

---

# **pyEchosign Documentation**

*Release 1.0.1*

**Jens Astrup**

**Nov 07, 2017**



---

# Contents

---

<b>1</b>	<b>About</b>	<b>1</b>
1.1	Documentation . . . . .	1
1.2	Maintained on GitLab . . . . .	1
<b>2</b>	<b>Notes</b>	<b>3</b>
2.1	JSON Deserialization . . . . .	3
2.2	Internal Methods and Classes . . . . .	3
2.2.1	Quickstart . . . . .	3
2.2.2	EchosignAccount . . . . .	5
2.2.3	Agreements . . . . .	5
2.2.4	Documents . . . . .	9
2.2.5	Users . . . . .	10
<b>3</b>	<b>Indices and tables</b>	<b>13</b>



A Python module for connecting to the Adobe Echosign REST API, without the hassle of dealing with the JSON formatting for requests/responses and the REST endpoints and their varying requirements

## 1.1 Documentation

The most up to date documentation can be found on [pyEchosign's RTD page](#).

## 1.2 Maintained on GitLab

This project is maintained on [GitLab](#) and mirrored to [GitHub](#). Issues opened on the latter are still addressed.



## 2.1 JSON Deserialization

Most classes contain two methods to facilitate the process of receiving JSON from the REST API and turning that into Python classes. One, `json_to_X()` will handle the JSON formatting for a single instance, while the second - `json_to_Xs()` processes JSON for multiple instances. Generally, the latter is simply returning a list comprehension that calls the former.

While this is primarily useful for internal purposes - every method retrieving an Agreement from the API will call `Agreement.json_to_agreement()` for example - the methods are not private and available for use. Any changes to their interface will only be made following deprecation warnings.

## 2.2 Internal Methods and Classes

All protected and private methods; and any classes, functions, or methods found under `pyEchosign.utils` are subject to change without deprecation warnings however.

### 2.2.1 Quickstart

#### Account Instantiation

In order to interact with the API, you need to create an instance of `EchosignAccount`, which will allow you to send/retrieve agreements, documents, etc.

Note that this module does not handle the OAuth process, gaining an access token must be done outside of this module.

```
from pyEchosign import *

token = 'My Access Token'
account = EchosignAccount(token)
```

```
# When the access token is refreshed
account.access_token = 'new access token'
```

### Sending Agreements

```
from pyEchosign import *

account = EchosignAccount('')

agreement = Agreement(account, name='My Agreement')

# MIME type is optional - it will be inferred from the file extension by Adobe if not
# provided
file = TransientDocument(account, 'To be Signed.pdf', 'some bytes', 'application/pdf')
agreement.files = [file]

# If your document utilizes merge fields, you can specify which fields should be
# merged with what values.
# If you have no idea what this is, just ignore it - it's not required :)
merge_fields = [dict(field_name='some_field_name', default_value='some default value
#')]

recipients = [Recipient('dude@gmail.com'), Recipient('i_sign_second@gmail.com')]

agreement.send(recipients, merge_fields=merge_fields, ccs=['please_cc_me@gmail.com'])
```

### Retrieving Agreements

This method retrieves the most recent 9 agreements from the account. A query can be specified to search through the account's agreements.

```
from pyEchosign import *

account = EchosignAccount('')

agreements = account.get_agreements()
agreements[0]
>>> Some Agreement Title

agreements = account.get_agreements('query')
agreements[0]
>>> 'Some Agreement Title with the Word query In It'
```

### Manage Agreements

You can either cancel an agreement, which will make it still visible on the user's Manage page, or delete it which removes it entirely.

```
from pyEchosign import *

account = EchosignAccount('')

agreements = account.get_agreements()
```



```

agreement = agreements[0]

print(agreement.status)
>>> Agreement.Status.OUT_FOR_SIGNATURE

agreement.cancel()
# Still visible, but no longer waiting for signature

print(agreement.status)
>>> Agreement.Status.RECALLED

agreement.delete()
# and now it's gone

```

## 2.2.2 EchosignAccount

**class EchosignAccount** (*access\_token*, *\*\*kwargs*)

Saves OAuth Information for connecting to Echosign

**access\_token**

The OAuth Access token to use for authenticating to Echosign

**user\_id**

The ID of the user to specify as the API caller, if not provided the caller is inferred from the token

**user\_email**

The email of the user to specify as the API caller, if not provided the caller is inferred from the token

**api\_access\_point**

The API endpoint used as a base for all API calls

**get\_agreements** (*query=None*)

Gets all agreements for the EchosignAccount

**Keyword Arguments** **query** – (str) A search query to filter results by

Returns: A list of *Agreement* objects

**get\_library\_documents** ()

Gets all Library Documents for the EchosignAccount

Returns: A list of *Agreement* objects

**headers** (*content\_type='application/json'*)

Return headers using account information

**Parameters** **content\_type** – The Content-Type to use in the request headers. Defaults to application/json

Returns: A dict of headers

## 2.2.3 Agreements

**class Agreement** (*account*, *\*\*kwargs*)

Represents either a created agreement in Echosign, or one built in Python which can be sent through, and created in Echosign.

**Parameters** **account** (*EchosignAccount*) – An instance of *EchosignAccount*. All Agreement actions will be conducted under this account.

### Keyword Arguments

- **fully\_retrieved** (*bool*) – Whether or not the agreement has all information retrieved, or if only the basic information was pulled (such as when getting all agreements instead of requesting the specific agreement)
- **echosign\_id** (*str*) – The ID assigned to the agreement by Echosign, used to identify the agreement via the API
- **name** (*str*) – The name of the document as specified by the sender
- **status** (*Agreement.Status*) – The current status of the document (OUT\_FOR\_SIGNATURE, SIGNED, APPROVED, etc)
- **users** (*list[DisplayUser]*) – The users associated with this agreement, represented by *EchosignAccount*
- **files** (*list*) – A list of *TransientDocument* instances which will become the documents within the agreement. This information is not provided when retrieving agreements from Echosign.

#### **account**

*EchosignAccount* – An instance of *EchosignAccount*. All Agreement actions will be conducted under this account.

#### **fully\_retrieved**

*bool* – Whether or not the agreement has all information retrieved, or if only the basic information was pulled (such as when getting all agreements instead of requesting the specific agreement)

#### **echosign\_id**

*str* – The ID assigned to the agreement by Echosign, used to identify the agreement via the API

#### **name**

*str* – The name of the document as specified by the sender

#### **status**

*Agreement.Status* – The current status of the document (OUT\_FOR\_SIGNATURE, SIGNED, APPROVED, etc)

#### **users**

*list[DisplayUser]* – The users associated with this agreement, represented by *EchosignAccount*

#### **files**

*list* – A list of *TransientDocument* instances which will become the documents within the agreement. This information is not provided when retrieving agreements from Echosign.

#### **class SignatureFlow**

**PARALLEL** = 'PARALLEL'

**SENDER\_SIGNS\_ONLY** = 'SENDER\_SIGNS\_ONLY'

**SEQUENTIAL** = 'SEQUENTIAL'

#### **class Status**

Possible status of agreements

---

**Note:** Echosign provides 'WAITING\_FOR\_FAXIN' in their API documentation, so pyEchosign has also included 'WAITING\_FOR\_FAXING' in case that's just a typo in their documentation. Once it's determined which is used, the other will be removed.

---

ACCEPTED = 'ACCEPTED'  
APPROVED = 'APPROVED'  
ARCHIVED = 'ARCHIVED'  
DELIVERED = 'DELIVERED'  
EXPIRED = 'EXPIRED'  
FORM = 'FORM'  
FORM\_FILLED = 'FORM\_FILLED'  
OTHER = 'OTHER'  
OUT\_FOR\_ACCEPTANCE = 'OUT\_FOR\_ACCEPTANCE'  
OUT\_FOR\_APPROVAL = 'OUT\_FOR\_APPROVAL'  
OUT\_FOR\_DELIVERY = 'OUT\_FOR\_DELIVERY'  
OUT\_FOR\_FORM\_FILLING = 'OUT\_FOR\_FORM\_FILLING'  
OUT\_FOR\_SIGNATURE = 'OUT\_FOR\_SIGNATURE'  
RECALLED = 'RECALLED'  
SIGNED = 'SIGNED'  
WAITING\_FOR\_AUTHORIZING = 'WAITING\_FOR\_AUTHORIZING'  
WAITING\_FOR\_FAXIN = 'WAITING\_FOR\_FAXIN'  
WAITING\_FOR\_FAXING = 'WAITING\_FOR\_FAXING'  
WAITING\_FOR\_MY\_ACCEPTANCE = 'WAITING\_FOR\_MY\_ACCEPTANCE'  
WAITING\_FOR\_MY\_ACKNOWLEDGEMENT = 'WAITING\_FOR\_MY\_ACKNOWLEDGEMENT'  
WAITING\_FOR\_MY\_APPROVAL = 'WAITING\_FOR\_MY\_APPROVAL'  
WAITING\_FOR\_MY\_DELEGATION = 'WAITING\_FOR\_MY\_DELEGATION'  
WAITING\_FOR\_MY\_FORM\_FILLING = 'WAITING\_FOR\_MY\_FORM\_FILLING'  
WAITING\_FOR\_MY\_SIGNATURE = 'WAITING\_FOR\_MY\_SIGNATURE'  
WIDGET = 'WIDGET'

**audit\_trail\_file**

The PDF file of the audit trail.

**cancel ()**

Cancels the agreement on Echosign. Agreement will still be visible in the Manage page.

**combined\_document**

The PDF file containing all documents within this agreement.

**delete ()**

Deletes the agreement on Echosign. Agreement will not be visible in the Manage page.

---

**Note:** This action requires the 'agreement\_retention' scope, which doesn't appear to be actually available via OAuth

---

**documents**

Retrieve the *AgreementDocuments* associated with this agreement. If the files have not already been retrieved, this will result in an additional request to the API.

Returns: A list of *AgreementDocument*

**get\_form\_data()**

Retrieves the form data for this agreement as CSV.

Returns: StringIO

**get\_signing\_urls()**

Associate the signing URLs for this agreement with its *recipients*

**classmethod json\_to\_agreement(account, json\_data)****classmethod json\_to\_agreements(account, json\_data)**

**send**(*recipients, agreement\_name=None, ccs=None, days\_until\_signing\_deadline=0, external\_id='', signature\_flow='SEQUENTIAL', message='', merge\_fields=None*)  
Sends this agreement to Echosign for signature

**Parameters**

- **agreement\_name** – A string for the document name which will appear in the Echosign Manage page, the email to recipients, etc. Defaults to the name for the Agreement.
- **recipients** – A list of *Users*. The order which they are provided in the list determines the order in which they sign.
- **ccs** – (optional) A list of email addresses to be CC'd on the Echosign agreement emails (document sent, document fully signed, etc)
- **days\_until\_signing\_deadline** – (optional) “The number of days that remain before the document expires. You cannot sign the document after it expires” Defaults to 0, for no expiration.
- **external\_id** – (optional) “A unique identifier for your transaction... You can use the ExternalID to search for your transaction through [the] API”
- **signature\_flow** – (optional) (SignatureFlow): The routing style of this agreement, defaults to Sequential.
- **merge\_fields** – (optional) A list of dictionaries, with each one providing the ‘field\_name’ and ‘default\_value’ keys. The field name maps to the field on the document, and the default value is what will be placed inside.
- **message** – (optional) A message which will be displayed to recipients of the agreement

**Returns**

A namedtuple representing the information received back from the API. Contains the following attributes

**agreement\_id** “The unique identifier that can be used to query status and download signed documents”

**embedded\_code** “Javascript snippet suitable for an embedded page taking a user to a URL”

**expiration** “Expiration date for autologin. This is based on the user setting, API\_AUTO\_LOGIN\_LIFETIME”

**url** “Standalone URL to direct end users to”

**Raises** `ApiError` – If the API returns an error, such as a 403. The exact response from the API is provided.

**send\_reminder** (*comment*='')

Send a reminder for an agreement to the participants.

**Parameters** **comment** – An optional comment that will be sent with the reminder

## 2.2.4 Documents

### Agreement Documents

**class AgreementDocument** (*echosign\_id, mime\_type, name, page\_count, supporting\_document=False, field\_name=None*)

Represents a document used in an Agreement.

**echosign\_id**

The ID of the Document which can be used to retrieve its file stream

**mime\_type**

The MIME type of the document

**name**

The name of the document

**page\_count**

The number of pages in the document

**supporting\_document**

Whether or not this document is a “supporting document” as specified by the API

**field\_name**

If a supporting document, what the name is of the supporting document field

### Library Documents

**class LibraryDocument** (*account, echosign\_id, template\_type, name, modified\_date, scope*)

Represents a Library Document in Echosign. When pulling all Library Documents, only the echosign\_id, template\_type, modified\_date, name, and scope are available. Accessing all other attributes results in an HTTP request to pull the full document information.

**account**

*EchosignAccount* – An instance of *EchosignAccount*. All Agreement actions will be conducted under this account.

**echosign\_id**

*str* – The ID for this document in Echosign

**document**

*bool* – If this LibraryDocument is a document in Echosign

**form\_field\_layer**

*bool* – If this LibraryDocument is a form field layer

**modified\_date**

*datetime* – The day on which the LibraryDocument was last modified

**name**

*str* – The name of the LibraryDocument in Echosign

**scope**

*str* – The visibility of this LibraryDocument, either ‘PERSONAL’, ‘SHARED’, or ‘GLOBAL’

**GLOBAL** = 'GLOBAL'

**PERSONAL** = 'PERSONAL'

**SHARED** = 'SHARED'

**audit\_trail\_file**

The PDF file of the audit for this Library Document.

**delete** ()

Deletes the LibraryDocument from Echosign. It will not be visible on the Manage page.

**classmethod json\_to\_agreement** (*account, json\_data*)

**classmethod json\_to\_agreements** (*account, json\_data*)

**locale**

**retrieve\_complete\_document** ()

Retrieves the remaining data for the LibraryDocument, such as locale, status, and security options.

**scope** = None

## Transient Documents

**class TransientDocument** (*account, file\_name, file, mime\_type=None*)

A document which can be used in Agreements - is deleted by Echosign after 7 days. The TransientDocument is created in Echosign on instantiation.

### Parameters

- **account** – The *EchosignAccount* to be associated with this document
- **file\_name** (*str*) – The name of the file
- **file** – The actual file object to upload to Echosign, accepts a stream of bytes.
- **mime\_type** – (optional) The MIME type of the file. Echosign will infer the type from the file extension if not provided.

**file\_name**

The name of the file

**file**

The actual file object to upload to Echosign

**mime\_type**

The MIME type of the file

**document\_id**

The ID provided by Echosign, used to reference it in creating agreements

**expiration\_date**

The date Echosign will delete this document (not provided by Echosign, calculated for convenience)

## 2.2.5 Users

**class User** (*email, \*\*kwargs*)

Bases: `object`

Maps to the DisplayUserInfo provided by Echosign for agreements fetched in bulk. Provides additional attributes to facilitate sending documents to recipients, such as Security Options.

**agreement**

*Agreement* – The *Agreement* to be associated with this User

**authentication\_method**

*str* – A “The authentication method for the recipients to have access to view and sign the document” (Echosign API Docs). Available options are ‘NONE’ (string), ‘INHERITED\_FROM\_DOCUMENT’ or ‘PASSWORD’ or ‘WEB\_IDENTITY’ or ‘KBA’ or ‘PHONE’.

**password**

*str* – Optional - “The password required for the recipient to view and sign the document”

**signing\_url**

*str* – If this recipient is associated with an *Agreement* this is the URL that the user can visit to complete/sign the agreement.

Any content in between double quotes (“like this”) is taken from the [Echosign API documentation](#).





## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**A**

ACCEPTED (Agreement.Status attribute), 6  
 access\_token (EchosignAccount attribute), 5  
 account (Agreement attribute), 6  
 account (LibraryDocument attribute), 9  
 Agreement (class in pyEchosign.classes.agreement), 5  
 agreement (User attribute), 10  
 Agreement.SignatureFlow (class in pyE-  
 chosign.classes.agreement), 6  
 Agreement.Status (class in pyE-  
 chosign.classes.agreement), 6  
 AgreementDocument (class in pyE-  
 chosign.classes.documents), 9  
 api\_access\_point (EchosignAccount attribute), 5  
 APPROVED (Agreement.Status attribute), 7  
 ARCHIVED (Agreement.Status attribute), 7  
 audit\_trail\_file (Agreement attribute), 7  
 audit\_trail\_file (LibraryDocument attribute), 10  
 authentication\_method (User attribute), 11

**C**

cancel() (Agreement method), 7  
 combined\_document (Agreement attribute), 7

**D**

delete() (Agreement method), 7  
 delete() (LibraryDocument method), 10  
 DELIVERED (Agreement.Status attribute), 7  
 document (LibraryDocument attribute), 9  
 document\_id (TransientDocument attribute), 10  
 documents (Agreement attribute), 7

**E**

echosign\_id (Agreement attribute), 6  
 echosign\_id (AgreementDocument attribute), 9  
 echosign\_id (LibraryDocument attribute), 9  
 EchosignAccount (class in pyEchosign.classes.account),  
 5  
 expiration\_date (TransientDocument attribute), 10

EXPIRED (Agreement.Status attribute), 7

**F**

field\_name (AgreementDocument attribute), 9  
 file (TransientDocument attribute), 10  
 file\_name (TransientDocument attribute), 10  
 files (Agreement attribute), 6  
 FORM (Agreement.Status attribute), 7  
 form\_field\_layer (LibraryDocument attribute), 9  
 FORM\_FILLED (Agreement.Status attribute), 7  
 fully\_retrieved (Agreement attribute), 6

**G**

get\_agreements() (EchosignAccount method), 5  
 get\_form\_data() (Agreement method), 8  
 get\_library\_documents() (EchosignAccount method), 5  
 get\_signing\_urls() (Agreement method), 8  
 GLOBAL (LibraryDocument attribute), 9

**H**

headers() (EchosignAccount method), 5

**J**

json\_to\_agreement() (pyE-  
 chosign.classes.agreement.Agreement class  
 method), 8  
 json\_to\_agreement() (pyE-  
 chosign.classes.library\_document.LibraryDocument  
 class method), 10  
 json\_to\_agreements() (pyE-  
 chosign.classes.agreement.Agreement class  
 method), 8  
 json\_to\_agreements() (pyE-  
 chosign.classes.library\_document.LibraryDocument  
 class method), 10

**L**

LibraryDocument (class in pyE-  
 chosign.classes.library\_document), 9

locale (LibraryDocument attribute), 10

## M

mime\_type (AgreementDocument attribute), 9

mime\_type (TransientDocument attribute), 10

modified\_date (LibraryDocument attribute), 9

## N

name (Agreement attribute), 6

name (AgreementDocument attribute), 9

name (LibraryDocument attribute), 9

## O

OTHER (Agreement.Status attribute), 7

OUT\_FOR\_ACCEPTANCE (Agreement.Status attribute), 7

OUT\_FOR\_APPROVAL (Agreement.Status attribute), 7

OUT\_FOR\_DELIVERY (Agreement.Status attribute), 7

OUT\_FOR\_FORM\_FILLING (Agreement.Status attribute), 7

OUT\_FOR\_SIGNATURE (Agreement.Status attribute), 7

## P

page\_count (AgreementDocument attribute), 9

PARALLEL (Agreement.SignatureFlow attribute), 6

password (User attribute), 11

PERSONAL (LibraryDocument attribute), 10

## R

RECALLED (Agreement.Status attribute), 7

retrieve\_complete\_document() (LibraryDocument method), 10

## S

scope (LibraryDocument attribute), 9, 10

send() (Agreement method), 8

send\_reminder() (Agreement method), 9

SENDER\_SIGNS\_ONLY (Agreement.SignatureFlow attribute), 6

SEQUENTIAL (Agreement.SignatureFlow attribute), 6

SHARED (LibraryDocument attribute), 10

SIGNED (Agreement.Status attribute), 7

signing\_url (User attribute), 11

status (Agreement attribute), 6

supporting\_document (AgreementDocument attribute), 9

## T

TransientDocument (class in pyEchosign.classes.documents), 10

## U

User (class in pyEchosign.classes.users), 10

user\_email (EchosignAccount attribute), 5

user\_id (EchosignAccount attribute), 5

users (Agreement attribute), 6

## W

WAITING\_FOR\_AUTHORIZING (Agreement.Status attribute), 7

WAITING\_FOR\_FAXIN (Agreement.Status attribute), 7

WAITING\_FOR\_FAXING (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_ACCEPTANCE (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_ACKNOWLEDGEMENT (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_APPROVAL (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_DELEGATION (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_FORM\_FILLING (Agreement.Status attribute), 7

WAITING\_FOR\_MY\_SIGNATURE (Agreement.Status attribute), 7

WIDGET (Agreement.Status attribute), 7