pyEchosign Documentation

Release 0.3.8dev

Jens Astrup

Oct 29, 2017

Contents

CHAPTER 1

About

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

Documentation

The most up to date documentation can be found on pyEchosign's RTD page.

Maintained on GitLab

This project is maintained on GitLab and mirrored to GitHub. Issues opened on the latter are still addressed.

CHAPTER 2

Notes

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.

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.

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
```

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

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 Agreement.**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_AUTHORING = 'WAITING_FOR_AUTHORING'

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'

Agreement.**audit_trail_file**
The PDF file of the audit trail.

Agreement.**cancel**()
Cancels the agreement on Echosign. Agreement will still be visible in the Manage page.

Agreement.**combined_document**
The PDF file containing all documents within this agreement.

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

---

Agreement.**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*

Agreement.**get_form_data**()
Retrieves the form data for this agreement as CSV.

Returns: StringIO

Agreement.**get_signing_urls**()
Associate the signing URLs for this agreement with its *recipients*

classmethod Agreement.**json_to_agreement**(*account*, *json_data*)

classmethod Agreement.**json_to_agreements**(*account*, *json_data*)

Agreement.**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.

`Agreement.`**`send_reminder`**(*comment=''*)
Send a reminder for an agreement to the participants.

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

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)

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

Index