hellosign-python-sdk Documentation

Release 3.2

HelloSign

March 02, 2017

Contents

Contents:

CHAPTER 1

hellosign_sdk package

Subpackages

hellosign_sdk.resource package

Submodules

hellosign_sdk.resource.account module

class hellosign_sdk.resource.account.**Account**(*jsonstr=None*, *key=None*, *warnings=None*)

Bases: *hellosign_sdk.resource.resource.Resource*

Contains information about an account and its settings.

Attributes: account_id (str): The id of the Account

email_address (str): The email address associated with the Account

is_paid_hs (bool) : If the user has a paid HelloSign license will return true

is_paid_hf (bool): If the user has a paid HelloFax license will return true

quotas (dict) : An object detailing remaining monthly quotas, which has the following attributes: templates_left (int): API templates remaining api_signature_requests_left (int): API signature requests remaining

callback_url (str): The URL that HelloSign events will be POSTed to

role_code (str): The membership role for the team. a = Admin, m = Member d = Developer

Examples: To print the account_id

```
>>> from hsclient import HSClient
>>> client = HSClient()
>>> account = client.get_account_info()
>>> print account.account_id
```

ROLE_ADMIN = 'a'

ROLE_DEVELOPER = 'd'

ROLE_MEMBER = 'm'

hellosign_sdk.resource.embedded module

class hellosign_sdk.resource.embedded.**Embedded**(*jsonstr=None*, *key=None*, *warnings=None*)
Bases: *hellosign_sdk.resource.resource.Resource*

An object that contains necessary information to set up embedded signing.

Attributes: sign_url (str): URL of the signature page to display in the embedded iFrame

expires_at (str): When the link expires

hellosign_sdk.resource.resource module

class hellosign_sdk.resource.resource.**Resource**(*jsonstr=None*, *key=None*, *warnings=None*)
Bases: object

An abstract class to represent some objects used by our SDK such as Account, SignatureRequest, Template, Team, UnclaimedDraft, Embedded. These objects share the same way of storing data.

Information and settings are stored physically in *self.json_data*, and can be retrieved by using the OOP way.

INTERNALS = ['json_data', 'warnings']

get_warnings()
Return the list of warnings associated with this object, or None if there aren't any

json_data = None

warnings = None

hellosign_sdk.resource.template module

class hellosign_sdk.resource.template.**Template**(*jsonstr=None*, *key=None*, *warnings=None*)
Bases: *hellosign_sdk.resource.resource.Resource*

Contains information about the templates you and your team have created

Attributes: template_id (str): The id of the Template

title (str): The title of the Template which will also be the default subject of the message sent to signers when using this Template to send a SignatureRequest.

message (str): The default message that will be sent to signers when using this Template to send a SignatureRequest.

signer_roles (list of dict): An array of the designated signer roles that must be specified when sending a SignatureRequest using this Template.

name (str): The name of the Role order (int): If signer order is assigned this is the 0-based index for this role

cc_roles (list of dict): An array of the designated CC roles that must be specified when sending a SignatureRequest using this Template.

name (str): The name of the Role

documents (list of dict): An array describing each document associated with this Template. Includes form field data for each document.

name (str): Name of the associated file index (int): Document ordering, the lowest index is diplayed first

and the highest last

form_fields (list of dict): An array of Form Field objects containing the name and type of each named textbox and checkmark field.

api_id (str): A unique id for the form field name (str): The name of the form field type (str): The type of this form field x (int): The horizontal offset in pixels for this form field y (int): The vertical offset in pixels for this form field width (int): The width in pixels of this form field height (int): The height in pixels of this form field required (bool): Boolean showing whether or not this field is

required

custom_fields (list of dict): An array of Custom Field objects containing the name and type of each custom field

name (str): The name of the Custom Field type (str): The type of this Custom Field. Currently, 'text' is

the only valid value

named_form_fields (DEPRECATED): Use "form_fields" under the "documents" array instead.

accounts (list of dict): An array of the Accounts that can use this Template.

account_id (str): The id of the Account email_address (str): The email address associated with the Account

is_creator (bool): True if you are the owner of this template, false if it's been shared with you by a team member.

can_edit (bool): Indicates whether edit rights have been granted to you by the owner (always true if that's you).

edit_url (str): Returned when creating a new embedded draft

expires_at (int): Date when the edit_url expires

hellosign_sdk.resource.signature_request module

class hellosign_sdk.resource.signature_request.**SignatureRequest**(*jsonstr=None*, *key=None*, *warnings=None*)

Bases: *hellosign_sdk.resource.resource.Resource*

Contains information regarding documents that need to be signed

Comprises the following attributes:

test_mode (bool): Whether this is a test signature request. Test requests have no legal value. Defaults to False.

signature_request_id (str): The id of the SignatureRequest

requester_email_address (str): The email address of the initiator of the SignatureRequest

title (str): The title the specified Account uses for the SignatureRequest

subject (str): The subject in the email that was initially sent to the signers

message (str): The custom message in the email that was initially sent to the signers

is_complete (bool): Whether or not the SignatureRequest has been fully executed by all signers

has_error (bool): Whether or not an error occurred (either during the creation of the SignatureRequest or during one of the signings)

files_url (str): The URL where a copy of the request's documents can be downloaded

signing_url (str): The URL where a signer, after authenticating, can sign the documents

details_url (str): The URL where the requester and the signers can view the current status of the SignatureRequest

cc_email_addresses (list): A list of email addresses that were CCed on the SignatureRequest. They will receive a copy of the final PDF once all the signers have signed

signing_redirect_url (str): The URL you want the signer redirected to after they successfully sign

custom_fields (list of dict): An array of Custom Field objects containing the name and type of each custom field

name (str): The name of the Custom Field type (str): The type of this Custom Field. Currently, *text* and *checkmark* are the only valid values

response_data (list of dict): An array of form field objects containing the name, value, and type of each textbox or checkmark field filled in by the signers

api_id (str): The unique ID for this field signature_id (str): The ID of the signature to which this response is linked name (str): The name of the form field value (str): The value of the form field type (str): The type of this form field

signatures (list of dict): An array of signature objects, 1 for each signer

signature_id (str): Signature identifier signer_email_address (str): The email address of the signer signer_name (str): The name of the signer order (str): If signer order is assigned this is the 0-based index for this signer status_code (str): The current status of the signature. eg: *awaiting_signature*, *signed*, *on_hold* signed_at (str): Time that the document was signed or null last_viewed_at (str): The time that the document was last viewed by this signer or null last_reminded_at (str): The time the last reminder email was sent to the signer or null has_pin (bool): Boolean to indicate whether this signature requires a PIN to access

find_response_component(*api_id=None*, *signature_id=None*)

Find one or many repsonse components.

Args:

api_id (str): Api id associated with the component(s) to be retrieved.

signature_id (str): Signature id associated with the component(s) to be retrieved.

Returns: A list of dictionaries containing component data

find_signature(*signature_id=None*, *signer_email_address=None*)

Return a signature for the given parameters

Args:

signature_id (str): Id of the signature to retrieve. signer_email_address (str): Email address of the associated signer for the signature to retrieve.

Returns: A Signature object or None

hellosign_sdk.resource.team module

class hellosign_sdk.resource.team.**Team**(*jsonstr=None*, *key=None*, *warnings=None*)
Bases: *hellosign_sdk.resource.resource.Resource*

Contains information about your team and its members

Comprises the following attributes:

name (str): The name of your Team

accounts (list of dict): A list of all Accounts belonging to your Team. Note that this response is a subset of the response parameters found in GET /account.

account_id (str): The id of the Account email_address (str): The email address associated with the Account role_code (str): The membership role for the team.

a = Admin, m = Member d = Developer

invited_accounts (str): A list of all Accounts that have an outstanding invitation to join your Team. Note that this response is a subset of the response parameters found in GET /account.

account_id (str): The id of the Account email_address (str): The email address associated with the Account

hellosign_sdk.resource.unclaimed_draft module

class hellosign_sdk.resource.unclaimed_draft.**UnclaimedDraft**(*jsonstr=None*, *key=None*, *warnings=None*)
Bases: *hellosign_sdk.resource.resource.Resource*

UNCLAIMED_DRAFT_REQUEST_SIGNATURE_TYPE = 'request_signature'
A group of documents that a user can take ownership of by going to the claim URL

Comprises the following attributes:

claim_url (str): The URL to be used to claim this UnclaimedDraft

signing_redirect_url (str): The URL you want signers redirected to after they successfully sign.

test_mode (bool): Whether this is a test draft. Signature requests made from test drafts have no legal value. Defaults to 0.

UNCLAIMED_DRAFT_SEND_DOCUMENT_TYPE = 'send_document'

Module contents

hellosign_sdk.tests package

Submodules

hellosign_sdk.tests.test_helper module

Module contents

hellosign_sdk.utils package

Submodules

hellosign_sdk.utils.exception module

exception hellosign_sdk.utils.exception.**BadGateway**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Bad gateway

exception hellosign_sdk.utils.exception.**BadRequest**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Bad request data

exception hellosign_sdk.utils.exception.**Conflict**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Request correctly formulated but unable to proceed due to a conflict

exception hellosign_sdk.utils.exception.**Forbidden**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Not authorized to proceed

exception hellosign_sdk.utils.exception.**GatewayTimeout**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Gateway timeout

exception hellosign_sdk.utils.exception.**Gone**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Resource deleted

exception hellosign_sdk.utils.exception.**HSException**(*message*)
Bases: exceptions.Exception

General HelloSign exception

We use this object to raise exceptions when none of its child classes is suitable for use.

exception hellosign_sdk.utils.exception.**HTTPError**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HSException*

General HTTP error

exception hellosign_sdk.utils.exception.**InternalServerError**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Server error

exception hellosign_sdk.utils.exception.**MethodNotAllowed**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

HTTP method not supported

exception hellosign_sdk.utils.exception.**MethodNotImplemented**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Not implemented

exception hellosign_sdk.utils.exception.**NoAuthMethod**(*message*)
Bases: *hellosign_sdk.utils.exception.HSException*

No authentication data

exception hellosign_sdk.utils.exception.**NotAcceptable**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Accept header conflicts with the returned resource type

exception hellosign_sdk.utils.exception.**NotFound**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Resource not found

exception hellosign_sdk.utils.exception.**PaymentRequired**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Payment/upgrade required to proceed

exception hellosign_sdk.utils.exception.**RequestTimeout**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Request timeout

exception hellosign_sdk.utils.exception.**RequestURITooLong**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Request URI too long

exception hellosign_sdk.utils.exception.**RequestedRangeNotSatisfiable**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Invalid resource data chunk requested

exception hellosign_sdk.utils.exception.**ServiceUnavailable**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Service unavailable

exception hellosign_sdk.utils.exception.**Unauthorized**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Bad authentication data

exception hellosign_sdk.utils.exception.**UnsupportedMediaType**(*message*, *http_code=None*)
Bases: *hellosign_sdk.utils.exception.HTTPError*

Unsupported media type

hellosign_sdk.utils.hsaccesstokenauth module

class hellosign_sdk.utils.hsaccesstokenauth.**HSAccessTokenAuth**(*access_token*, *access_token_type*, *refresh_token=None*, *expires_in=None*, *state=None*)
Bases: requests.auth.AuthBase

Authentication object using HelloSign's access token

classmethod from_response(*response_data*)
Builds a new HSAccessTokenAuth straight from response data

Args: response_data (dict): Response data to use

Returns: A HSAccessTokenAuth objet

hellosign_sdk.utils.request module

class hellosign_sdk.utils.request.**HSRequest**(*auth*, *env='production'*)
Bases: object

Object to handle HTTP requests

Although we have greate requests package which can handle the HTTP request beautifully, we need this class to fit better our need like sending the requests with authentication information, download files, check HTTP errors...

Attributes: DEFAULT_ENCODING (str): Default encoding for requests USER_AGENT (str): HTTP User agent used when sending requests parameters (dict): Some parameters for GET requests headers (dict): Custome headers for every requests http_status_code (int): HTTP status code returned of requests

DEFAULT_ENCODING = 'UTF-8'

USER_AGENT = 'hellosign-python-sdk'

get(*url*, *headers=None*, *parameters=None*, *get_json=True*)
Send a GET request with custome headers and parameters

Args: url (str): URL to send the request to headers (str, optional): custom headers parameters (str, optional): optional parameters

Returns: A JSON object of the returned response if *get_json* is True, Requests' response object otherwise

get_file(*url*, *path_or_file=None*, *headers=None*, *filename=None*)
Get a file from a url and save it as *filename*

Args: url (str): URL to send the request to

path_or_file (str or file): A writable File-like object or a path to save the file to.

filename (str): [DEPRECATED] File name to save the file as, this can be either a full path or a relative path

headers (str, optional): custom headers

Returns: True if file is downloaded and written successfully, False otherwise.

get_warnings()
Return the list of warnings associated with this request, or None if there aren't any

headers = None

http_status_code = 0

parameters = None

post(*url*, *data=None*, *files=None*, *headers=None*, *get_json=True*)
Make POST request to a url

Args: url (str): URL to send the request to data (dict, optional): Data to send files (dict, optional): Files to send with the request headers (str, optional): custom headers

Returns: A JSON object of the returned response if *get_json* is True, Requests' response object otherwise

response_callback = None

verify_ssl = True

warnings = None

hellosign_sdk.utils.utils module

Module contents

class hellosign_sdk.utils.**api_resource**(*obj_cls*)
Decorator that transforms response data into a Resource

class hellosign_sdk.utils.**api_resource_list**(*obj_cls*)
Bases: *hellosign_sdk.utils.api_resource*

Decorator that transforms response data into a ResourceList

Submodules

hellosign_sdk.hsclient module

class hellosign_sdk.hsclient.**HSClient**(*email_address=None*, *password=None*, *api_key=None*, *access_token=None*, *access_token_type='Bearer'*, *env='production'*)
Bases: object

Client object to interact with the API urls

Most of the operations of the SDK is made through this object. Please refer to the README.rst file for more details on how to use the client object.

ACCOUNT_CREATE_URL = ''

ACCOUNT_INFO_URL = ''

ACCOUNT_UPDATE_URL = ''

ACCOUNT_VERIFY_URL = ''

API_URL = ''

API_VERSION = 'v3'

EMBEDDED_OBJECT_GET_URL = ''

EMBEDDED_TEMPLATE_EDIT_URL = ''

OAUTH_TOKEN_URL = ''

SIGNATURE_REQUEST_CANCEL_URL = ''

SIGNATURE_REQUEST_CREATE_EMBEDDED_URL = ''

SIGNATURE_REQUEST_CREATE_EMBEDDED_WITH_TEMPLATE_URL = ''

SIGNATURE_REQUEST_CREATE_URL = ''

SIGNATURE_REQUEST_CREATE_WITH_TEMPLATE_URL = ''

SIGNATURE_REQUEST_DOWNLOAD_PDF_URL = ''

SIGNATURE_REQUEST_INFO_URL = ''

SIGNATURE_REQUEST_LIST_URL = ''

SIGNATURE_REQUEST_REMIND_URL = ''

TEAM_ADD_MEMBER_URL = ''

TEAM_CREATE_URL = ''

TEAM_DESTROY_URL = ''

TEAM_INFO_URL = ''

TEAM_REMOVE_MEMBER_URL = ''

TEAM_UPDATE_URL = ''

TEMPLATE_ADD_USER_URL = ''

TEMPLATE_CREATE_EMBEDDED_DRAFT_URL = ''

TEMPLATE_DELETE_URL = ''

TEMPLATE_GET_FILES_URL = ''

TEMPLATE_GET_LIST_URL = ''

TEMPLATE_GET_URL = ''

TEMPLATE_REMOVE_USER_URL = ''

UNCLAIMED_DRAFT_CREATE_EMBEDDED_URL = ''

UNCLAIMED_DRAFT_CREATE_EMBEDDED_WITH_TEMPLATE_URL = ''

UNCLAIMED_DRAFT_CREATE_URL = ''

add_team_member(*account_id=None*, *email_address=None*)
Add or invite a user to your Team

Args:

account_id (str): The id of the account of the user to invite to your team.

email_address (str): The email address of the account to invite to your team. The account id prevails if both account_id and email_address are provided.

Returns: A Team object

add_user_to_template(*template_id*, *account_id=None*, *email_address=None*)
Gives the specified Account access to the specified Template

Args:

template_id (str): The id of the template to give the account access to

account_id (str): The id of the account to give access to the template. The account id prevails if both account_id and email_address are provided.

email_address (str): The email address of the account to give access to.

Returns: A Template object

cancel_signature_request(*signature_request_id*)
Cancels a SignatureRequest

Cancels a SignatureRequest. After canceling, no one will be able to sign or access the SignatureRequest or its documents. Only the requester can cancel and only before everyone has signed.

Args:

signing_request_id (str): The id of the signature request to cancel

Returns: None

create_account(**args*, ***kwargs*)
Make a Resource instance

create_embedded_template_draft(*client_id*, *signer_roles*, *test_mode=False*, *files=None*, *file_urls=None*, *title=None*, *subject=None*, *message=None*, *cc_roles=None*, *merge_fields=None*, *use_preexisting_fields=False*)
Creates an embedded Template draft for further editing.

Args:

test_mode (bool, optional): Whether this is a test, the signature request created from this draft will not be legally binding if set to 1. Defaults to 0.

client_id (str): Client id of the app you're using to create this draft.

files (list of str): The file(s) to use for the template.

file_urls (list of str): URLs of the file for HelloSign to use for the template. Use either *files* or *file_urls*, but not both.

title (str, optional): The template title

subject (str, optional): The default template email subject

message (str, optional): The default template email message

signer_roles (list of dict): A list of signer roles, each of which has the following attributes:

name (str): The role name of the signer that will be displayed when the template is used to create a signature request. order (str, optional): The order in which this signer role is required to sign.

cc_roles (list of str, optional): The CC roles that must be assigned when using the template to send a signature request

merge_fields (list of dict, optional): The merge fields that can be placed on the template's document(s) by the user claiming the template draft. Each must have the following two parameters:

name (str): The name of the merge field. Must be unique. type (str): Can only be "text" or "checkbox".

use_preexisting_fields (bool): Whether to use preexisting PDF fields

Returns: A Template object specifying the Id of the draft

create_embedded_unclaimed_draft(*test_mode=False*, *client_id=None*, *is_for_embedded_signing=False*, *requester_email_address=None*, *files=None*, *file_urls=None*, *draft_type=None*, *subject=None*, *message=None*, *signers=None*, *cc_email_addresses=None*, *signing_redirect_url=None*, *requesting_redirect_url=None*, *form_fields_per_document=None*, *metadata=None*, *use_preexisting_fields=False*)
Creates a new Draft to be used for embedded requesting

Args:

test_mode (bool, optional): Whether this is a test, the signature request created from this draft will not be legally binding if set to True. Defaults to False.

client_id (str): Client id of the app used to create the embedded draft.

is_for_embedded_signing (bool, optional): Whether this is also for embedded signing. Defaults to False.

requester_email_address (str): Email address of the requester.

files (list of str): The uploaded file(s) to send for signature.

file_urls (list of str): URLs of the file for HelloSign to download to send for signature. Use either *files* or *file_urls*

draft_type (str): The type of unclaimed draft to create. Use "send_document" to create a claimable file, and "request_signature" for a claimable signature request. If the type is "request_signature" then signers name and email_address are not optional.

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer order (str, optional): The order the signer is required to sign in

cc_email_addresses (list of str, optional): A list of email addresses that should be CC'd

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

requesting_redirect_url (str, optional): The URL you want the signer to be redirected to after the request has been sent.

form_fields_per_document (str, optional): The fields that should appear on the document, expressed as a serialized JSON data structure which is a list of lists of the form fields. Please refer to the API reference of HelloSign for more details (https://www.hellosign.com/api/reference#SignatureRequest)

metadata (dict, optional): Metadata to associate with the draft

use_preexisting_fields (bool): Whether to use preexisting PDF fields

Returns: An UnclaimedDraft object

create_embedded_unclaimed_draft_with_template(*test_mode=False*, *client_id=None*, *is_for_embedded_signing=False*, *template_id=None*, *template_ids=None*, *requester_email_address=None*, *title=None*, *subject=None*, *message=None*, *signers=None*, *ccs=None*, *signing_redirect_url=None*, *requesting_redirect_url=None*, *metadata=None*, *custom_fields=None*)

Creates a new Draft to be used for embedded requesting

Args:

test_mode (bool, optional): Whether this is a test, the signature request created from this draft will not be legally binding if set to True. Defaults to False.

client_id (str): Client id of the app you're using to create this draft. Visit our embedded page to learn more about this parameter.

template_id (str): The id of the Template to use when creating the Unclaimed Draft. Mutually exclusive with template_ids.

template_ids (list of str): The ids of the Templates to use when creating the Unclaimed Draft. Mutually exclusive with template_id.

requester_email_address (str): The email address of the user that should be designated as the requester of this draft, if the draft type is "request_signature."

title (str, optional): The title you want to assign to the Unclaimed Draft

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer

ccs (list of str, optional): A list of email addresses that should be CC'd

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

requesting_redirect_url (str, optional): The URL you want the signer to be redirected to after the request has been sent.

is_for_embedded_signing (bool, optional): The request created from this draft will also be signable in embedded mode if set to True. The default is False.

metadata (dict, optional): Metadata to associate with the draft. Each request can include up to 10 metadata keys, with key names up to 40 characters long and values up to 500 characters long.

custom_fields (list of dict, optional): A list of custom fields. Required when a CustomField exists in the Template. An item of the list should look like this: *{'name: value'}*

create_team(**args*, ***kwargs*)
Make a Resource instance

create_unclaimed_draft(*test_mode=False*, *files=None*, *file_urls=None*, *draft_type=None*, *subject=None*, *message=None*, *signers=None*, *cc_email_addresses=None*, *signing_redirect_url=None*, *form_fields_per_document=None*, *metadata=None*, *use_preexisting_fields=False*)
Creates a new Draft that can be claimed using the claim URL

Creates a new Draft that can be claimed using the claim URL. The first authenticated user to access the URL will claim the Draft and will be shown either the "Sign and send" or the "Request signature" page with the Draft loaded. Subsequent access to the claim URL will result in a 404. If the type is "send_document" then only the file parameter is required. If the type is "request_signature", then the identities of the signers and optionally the location of signing elements on the page are also required.

Args:

test_mode (bool, optional): Whether this is a test, the signature request created from this draft will not be legally binding if set to True. Defaults to False.

files (list of str): The uploaded file(s) to send for signature

file_urls (list of str): URLs of the file for HelloSign to download to send for signature. Use either *files* or *file_urls*

draft_type (str): The type of unclaimed draft to create. Use "send_document" to create a claimable file, and "request_signature" for a claimable signature request. If the type is "request_signature" then signers name and email_address are not optional.

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer order (str, optional): The order the signer is required to sign in

cc_email_addresses (list of str, optional): A list of email addresses that should be CC'd

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

form_fields_per_document (str, optional): The fields that should appear on the document, expressed as a serialized JSON data structure which is a list of lists of the form fields. Please refer to the API reference of HelloSign for more details (https://www.hellosign.com/api/reference#SignatureRequest)

metadata (dict, optional): Metadata to associate with the draft

use_preexisting_fields (bool): Whether to use preexisting PDF fields

Returns: An UnclaimedDraft object

delete_template(*template_id*)

Deletes the specified template

Args:

template_id (str): The id of the template to delete

Returns: A status code

destroy_team()

Delete your Team

Deletes your Team. Can only be invoked when you have a team with only one member left (yourself).

Returns: None

get_account_info()

Get current account information

The information then will be saved in *self.account* so that you can access the information like this:

```
>>> hsclient = HSClient()
>>> acct = hsclient.get_account_info()
>>> print acct.email_address
```

Returns: An Account object

get_embedded_object(**args*, ***kwargs*)

Make a Resource instance

get_last_warnings()
Return the warnings associated with the last request

get_oauth_data(*code*, *client_id*, *client_secret*, *state*)
Get Oauth data from HelloSign

Args:

code (str): Code returned by HelloSign for our callback url

client_id (str): Client id of the associated app

client_secret (str): Secret token of the associated app

Returns: A HSAccessTokenAuth object

get_signature_request(**args*, ***kwargs*)
Make a Resource instance

get_signature_request_file(*signature_request_id*, *path_or_file=None*, *file_type=None*, *filename=None*)
Download the PDF copy of the current documents

Args:

signature_request_id (str): Id of the signature request

path_or_file (str or file): A writable File-like object or a full path to save the PDF file to.

filename (str): [DEPRECATED] Filename to save the PDF file to. This should be a full path.

file_type (str): Type of file to return. Either "pdf" for a single merged document or "zip" for a collection of individual documents. Defaults to "pdf" if not specified.

Returns: True if file is downloaded and successfully written, False otherwise.

get_signature_request_list(**args*, ***kwargs*)
Make a ResourceList instance

get_team_info(**args*, ***kwargs*)
Make a Resource instance

get_template(**args*, ***kwargs*)
Make a Resource instance

get_template_edit_url(**args*, ***kwargs*)
Make a Resource instance

get_template_files(*template_id*, *filename*)
Download a PDF copy of a template's original files

Args:

template_id (str): The id of the template to retrieve.

filename (str): Filename to save the PDF file to. This should be a full path.

Returns: Returns a PDF file

get_template_list(**args*, ***kwargs*)
Make a ResourceList instance

refresh_access_token(*refresh_token*)
Refreshes the current access token.

Gets a new access token, updates client auth and returns it.

Args:

refresh_token (str): Refresh token to use

Returns: The new access token

remind_signature_request(**args*, ***kwargs*)
Make a Resource instance

remove_team_member(*account_id=None*, *email_address=None*)
Remove a user from your Team

Args:

account_id (str): The id of the account of the user to remove from your team.

email_address (str): The email address of the account to remove from your team. The account id prevails if both account_id and email_address are provided.

Returns: A Team object

remove_user_from_template(*template_id*, *account_id=None*, *email_address=None*)
Removes the specified Account's access to the specified Template

Args:

template_id (str): The id of the template to remove the account's access from.

account_id (str): The id of the account to remove access from the template. The account id prevails if both account_id and email_address are provided.

email_address (str): The email address of the account to remove access from.

Returns: An Template object

request = None

response_callback = None

send_signature_request(*test_mode=False*, *files=None*, *file_urls=None*, *title=None*, *subject=None*, *message=None*, *signing_redirect_url=None*, *signers=None*, *cc_email_addresses=None*, *form_fields_per_document=None*, *use_text_tags=False*, *hide_text_tags=False*, *metadata=None*, *ux_version=None*)
Creates and sends a new SignatureRequest with the submitted documents

Creates and sends a new SignatureRequest with the submitted documents. If form_fields_per_document is not specified, a signature page will be affixed where all signers will be required to add their signature, signifying their agreement to all contained documents.

Args:

test_mode (bool, optional): Whether this is a test, the signature request will not be legally binding if set to True. Defaults to False.

files (list of str): The uploaded file(s) to send for signature

file_urls (list of str): URLs of the file for HelloSign to download to send for signature. Use either *files* or *file_urls*

title (str, optional): The title you want to assign to the SignatureRequest

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer order (str, optional): The order the signer is required to sign in pin (str, optional): The 4- to 12-character access code that will secure this signer's signature page

cc_email_addresses (list, optional): A list of email addresses that should be CC'd

form_fields_per_document (str): The fields that should appear on the document, expressed as a serialized JSON data structure which is a list of lists of the form fields. Please refer to the API reference of HelloSign for more details (https://www.hellosign.com/api/reference#SignatureRequest)

use_text_tags (bool, optional): Use text tags in the provided file(s) to create form fields

hide_text_tags (bool, optional): Hide text tag areas

metadata (dict, optional): Metadata to associate with the signature request

ux_version (int): UX version, either 1 (default) or 2.

Returns: A SignatureRequest object

send_signature_request_embedded(*test_mode=False*, *client_id=None*, *files=None*, *file_urls=None*, *title=None*, *subject=None*, *message=None*, *signing_redirect_url=None*, *signers=None*, *cc_email_addresses=None*, *form_fields_per_document=None*, *use_text_tags=False*, *hide_text_tags=False*, *metadata=None*, *ux_version=None*)

Creates and sends a new SignatureRequest with the submitted documents

Creates a new SignatureRequest with the submitted documents to be signed in an embedded iFrame . If form_fields_per_document is not specified, a signature page will be affixed where all signers will be required to add their signature, signifying their agreement to all contained documents. Note that embedded signature requests can only be signed in embedded iFrames whereas normal signature requests can only be signed on HelloSign.

Args:

test_mode (bool, optional): Whether this is a test, the signature request will not be legally binding if set to True. Defaults to False.

client_id (str): Client id of the app you're using to create this embedded signature request. Visit the embedded page to learn more about this parameter (https://www.hellosign.com/api/embeddedSigningWalkthrough)

files (list of str): The uploaded file(s) to send for signature

file_urls (list of str): URLs of the file for HelloSign to download to send for signature. Use either *files* or *file_urls*

title (str, optional): The title you want to assign to the SignatureRequest

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer order (str, optional): The order the signer is required to sign in pin (str, optional): The 4- to 12-character access code that will secure this signer's signature page

cc_email_addresses (list, optional): A list of email addresses that should be CCed

form_fields_per_document (str): The fields that should appear on the document, expressed as a serialized JSON data structure which is a list of lists of the form fields. Please refer to the API reference of HelloSign for more details (https://www.hellosign.com/api/reference#SignatureRequest)

use_text_tags (bool, optional): Use text tags in the provided file(s) to create form fields

hide_text_tags (bool, optional): Hide text tag areas

metadata (dict, optional): Metadata to associate with the signature request

ux_version (int): UX version, either 1 (default) or 2.

Returns: A SignatureRequest object

send_signature_request_embedded_with_template(*test_mode=False*, *client_id=None*, *template_id=None*, *template_ids=None*, *title=None*, *subject=None*, *message=None*, *signing_redirect_url=None*, *signers=None*, *ccs=None*, *custom_fields=None*, *metadata=None*, *ux_version=None*)

Creates and sends a new SignatureRequest based off of a Template

Creates a new SignatureRequest based on the given Template to be signed in an embedded iFrame. Note that embedded signature requests can only be signed in embedded iFrames whereas normal signature requests can only be signed on HelloSign.

Args:

test_mode (bool, optional): Whether this is a test, the signature request will not be legally binding if set to True. Defaults to False.

client_id (str): Client id of the app you're using to create this embedded signature request. Visit the embedded page to learn more about this parameter (https://www.hellosign.com/api/embeddedSigningWalkthrough)

template_id (str): The id of the Template to use when creating the SignatureRequest. Mutually exclusive with template_ids.

template_ids (list): The ids of the Templates to use when creating the SignatureRequest. Mutually exclusive with template_id.

title (str, optional): The title you want to assign to the SignatureRequest

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

signers (list of dict): A list of signers, which each has the following attributes:

name (str): The name of the signer email_address (str): Email address of the signer pin (str, optional): The 4- to 12-character access code that will secure this signer's signature page

ccs (list of dict, optional): The email address of the CC filling the role of RoleName. Required when a CC role exists for the Template. Each dict has the following attributes:

role_name (str): CC role name email_address (str): CC email address

custom_fields (list of dict, optional): A list of custom fields. Required when a CustomField exists in the Template. An item of the list should look like this: *{'name: value'}*

metadata (dict, optional): Metadata to associate with the signature request

ux_version (int): UX version, either 1 (default) or 2.

Returns: A SignatureRequest object of the newly created Signature Request

send_signature_request_with_template(*test_mode=False*, *template_id=None*, *template_ids=None*, *title=None*, *subject=None*, *message=None*, *signing_redirect_url=None*, *signers=None*, *ccs=None*, *custom_fields=None*, *metadata=None*, *ux_version=None*)

Creates and sends a new SignatureRequest based off of a Template

Creates and sends a new SignatureRequest based off of the Template specified with the template_id parameter.

Args:

test_mode (bool, optional): Whether this is a test, the signature request will not be legally binding if set to True. Defaults to False.

template_id (str): The id of the Template to use when creating the SignatureRequest. Mutually exclusive with template_ids.

template_ids (list): The ids of the Templates to use when creating the SignatureRequest. Mutually exclusive with template_id.

title (str, optional): The title you want to assign to the SignatureRequest

subject (str, optional): The subject in the email that will be sent to the signers

message (str, optional): The custom message in the email that will be sent to the signers

signing_redirect_url (str, optional): The URL you want the signer redirected to after they successfully sign.

signers (list of dict): A list of signers, which each has the following attributes:

role_name (str): Signer role name (str): The name of the signer email_address (str): Email address of the signer pin (str, optional): The 4- to 12-character access code that will secure this signer's signature page

ccs (list of str, optional): The email address of the CC filling the role of RoleName. Required when a CC role exists for the Template. Each dict has the following attributes:

role_name (str): CC role name email_address (str): CC email address

custom_fields (list of dict, optional): A list of custom fields. Required when a CustomField exists in the Template. An item of the list should look like this: *{'name: value'}*

metadata (dict, optional): Metadata to associate with the signature request

ux_version (int): UX version, either 1 (default) or 2.

Returns: A SignatureRequest object

update_account_info(**args*, ***kwargs*)
Make a Resource instance

update_team_name(**args*, ***kwargs*)
Make a Resource instance

verify_account(*email_address*)
Verify whether a HelloSign Account exists

Args:

email_address (str): Email address for the account to verify

Returns: True or False

version = '3.8.5'

hellosign_sdk.test_hsclient module

Module contents

==

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

h

A

B

C

D

E

F

G

H

I

J

M

N

O

P

R

S

T

U

V

W