will use *pydocusign* as a dependency of another project. In such a case, you should add `pydocusign` in your main project's requirements. Typically in `setup.py`:

```
from setuptools import setup

setup(
    install_requires=[
        'pydocusign',
        #...
    ]
    # ...
)
```

Then when you install your main project with your favorite package manager (like pip [2]), *pydocusign* will automatically be installed.

3.1.3 Standalone

You can install *pydocusign* with your favorite Python package manager. As an example with pip [2]:

```
pip install pydocusign
```

[1] https://www.python.org
[2] https://pypi.python.org/pypi/pip/

3.1.4 Check

Check *pydocusign* has been installed:

```
python -c "import pydocusign;print(pydocusign.__version__)"
```

You should get *pydocusign*'s version.

References

3.2 Demo

Here is sample code to illustrate *pydocusign* usage.

To run the demo code, you need a development environment. See *Contributing*.

Note: The demo can use the same environment variables as tests. See *Contributing*. If you do not set environment variables, you will be prompted for some configuration.

3.2.1 Embedded signing

```
#!/usr/bin/env python
# coding=utf-8
"""Sample script that demonstrates 'pydocusign' usage for embedded signing.

See also http://iodocs.docusign.com/APIWalkthrough/embeddedSigning

"""
from __future__ import print_function
import os
import sha
import uuid

import pydocusign
from pydocusign.test import fixtures_dir


def prompt(environ_key, description, default):
    try:
        return os.environ[environ_key]
    except KeyError:
        value = raw_input('{description} (default: "{default}"): '.format(
            default=default, description=description))
        if not value:
            return default
        else:
            return value


# Get configuration from environment or prompt the user...
root_url = prompt(
    'DOCUSIGN_ROOT_URL',
    'DocuSign API URL',
    'https://demo.docusign.net/restapi/v2')
```

```
username = prompt(
    'DOCUSIGN_USERNAME',
    'DocuSign API username',
    '')
password = prompt(
    'DOCUSIGN_PASSWORD',
    'DocuSign API password',
    '')
integrator_key = prompt(
    'DOCUSIGN_INTEGRATOR_KEY',
    'DocuSign API integrator key',
    '')
callback_url = prompt(
    'DOCUSIGN_TEST_CALLBACK_URL',
    'Envelope callback URL',
    '')
signer_return_url = prompt(
    'DOCUSIGN_TEST_SIGNER_RETURN_URL',
    'Signer return URL',
    '')


# Create a client.
client = pydocusign.DocuSignClient(
    root_url=root_url,
    username=username,
    password=password,
    integrator_key=integrator_key,
)


# Login. Updates API URLs in client.
print("1. GET /login_information")
login_information = client.login_information()
print("   Received data: {data}".format(data=login_information))


# Prepare list of signers. Ordering matters.
signers = [
    pydocusign.Signer(
        email='jean.francais@example.com',
        name=u'Jean Français',
        recipientId=1,
        clientUserId=str(uuid.uuid4()),  # Something unique in your database.
        tabs=[
            pydocusign.SignHereTab(
                documentId=1,
                pageNumber=1,
                xPosition=100,
                yPosition=100,
            ),
        ],
        emailSubject='Voici un sujet',
        emailBody='Voici un message',
        supportedLanguage='fr',
    ),
    pydocusign.Signer(
        email='paul.english@example.com',
```

```
        name=u'Paul English',
        recipientId=2,
        clientUserId=str(uuid.uuid4()),  # Something unique in your database.
        tabs=[],  # No tabs means user places tabs himself in DocuSign UI.
        emailSubject='Here is a subject',
        emailBody='Here is a message',
        supportedLanguage='en',
    ),
]


# Create envelope with embedded signing.
print("2. POST {account}/envelopes")
event_notification = pydocusign.EventNotification(
    url=callback_url,
)
input_document_path = os.path.join(fixtures_dir(), 'test.pdf')
with open(input_document_path, 'rb') as pdf_file:
    envelope = pydocusign.Envelope(
        documents=[
            pydocusign.Document(
                name='document.pdf',
                documentId=1,
                data=pdf_file,
            ),
        ],
        emailSubject='This is the subject',
        emailBlurb='This is the body',
        eventNotification=event_notification,
        status=pydocusign.Envelope.STATUS_SENT,
        recipients=signers,
    )
    client.create_envelope_from_document(envelope)
print("   Received envelopeId {id}".format(id=envelope.envelopeId))


# Update recipient list of envelope: fetch envelope's ``UserId`` from DocuSign.
print("3. GET {account}/envelopes/{envelopeId}/recipients")
envelope.get_recipients()
print("   Received UserId for recipient 0: {0}".format(
    envelope.recipients[0].userId))
print("   Received UserId for recipient 1: {0}".format(
    envelope.recipients[1].userId))


# Retrieve embedded signing for first recipient.
print("4. Get DocuSign Recipient View")
signing_url = envelope.post_recipient_view(
    routingOrder=1,
    returnUrl=signer_return_url)
print("   Received signing URL for recipient 0: {0}".format(signing_url))
signing_url = envelope.post_recipient_view(
    routingOrder=2,
    returnUrl=signer_return_url)
print("   Received signing URL for recipient 1: {0}".format(signing_url))


# Download signature documents.
```

```
print("5. List signature documents.")
document_list = envelope.get_document_list()
print("   Received document list: {0}".format(document_list))
print("6. Download document from DocuSign.")
document = envelope.get_document(document_list[0]['documentId'])
document_sha = sha.new(document.read()).hexdigest()
print("   Document SHA1: {0}".format(document_sha))
document.close()
print("7. Download signature certificate from DocuSign.")
document = envelope.get_certificate()
document_sha = sha.new(document.read()).hexdigest()
print("   Certificate SHA1: {0}".format(document_sha))
document.close()
```

You can run this code with:

```
python demo/embeddedsigning.py
```

3.2.2 Creating envelope using template

```
#!/usr/bin/env python
# coding=utf-8
"""Sample script that demonstrates `pydocusign` usage for template signing.

See also http://iodocs.docusign.com/APIWalkthrough/requestSignatureFromTemplate

"""
from __future__ import print_function
import os
import sha
import uuid

import pydocusign


def prompt(environ_key, description, default):
    try:
        return os.environ[environ_key]
    except KeyError:
        value = raw_input('{description} (default: "{default}"): '.format(
            default=default, description=description))
        if not value:
            return default
        else:
            return value


# Get configuration from environment or prompt the user...
root_url = prompt(
    'DOCUSIGN_ROOT_URL',
    'DocuSign API URL',
    'https://demo.docusign.net/restapi/v2')
username = prompt(
    'DOCUSIGN_USERNAME',
    'DocuSign API username',
    '')
password = prompt(
```

```
    'DOCUSIGN_PASSWORD',
    'DocuSign API password',
    '')
integrator_key = prompt(
    'DOCUSIGN_INTEGRATOR_KEY',
    'DocuSign API integrator key',
    '')
template_id = prompt(
    'DOCUSIGN_TEST_TEMPLATE_ID',
    'DocuSign Template ID',
    '')
callback_url = prompt(
    'DOCUSIGN_TEST_CALLBACK_URL',
    'Envelope callback URL',
    '')
signer_return_url = prompt(
    'DOCUSIGN_TEST_SIGNER_RETURN_URL',
    'Signer return URL',
    '')


# Create a client.
client = pydocusign.DocuSignClient(
    root_url=root_url,
    username=username,
    password=password,
    integrator_key=integrator_key,
)


# Login. Updates API URLs in client.
print("1. GET /login_information")
login_information = client.login_information()
print("   Received data: {data}".format(data=login_information))


# Get the template definition
print("2. GET {account}/templates/{templateId}")
template_definition = client.get_template(template_id)
assert template_definition['recipients']['signers'][0][
    'roleName'] == 'employee'
assert template_definition['recipients']['signers'][0][
    'routingOrder'] == '1'
assert template_definition['recipients']['signers'][1][
    'roleName'] == 'employer'
assert template_definition['recipients']['signers'][1][
    'routingOrder'] == '1'  # Same routingOrder, important for testing


# Prepare list of roles. Ordering matters
roles = [
    pydocusign.Role(
        email='jean.francais@example.com',
        name=u'Jean Français',
        roleName='employer',
        clientUserId=str(uuid.uuid4()),  # Something unique in your database.
    ),
    pydocusign.Role(
```

```
        email='paul.english@example.com',
        name=u'Paul English',
        roleName='employee',
        clientUserId=str(uuid.uuid4()),  # Something unique in your database.
    ),
]


# Create envelope with embedded signing.
print("3. POST {account}/envelopes")
event_notification = pydocusign.EventNotification(
    url=callback_url,
)
envelope = pydocusign.Envelope(
    emailSubject='This is the subject',
    emailBlurb='This is the body',
    eventNotification=event_notification,
    status=pydocusign.Envelope.STATUS_SENT,
    templateId=template_id,
    templateRoles=roles,
)
client.create_envelope_from_template(envelope)
print("   Received envelopeId {id}".format(id=envelope.envelopeId))


# Update recipient list of envelope: fetch envelope's ``UserId`` from DocuSign.
print("4. GET {account}/envelopes/{envelopeId}/recipients")
envelope.get_recipients()
print("   Received UserId for recipient 0: {0}".format(
    envelope.templateRoles[0].userId))
print("   Received UserId for recipient 1: {0}".format(
    envelope.templateRoles[1].userId))


# Retrieve template signing for first recipient.
print("5. Get DocuSign Recipient View")
signing_url = envelope.post_recipient_view(
    routingOrder=1,
    returnUrl=signer_return_url)
print("   Received signing URL for recipient 0: {0}".format(signing_url))
signing_url = envelope.post_recipient_view(
    routingOrder=2,
    returnUrl=signer_return_url)
print("   Received signing URL for recipient 1: {0}".format(signing_url))


# Download signature documents.
print("6. List signature documents.")
document_list = envelope.get_document_list()
print("   Received document list: {0}".format(document_list))
print("7. Download document from DocuSign.")
document = envelope.get_document(document_list[0]['documentId'])
document_sha = sha.new(document.read()).hexdigest()
print("   Document SHA1: {0}".format(document_sha))
document.close()
print("8. Download signature certificate from DocuSign.")
document = envelope.get_certificate()
document_sha = sha.new(document.read()).hexdigest()
```

```
print("   Certificate SHA1: {0}".format(document_sha))
document.close()
```

You can run this code with:

```
python demo/templates.py
```

3.2.3 Managing accounts

```
#!/usr/bin/env python
# coding=utf-8
"""Sample script that demonstrates `pydocusign` usage for managing accounts.

See also http://iodocs.docusign.com/#tabEndpoint6

"""
import os

import pydocusign


def prompt(environ_key, description, default):
    try:
        return os.environ[environ_key]
    except KeyError:
        value = raw_input('{description} (default: "{default}"): '.format(
            default=default, description=description))
        if not value:
            return default
        else:
            return value


# Get configuration from environment or prompt the user...
root_url = prompt(
    'DOCUSIGN_ROOT_URL',
    'DocuSign API URL',
    'https://demo.docusign.net/restapi/v2')
username = prompt(
    'DOCUSIGN_USERNAME',
    'DocuSign API username',
    '')
password = prompt(
    'DOCUSIGN_PASSWORD',
    'DocuSign API password',
    '')
integrator_key = prompt(
    'DOCUSIGN_INTEGRATOR_KEY',
    'DocuSign API integrator key',
    '')
distributor_code = prompt(
    'DOCUSIGN_TEST_DISTRIBUTOR_CODE',
    'DocuSign API distributor code',
    '')
distributor_password = prompt(
    'DOCUSIGN_TEST_DISTRIBUTOR_PASSWORD',
    'DocuSign API distributor password',
```

```
    '')
callback_url = prompt(
    'DOCUSIGN_TEST_CALLBACK_URL',
    'Envelope callback URL',
    '')
signer_return_url = prompt(
    'DOCUSIGN_TEST_SIGNER_RETURN_URL',
    'Signer return URL',
    '')


# Create a client.
client = pydocusign.DocuSignClient(
    root_url=root_url,
    username=username,
    password=password,
    integrator_key=integrator_key,
)


# Login. Updates API URLs in client.
print("1. GET /login_information")
login_information = client.login_information()
print("   Received data: {data}".format(data=login_information))


# Get main account information.
print("2. GET /accounts/{accountId}".format(accountId=client.account_id))
account_information = client.get_account_information(client.account_id)
print("   Received data: {data}".format(data=account_information))


# Create a subsidiary account.
print("3. POST /accounts")
account_email = "benoit.bryon+pydocusign-test@novapost.fr"
account_password = "notasecret"
account_input = {
    "accountName": "Pydocusign Test Account",
    "accountSettings": [],
    "addressInformation": {
        "address1": "32 rue de Paradis",
        "address2": "",
        "city": "Paris",
        "country": "France",
        "fax": "",
        "phone": "",
        "postalCode": "75010",
        "state": "",
    },
    "creditCardInformation": None,
    "distributorCode": distributor_code,
    "distributorPassword": distributor_password,
    "initialUser": {
        "email": account_email,
        "firstName": "John",
        "lastName": "Doe",
        "middleName": "Jean",
        "password": account_password,
```

```
        "suffixName": "",
        "title": "M",
        "userName": account_email,
        "userSettings": [],
    },
    "planInformation": {
        "planId": "64c9f146-91d8-4137-8367-fd3f3a23ef76",
    },
    "referralInformation": None,
    "socialAccountInformation": None,
}
account_data = client.post_account(account_input)
print("   Received data: {data}".format(data=account_data))

raw_input("Please activate account {} and type RET".format(account_email))

# Get subsidiary account information.
print("4. GET /accounts/{accountId}".format(
    accountId=account_data['accountId']))
account_information = client.get_account_information(client.account_id)
print("   Received data: {data}".format(data=account_information))


# In order to delete subsidiary account, we have to log in with this account.
subsidiary_client = pydocusign.DocuSignClient(
    root_url=root_url,
    username=account_data['userId'],
    password=account_password,
    integrator_key=integrator_key,  # Use the same key as main account!
)

# Login. Updates API URLs in client.
print("5. LOGIN WITH SUBSIDIARY ACCOUNT")
account_login_information = subsidiary_client.login_information()
print("   Received data: {data}".format(data=account_login_information))


# Delete subsidiary account.
print("6. DELETE /accounts/{accountId}".format(
    accountId=subsidiary_client.account_id))
deleted = subsidiary_client.delete_account(subsidiary_client.account_id)
print("   Received data: {data}".format(data=deleted))
```

You can run this code with:

```
python demo/accounts.py
```

3.3 Testing

pydocusign provides some tools to help you test your applications:

- `post_notification_callback()`
- `generate_notification_callback_body()`

3.3.1 post_notification_callback

Note: Uses third-party diecutter [3] online file generation service. See `generate_notification_callback_body()` for details.

3.3.2 generate_notification_callback_body

Note: This function uses third-party diecutter [1] online file generation service. This allows *pydocusign* to use templates without having template engines (Jinja2) as dependencies, since this callback notification feature is for test purpose only. At runtime, *pydocusign* does not require Jinja2.

Available templates are the ones in `pydocusign/templates` folder of pydocusign's code repository [4].

References

3.4 About pydocusign

This section is about the *pydocusign* project itself.

3.4.1 Vision

pydocusign is a Python client for DocuSign [5] signature SAAS platform.

As *pydocusign* author, I basically needed Python bindings for DocuSign API, but I did not find one either on https://pypi.python.org/pypi?%3Aaction=search&term=docusign or https://www.docusign.com/developer-center/helper-libraries. So initiating a new one looked like a fair start. I published it under BSD so that it can be used, improved and maintained by a community.

So, **feel free to report issues, request features or refactoring!** See *Contributing* for details.

Notes & references

3.4.2 Alternatives and related projects

This document presents other projects that provide similar or complementary functionalities. It focuses on differences or relationships with *pydocusign*.

DocuSign's helper libraries

There are other libraries related to DocuSign [6] service. Check https://www.docusign.com/developer-center/helper-libraries

[3] http://diecutter.io
[4] https://github.com/novafloss/pydocusign/
[5] https://www.docusign.com
[6] https://www.docusign.com/

django-docusign

django-docusign [7] integrates *pydocusign* in *Django*. It is built on top of django-anysign [8] generic API.

References

3.4.3 License

3.4.4 Authors & contributors

Maintainer: the PeopleDoc [9] team: https://github.com/novafloss/

Developers: https://github.com/novafloss/pydocusign/graphs/contributors

Notes & references

3.4.5 Changelog

This document describes changes between each past release. For information about future releases, check milestones [10] and *Vision*.

[7] https://github.com/novafloss/django-docusign/
[8] https://github.com/novafloss/django-anysign/
[9] http://www.people-doc.com
[10] https://github.com/novafloss/pydocusign/milestones

0.13 (2015-03-16)

Work around signer model: `routingOrder`, `recipientId` and `clientUserId`.

- Feature #51 - Added routingOrder attribute to Signer model. Defaults to 0.
- Feature #50 - `DocuSignCallbackParser.recipients_events` returns `recipientId` and `clientUserId` (former `recipient` value gets deprecated).
- Bug #49 - Improved update of envelope's recipients via `Envelope.get_recipients()`. Fixes potential `KeyError` from `models.Envelope.get_recipients()`.
- Refactoring #52 - Project's repository moved to https://github.com/novafloss/pydocusign

0.12 (2015-01-30)

Workaround environment variables for configuration, and bugfix.

- Feature #45 - `DocuSignClient` can be configured using environment variables. Supported environment variables are: `DOCUSIGN_ROOT_URL`, `DOCUSIGN_USERNAME`, `DOCUSIGN_PASSWORD`, `DOCUSIGN_INTEGRATOR_KEY`, `DOCUSIGN_ACCOUNT_ID`, `DOCUSIGN_APP_TOKEN` and `DOCUSIGN_TIMEOUT`.

  Obviously, arguments you explicitely pass to the constructor have priority over environment variables.

  Demo scripts were using environment variables with prefix `PYDOCUSIGN_TEST_`. The environment variables have been renamed with either `DOCUSIGN_` (the ones in the list above) or `DOCUSIGN_TEST_` prefix.
- Bug #44 - Using http://diecutter.io in `pydocusign.test.generate_notification_callback_body()`. Secure URL was broken because of SSL certificate.

0.11 (2015-01-23)

- Bug #41 - demo/accounts.py cannot be completed automatically: user interaction is required to activate account. Added an interactive prompt in demo. Removed demo/accounts.py from tests.
- Feature #34 - DocuSignClient accepts `timeout` option (defaults to 30 seconds). This timeout is used as connection timeout when performing HTTP requests to DocuSign API.

0.10 (2014-12-03)

Features around DocuSign accounts.

- Feature #28 - `pydocusign.DocuSignClient` can create, read and delete accounts.

0.9 (2014-11-26)

New feature around envelope creation.

- Feature #29 - `pydocusign.DocuSignClient` can create envelope using template. Added method `create_envelope_from_template()`. Check the `templatesigning.py` demo in documentation. When running tests or demo, you need an additional `DOCUSIGN_TEST_TEMPLATE_ID` environment variable, which is the UUID of a DocuSign template having 2 signers with routing order set to "1".

0.8 (2014-11-13)

Feature around internationalization.

- Feature #26 - `pydocusign.Signer` model accepts new optional arguments: `emailSubject`, `emailBody` and `supportedLanguage`. They are used to setup DocuSign's `emailNotification` for recipient, which makes it possible to change default language shown in DocuSign user interface for each recipient.

0.7.2 (2014-10-22)

Bugfixes.

- Bug #23 - Using pseudo-constants of `pydocusign.Envelope` and `pydocusign.Recipient` instead of raw strings in `DocuSignCallbackParser`. As a consequence, parser's `recipient_events`, `envelope_events` and `events` properties also use values from pseudo-constants. They use CamelCase (they used to be all lowercase).
- Bug #22 - Fixed `DeclineReason` in notification callback template.

0.7.1 (2014-10-17)

Improved retro-compatibility of release 0.7.

- Bug #20 - Restored pseudo-constants in `Envelope` model. They were removed in version 0.7, but it appeared they are useful, and probably the simplest way to access them, since `pydocusign.Envelope` is part of the main API, whereas `pydocusign.models` is not.

  Also:

  - added keys/values to `Envelope`'s pseudo-constants.
  - registered pseudo-constants in `Recipient` model.

0.7 (2014-10-16)

Testing tools around notification callbacks.

- Features #14 and #16 - `pydocusign.test` provides utilities to test notification callbacks: generate body content for notification requests, post fake notification requests to a custom URL.
- Added pseudo-constants in `pydocusign.models`, so that you can use them instead of strings. As an example, you'd better use `pydocusign.models.ENVELOPE_STATUS_SENT` instead of `"Sent"`.

  Removed `STATUS_SENT` and `STATUS_DRAFT` from `Envelope` model.

0.6 (2014-10-08)

Improvements around event notifications.

- Feature #15 - On envelope creation, register callback URL for recipient events.
- Feature #17 - `pydocusign.DocuSignCallbackParser` can extract data from DocuSign's event notification callbacks.

0.5 (2014-09-26)

Minor feature around document download.

- Feature #12: DocuSignClient.get_envelope_document() returns a file-like object with a close() method.

0.4 (2014-09-17)

Feature: download envelope's documents.

- Feature #10 - Introduced methods for client and envelope instances to get the list of envelope's documents, and download documents.

0.3 (2014-09-12)

Minor API simplification.

- Feature #7 - Since most methods of `DocuSignClient` require `account_url` attribute, they can initialize it automatically with a call to `login_information()`.
- Bug #6 - Added "URL" and "license" fields to Python package metadata (were missing).

0.2 (2014-09-05)

Integrate more DocuSign features.

- Feature #1 - Envelopes can be created with `eventNotification` argument, which allows to setup API callbacks on signature updates.
- Feature #3 - Added support for "approve tabs", which allow signer to sign a document without adding visible text/graphic to the document.
- Feature #5 - DocuSignClient.login_information raises DocuSignException and emits logs on error.

0.1 (2014-07-30)

Initial release.

- Introduced `DocuSignClient` client and models. Initial features around embedded signing workflow:
  - login
  - create envelope
  - get envelope's recipients (read envelope)
  - post recipient view (get embedded signing URL)

Notes & references

3.5 Contributing

This document provides guidelines for people who want to contribute to the *pydocusign* project.

3.5.1 Create tickets

Please use pydocusign bugtracker [11] **before** starting some work:

- check if the bug or feature request has already been filed. It may have been answered too!
- else create a new ticket.
- if you plan to contribute, tell us, so that we are given an opportunity to give feedback as soon as possible.
- Then, in your commit messages, reference the ticket with some `refs #TICKET-ID` syntax.

3.5.2 Use topic branches

- Work in branches.
- Prefix your branch with the ticket ID corresponding to the issue. As an example, if you are working on ticket #23 which is about contribute documentation, name your branch like `23-contribute-doc`.
- If you work in a development branch and want to refresh it with changes from master, please rebase [12] or merge-based rebase [13], i.e. do not merge master.

3.5.3 Fork, clone

Clone *pydocusign* repository (adapt to use your own fork):

```
git clone git@github.com:novapost/pydocusign.git
cd pydocusign/
```

3.5.4 Usual actions

The *Makefile* is the reference card for usual actions in development environment:

- Install development toolkit with pip [14]: `make develop`.
- Run tests with tox [15]: `make test`.
- Build documentation: `make documentation`. It builds Sphinx [16] documentation in *var/docs/html/index.html*.
- Release *pydocusign* project with zest.releaser [17]: `make release`.
- Cleanup local repository: `make clean`, `make distclean` and `make maintainer-clean`.

See also `make help`.

3.5.5 Create an account on DocuSign platform

In order to run the tests or to use *pydocusign* features, you will need an account on DocuSign platform. For testing purpose, a developer account is fine: see https://demo.docusign.net/.

[11] https://github.com/novafloss/pydocusign/issues
[12] http://git-scm.com/book/en/Git-Branching-Rebasing
[13] http://tech.novapost.fr/psycho-rebasing-en.html
[14] https://pypi.python.org/pypi/pip/
[15] https://pypi.python.org/pypi/tox/
[16] https://pypi.python.org/pypi/Sphinx/
[17] https://pypi.python.org/pypi/zest.releaser/

3.5.6 Use private credentials to run the tests

The test suite contains several integration tests, so it requires valid DocuSign account credentials. The test suite reads environment variables to get the setup. Here is an example to run the tests:

```
DOCUSIGN_ROOT_URL='https://demo.docusign.net/restapi/v2' \
DOCUSIGN_USERNAME='your-username' \
DOCUSIGN_PASSWORD='your-password' \
DOCUSIGN_INTEGRATOR_KEY='your-integrator-key' \
DOCUSIGN_TEST_TEMPLATE_ID='UUID-of-your-docusign-template' \
DOCUSIGN_TEST_SIGNER_RETURN_URL='http://example.com/signer-return/' \
DOCUSIGN_TEST_CALLBACK_URL='http://example.com/callback/' \
make test
```

Note: Environment variables with prefix `DOCUSIGN_TEST_` may be deprecated in future releases: they are only used in tests.

Whereas environment variables with prefix `DOCUSIGN_` are used both for tests and normal use: `pydocusign.DocuSignClient` use them as default values.

Notes & references

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*