
# Universign Signature

*Signature Service Guide*

This is a technical document, it has no legal binding value.


Universign
5-7 rue du Faubourg Poissonnière
F-75009 Paris
France

Tel: +33 1 44 08 73 00
Fax: +33 1 43 56 50 42
https://www.universign.com

# Contents

# 1 Introduction

This document explains general functionalities of the UNIVERSIGN Signature Service and how to integrate it with existing applications.

This document is structured as follows:

- section 2 introduces the functionalities of the service;
- section 3 is a guide to the API used to configure and use the service;
- the appendices contain more detailed references to technicals details such as error code meaning.

While great care has been taken to make integration as simple as possible, the reader is assumed to be familiar with programming concepts in the language used for integration.

# 2 General Information

## 2.1 Signature modes

The UNIVERSIGN signature service proposes three modes of signature.

### 2.1.1 Document signature (server side)

This mode allows a UNIVERSIGN user to perform a signature on a PDF document, using our web-service API. In practice, it consists in a single remote procedure call which returns the signed PDF synchronously.

The used private key has to be previously uploaded to UNIVERSIGN by the user, using *http://universign.com* website.

### 2.1.2 Contract signature (client side)

This mode allows one or more people to sign a list of documents. It implies the intervention of two kinds of actors:

- a single **requester**, who is a UNIVERSIGN user and creates a **transaction** with the documents to sign, a list of signers and some options;
- one or more **signers**, who are invited one after another to sign the documents using a user-friendly web interface.

The creation of the transaction, the retrieval of the signed documents and the retrieval of information on the status of a transaction are made through the UNIVERSIGN Web Service API by the requester. The requester also chooses which certificate types are allowed for the signers to use.

The selection of the private key to use, the approval of the terms of the documents and the launching of signature process are made by the signer. Each signer reaches the signature web interface via an URL containing a unique id. Each signer receives his URL by mail, except possibly the first if the requester chooses to.

Those interactions are depicted with a very simple example on figure 1.

This mode has an extension which is described in the following section.

### 2.1.3 Contract signature (client side) with signer registration

It's important to note that Universign is also a Registration Authority (RA), a signer can register himself in order to obtain a certificate issued from Universign. This will make his identity certified by Universign and allow him to sign

a **certified** signature. In order to register his identity, a user should send his ID documents to Universign RA, this can be done in two ways:

- By using the API: when requesting a transaction, the ID documents of a signer can be sent to Universign RA using the `requester.requestTransaction` service and setting the `idDocuments` key of the `TransactionSigner` data object.
  The API also allows a standalone registration of the signer using `requester.requestRegistration` or `requester.requestTwoStepsRegistration`
- By using the web interface: if a transaction was requested to use the **certified** signature and if the current signer is not yet certified by Universign, then when he will proceed to sign the transaction he will be requested to provide his ID documents.

The ID documents that can be used to register a user are:

- French ID card.
- French Passport.
- Residence Permit.
- European Driving License (New Format, issued since September 2013).

When registering his identity, a user will follow these steps:

- Check his identity information by indicating his birth date and confirming it and his phone number. The birth date may was already indicated when the transaction was requested with setting the `birthDate` key of the `TransactionSigner` data object.
- Provide his ID documents. If they were sent when requesting the transaction, then they will be displayed and locked so that the user cannot edit them.
- Validate his identity by agreeing to use Universign service by reading the **Universign Services Subscription Form** document, which contains the user information and indications about the subscriber commitment, and then signing it.

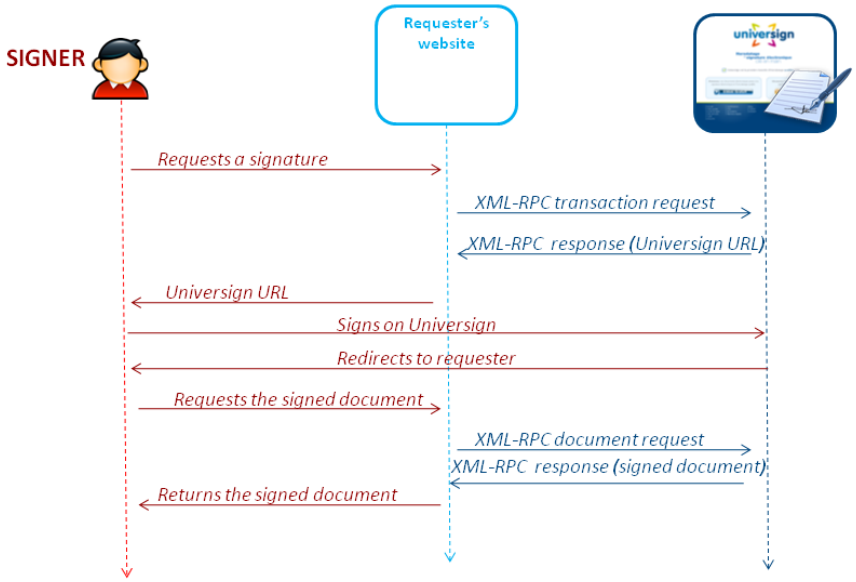


Figure 1: Contract signature interactions

## 2.2 Protocol

In order to allow integration from many different platforms and languages, UNIVERSIGN has chosen to provide a Web Services type API, under the form of XML-RPC (Remote Procedure Call). XML-RPC protocol allows the consistent transmission of parameters from any platform and has built-in support in many languages. For language with no support of this protocol, or if you wish to re-implement an XML-RPC library, the XML-RPC language specification is available at http://www.xmlrpc.com/.

The following conventions are in use with the UNIVERSIGN signature service:

- The URL for XML-RPC requests is: https://ws.universign.eu/sign/rpc/
- All strings must be encoded in UTF-8.
- For readability reason, we shall write byte[] in this document to denote binary data, instead of the XML-RPC base64 type.

- In the API, all parameters which are not XML-RPC predefined types are XML-RPC structs (e.g. hashtables or maps or dictionaries depending on your programming language). All keys in XML-RPC structs are case sensitive.
- In the API, all methods for which the return value is not specified return the integer 0.
- In case of failure while producing a signature the server returns an XML-RPC fault message with an error code and a string message describing the cause of failure. The error code list can be found in the appendixes.

## 2.3 Authentication and confidentiality

All UNIVERSIGN signature services communications are over HTTPS to guarantee the confidentiality of the data and the authentication of the server.

It is based on basic access authentication, with user name and password, as defined in the RFC 2617[2] and is supported by a wide range of HTTP implementations in most languages. The confidentiality of passwords is guaranteed by the SSL layer.

## 2.4 Signature format and standard

The UNIVERSIGN signature service API supports the creation of signed PDF document according to PAdES signature profiles as described in [1].

The PAdES signature service does not produce PDF document but adds *approval signatures* (also called *ordinary signatures* or *document signatures*) to existing PDF documents.

Note that the performed signatures include a timestamp.

## 2.5 Services pre-configuration

As exposed in the API documentation 3, the UNIVERSIGN user has a great choice of options and configuration parameters at request time. However, as to enable or personnalize some features, the UNIVERSIGN teams are in charge of a part of the configuration of the services, prior to their uses. The features that need such activation or configuration are presented below.

**The signature field** needs to be activated and personnalized in size and content to be used. The content can be a composition of images and texts.

**An URL for status push** in the client signature mode. A GET HTTP request will be performed on this optional property each time a signature step is completed (i.e. each time a signer succeeds, fails or cancels its signature).Three parameters are set to the request:

**id** : the id of the transaction

**signer** : the number of the signer who performed the last action on the transaction. The count of the signers is 0-based.

**status** : the status of the transaction. It can be 0 for ready (waiting for the next signer), 1 for expired (created but not completed after 14 days), 2 when completed (all signers signed), 3 when canceled by a signer, 4 when failed, 5 when pending a validation by UNIVERSIGN Registration Authority.

**Elements for the web interface personnalization** : a logo and a company name that will be displayed on the signature web page.

Configuring those items is made on request by the UNIVERSIGN teams. All these properties and options are stored in properties set called *profiles*.

## 2.6 Documents retention policy and documents retrieval

*In the case of server-side signatures, neither the original or the signed document are saved by* UNIVERSIGN *and the signed document is synchronously returned: this section does not apply.*

For client-side signatures, the documents of a transaction are kept by UNIVERSIGN for 30 days after the transaction creation date. During this period and as soon as the transaction is completed (i.e. all signers signed), the documents can be retrieved using the API.

Afterward, and for an unlimited time, the documents are archived. At this point, their retrieval is no more guaranteed to be synchronous, as a de-archival process may be necessary. In this case, a first request for the documents will send an error. A further request, once the de-archival process has finished, will return the documents. A de-archival process can take up to few hours.

If a UNIVERSIGN user does not want its documents to be archived, he must request it to the UNIVERSIGN teams, as part of the services configuration, as exposed in section 2.5. In this case, 30 days after the transaction creation, the documents are deleted from UNIVERSIGN.

In the particular cases of out-dated, canceled or failed transaction, the documents are immediately deleted from UNIVERSIGN.

# 3 Signature service API

- **byte[] signer.sign(byte[] document)**

  Signs a document, using the default signature profile and options.

- **byte[] signer.signWithOptions(byte[] document, SignOptions options)**

  Signs a document, using a set of options.

- **TransactionResponse requester.requestTransaction(TransactionRequest request)**

  Requests a new transaction for the client signature service. Sends the document to be signed and other parameters and returns an URL where the end user should be redirected to. A transaction must be completed whithin 14 days after its request.

- **TransactionDocument[] requester.getDocuments(String transactionId)**

  Requests signed and viewed documents (after their transaction is completed) by their transaction id. See section 2.6.

- **TransactionDocument[] requester.getDocumentsByCustomId(String customId)**

  Requests signed and viewed documents (after their transaction is finished) by their custom id. This method is very similar to *requester.getDocuments(String transactionId)*. It can be used if you do not want to store an external ID.

- **TransactionInfo requester.getTransactionInfo(String transactionId)**

  Requests information about the status of the transaction with this id.

- **TransactionInfo requester.getTransactionInfoByCustomId(String customId)**

  Requests information about the status of the transaction with this id. This method is very similar to the previous one. It can be used if you do not want to store an external ID.

- **requester.relaunchTransaction(String transactionId)**

  Refreshes the creation date for the transaction. The invitation email is sent again if the parameters allow it (*chainingMode* equals **email** and in the case of the first signer, *mustContactFirstSigner* equals **true**). This method can be used to postpone the expiration date of the transaction.

- **requester.cancelTransaction(String transactionId)**

  Cancel a transaction in progress with this id.

- **string[] requester.listTransactions(TransactionFilter filter)**

  Requests the list of transactions matching the given filter. At most 1000 results are returned: to have more results, use multiple requests and ranges in TransactionFilter.

- **TransactionResponse requester.requestRegistration( StandaloneRegistrationRequest request)**

  Requests the standalone registration of the signer. Sends the signer identity to be certified and returns an URL where the signer should be redirected to. The registration process is similar to the transaction one but without documents to sign.

- **requester.requestTwoStepsRegistration( StandaloneRegistrationRequest request)**

  Since 8.4

  Requests a **two steps** standalone registration of the signer. This service should be called by an operator after a face-to-face meeting with the signer. It sends the signer's identity to be certified including his identity documents. Once the identity documents are validated by **Universign**, an email will be sent to the signer with an URL where he can review his identity information and agree to the Universign Services Subscription Form. This registration process is similar to the transaction one but without documents to sign.

# 4 Data objects

In this section are presented the structure of the data objects used in the API.

All data structures are XML-RPC structs (i.e. dictionaries). The first column contains the key and a letter which can be:

**M** if the key is mandatory and must be affected a none-null value;

**O** if the key is optionnal.

The second column is the type of the corresponding value. The third column contains additional information about the parameter such as the optionality, details about the data format...

## 4.1 SignOptions

The `SignOptions` data structure contains options for a *document signature* and allows to specify a signature profile to use.

| Key | Type | Comments |
|---|---|---|
| profile [O] | string | The name of the signature profile to use. A signature profile contains a private key and a signature field. Signature profiles are set up by the UNIVERSIGN team. The default value is "default". |
| signatureField [O] | SignatureField | A description of a visible PDF signature field. If none is provided, no signature field will be produced on the signed document. |
| reason [O] | string | The reason for the digital signature. |
| location [O] | string | The signer's location. |
| signatureFormat [O] | string | The signature format. The available values are:<br>**PADES** The signature follows the format defined in ETSI TS 102 778-3 PAdES Part 3: PAdES Enhanced - PAdES-BES.<br>**PADES-COMP** The signature follows the format defined in ISO 32000-1 with the signing certificate attribute. This format is a compatibility mode with PAdES (same semantic than PAdES with the ISO 32000-1 format).<br>**ISO-32000-1** The default value. The signature follows the format defined in ETSI TS 102 778-2 PAdES Part 2: CMS Profile based on ISO 32000-1. |

| Key | Type | Comments |
|---|---|---|
| language [O] | string | The language for this transaction. The valid values are:<br>**bg** for Bulgarian<br>**ca** for Catalan<br>**de** for German<br>**en** for English (default value)<br>**es** for Spanish<br>**fr** for French<br>**it** for Italian<br>**nl** for Dutch<br>**pl** for Polish<br>**pt** for Portuguese<br>**ro** for Romanian |
| patternName [O] | string | The name of the pattern for the signature field. |

## 4.2 TransactionRequest

The `TransactionRequest` data structure contains informations and options for a Signature transaction creation.

| Key | Type | Comments |
|---|---|---|
| profile [O] | string | The name of the signature profile to use. Signature profiles mainly differ by the displayed company name and logo, and the pre-configured signature field stored within. Signature profiles are set up by the UNIVERSIGN team. The default value is "default". |
| customId [O] | string | A requester-set unique id that can be used to identify this transaction. If not unique, a fault will be thrown. Note that UNIVERSIGN generate its own unique id for each transaction and return it to the requester. |
| signers [M] | TransactionSigner[] | The signers that will have to take part to the transaction. Must contain at least one element. |
| documents [M] | TransactionDocument[] | The documents to be signed. Must contain at least one element. The size limit of each document is set to 10Mo. |

| Key | Type | Comments |
|---|---|---|
| mustContactFirstSigner [O] | boolean | If set to *True*, the first signer will receive an invitation to sign the document(s) by e-mail as soon as the transaction is requested. False by default. |
| finalDocSent [O] | boolean | Tells whether each signer must receive the signed documents by e-mail when the transaction is completed. False by default. |
| finalDocRequesterSent [O] | boolean | Tells whether the requester must receive the signed documents via e-mail when the transaction is completed. False by default. |
| finalDocObserverSent [O] | boolean | Tells whether the observers must receive the signed documents via e-mail when the transaction is completed. It takes the `finalDocSent` value by default. |
| description [O] | string | Description or title of the signature. |
| certificateType [O] | string | This option is used to indicate which certificate type will be used to perform the signature and therefore which type of signature will be performed. The available values are:<br>**certified** Allows signers to perform a certified signature.<br>**advanced** Allows signers to perform an advanced signature which requires the same options as a **certified** signature.<br>**simple** Allows signers to perform a simple signature. The default value. |

| Key | Type | Comments |
|---|---|---|
| language [O] | string | The interface language for this transaction. The valid values are:<br>**bg** for Bulgarian<br>**ca** for Catalan<br>**de** for German<br>**en** for English (default value)<br>**es** for Spanish<br>**fr** for French<br>**it** for Italian<br>**nl** for Dutch<br>**pl** for Polish<br>**pt** for Portuguese<br>**ro** for Romanian |
| handwrittenSignatureMode [O] | int | The mode to enable the `handwritten signature`. There are three modes:<br>• "0": disables the hand-written signature<br>• "1": enables the hand-written signature<br>• "2": enables the hand-written signature if only it is a touch interface<br>If `handwritten signature` is enabled, the signer is prompted to draw a signature on the web interface and the SignatureField bean becomes mandatory for each of the TransactionSigners. This signature is added in his signature field, as an image would be.<br>HandwrittenSignatureMode can not be enabled against a transaction containing only document for presentation. |

| Key | Type | Comments |
|---|---|---|
| chainingMode [O] | string | This option indicates how the signers are chained during the signing process. The valid values are:<br>**none** No invitation email is sent in this mode. Each signer is redirected to the `successURL` after signing. It is up to the requester to contact each of the signers.<br>**email** The default value. The signers receive the invitation email (except for the first one, see `mustContactFirstSigner`) and are redirected to the `successURL`.<br>**web** Enables the `linked signature` mode. In this mode, all signers are physically at the same place. After a signer completed its signature, he will be redirected to the next signer's signature page instead of being returned to the `successURL` and the next signer will not receive an invitation mail. The last signer will be redirected to the `successURL`. |
| finalDocCCeMails [O] | String[] | This option allows to send a copy of the final signed documents to a list of email addresses. This copy is send as cc for every final signed documents email addressed to a signer. For this option to be taken into account, the option `finalDocSent` must be sent to `True`. |
| twoStepsRegistration [O] | boolean | This option allows registration of signers via a two steps registration process. When activated, the RegistrationRequest bean becomes mandatory for each of the unregistered TransactionSigners, the `certicateType` must be set to *advanced*, the phoneNumber and the birthDate must be set.<br>Default value is *False*. |

## 4.3 TransactionSigner

A `TransactionSigner` describes and contains options for a document signer.

| Key | Type | Comments |
|---|---|---|
| firstname [O] | string | This signer's firstname. Note that this field is mandatory for a self-signed certificate.<br>When using validationSessionId, it must be set to the same value than the one used in the validation request. |
| lastname [O] | string | This signer's lastname. Note that this field is mandatory for a self-signed certificate.<br>When using validationSessionId, it must be set to the same value than the one used in the validation request. |
| organization [O] | string | This signer's organization. |
| profile [O] | string | The name of the signer profile to use for some customizations. It is set up by the UNIVERSIGN team. |
| emailAddress [O] | string | This signer's e-mail address. Note that all users except the first must have an email address set. The first user must have one if he has to be contacted by e-mail, e.g. for authentication or if the `mustContactFirstSigner` parameter of TransactionRequest is set to *true*. |
| phoneNum [O] | string | This signer's mobile phone number that should be written in the international format: the country code followed by the phone number (for example, in France 33 XXXXXXXXX). |
| language[O] | String | The language for the signer's transaction. The valid values are:<br>**bg** for Bulgarian<br>**ca** for Catalan<br>**de** for German<br>**en** for English (default value)<br>**es** for Spanish<br>**fr** for French<br>**it** for Italian<br>**nl** for Dutch<br>**pl** for Polish<br>**pt** for Portuguese<br>**ro** for Romanian |
| role [O] | string | The role of this transaction actor<br>**signer** (default) This actor is a signer and he will be able to view the documents and sign them.<br>**observer** This actor is an observer and he will be able only to view the documents. |

| Key | Type | Comments |
|---|---|---|
| birthDate [O] | date | This signer's birth date. This is an option for the **certified** signature, if it's set, the user won't be asked to provide it's birth date during the RA workflow.<br>When using validationSessionId, it must be set to the same value than the one used in the validation request. |
| universignId [O] | string | An external identifier given by the organization that indicates this signer. |
| successURL [O] | string | The url to where the signer will be redirected, after the signatures are completed. If it is null it takes the value of `TransactionRequest.successURL`. If it is also null, it takes the default Universign success URL. |
| cancelURL [O] | string | The url to where the signer will be redirected, after the signatures are canceled. If it is null it takes the value of `TransactionRequest.cancelURL`. If it is also null, it takes the value of successURL. If it is also null, it takes the default Universign cancel URL. |
| failURL [O] | string | The url to where the signer will be redirected, after the signatures are failed. If it is null it takes the value of `TransactionRequest.failURL`. If it is also null, it takes the value of cancelURL. If it is also null, it takes the default Universign failure URL. |
| certificateType [O] | string | Indicates which certificate type will be used to perform the signature and therefore which type of signature will be performed by this signer. The available values are:<br>**certified** Allows signers to perform a certified signature.<br>**advanced** Allows signers to perform an advanced signature which requires the same options as a **certified** signature.<br>**simple** Allows signers to perform a simple signature. The default value. |
| idDocuments [O] | RegistrationRequest | The ID documents to use in this signer registration. This is an option for the **certified** signature, if it's set, the user won't be prompted to provide its ID documents in the RA workflow. |

| Key | Type | Comments |
|---|---|---|
| validationSessionId | String | The ID of a valid ID Validation Session retrieved from a validation request (see universign-guide-8.8-ra.pdf). The documents in this ID Validation session will be used and no need to provide `idDocuments`. |

## 4.4 RegistrationRequest

The `RegistrationRequest` data structure contains information for the signer registration.

| Key | Type | Comments |
|---|---|---|
| documents | byte[][] | List of ID documents to use to register the signer. The number of these documents is indicated in the following comment. |
| type | String | The type of the provided ID documents.<br>**id_card_fr** French ID card. **Two** ID documents should be provided.<br>**passport_eu** French Passport. Only **one** ID document should be provided.<br>**titre_sejour** Residence Permit. **Two** ID documents should be provided.<br>**drive_license** European Driving License (New Format, issued since September 2013). **Two** ID documents should be provided. |

## 4.5 TransactionDocument

The `TransactionDocument` data structure contains information about a transaction document.

| Key | Type | Comments |
|---|---|---|
| documentType [O] | string | This `TransactionDocument` type. Valid values are:<br>**pdf** The default value. Makes all `TransactionDocument` members relevant, except for `SEPAData`<br>**pdf-for-presentation** This value marks the document as view only.<br>**sepa** Using this value, no PDF document is provided, but UNIVERSIGN creates a SEPA mandate from data sent in `SEPAData`, which becomes the single relevant member. |
| content [O] | byte[] | The raw content of the PDF document. You can provide the document using the `url` field, otherwise this field is mandatory. |
| url [O] | string | The URL to download the PDF document. Note that this field is mandatory if the `content` is not set. |
| fileName | string | The file name of this document. |
| signatureFields [O] | DocSignatureField[] | A description of a visible PDF signature field. |
| checkBoxTexts [O] | String[] | Texts of the agreement checkboxes. The last one should be the text of the checkbox related to signature fields labels agreement.<br>This attribute should not be used with documents of the type "pdf-for-presentation". Since agreement for "pdf-for-presentation" is not customisable. |
| metaData [O] | Struct | This structure can only contain simple types like integer, string or date. |
| title [O] | String | A title to be used for display purpose. |
| SEPAData [O] | SEPAData | A structure containing data to create a SEPA mandate PDF. |

## 4.6 SignatureField

The `SignatureField` data structure describes the content of a PDF visible signature field. A default Pattern of signature is provided by Universign. This pattern is customizable (see 2.5).

| Key | Type | Comments |
|---|---|---|
| name [O] | string | The name of the field. If the PDF already contains a named signature field, you can use this parameter instead of giving the coordinates (which will be ignored). If the name of this field does not exist in the document, the given coordinates will be used instead. |
| page [M] | int | The page on which the field must appear (starting at '1' for the first page). Pages are enumerated starting at 1. The value '-1' points at the last page. |
| x [M] | int | The field horizontal coordinate on the page. |
| y [M] | int | The field vertical coordinate on the page. |

A `DocSignatureField` structure have the same data of the `SignatureField` structure plus the following:

| Key | Type | Comments |
|---|---|---|
| signerIndex [M] | int | The index of the signer which uses this field. Signers are enumerated starting at 0. |
| patternName [O] | string | The name of the pattern. May be used if more than one pattern is set. The default value is "default". The magic value "invisible" means that the field will not be visible in the PDF. |
| label [O] | string | A label which defines the signature field. This label will be printed in the signature page if set. If a signer has more than one field on the same document, label becomes mandatory. |

## 4.7 SEPAData

The `SEPAData` data structure contains information needed to create a SEPA mandate PDF.

| Key | Type | Comments |
|---|---|---|
| rum [M] | string | A unique mandate identifier. |
| ics [M] | string | A unique creditor identifier. |
| iban [M] | string | The debtor International Bank Account Number. |
| bic [M] | string | The debtor Bank Identifier Code. |
| recurring [M] | boolean | Whether this SEPA mandate describe a recurring payment (`true`) or a single-shot payement (`false`). |
| debtor [M] | SEPAThirdParty | Information on the debtor. |
| creditor [M] | SEPAThirdParty | Information on the creaditor |

## 4.8 SEPAThirdParty

The `SEPAThirdParty` data structure is used to define information on both the debtor and the creditor of a SEPA mandate.

| Key | Type | Comments |
|---|---|---|
| name [M] | string | The full name of this debtor/creditor. |
| address [M] | string | The address of this debtor/creditor. |
| postalCode [M] | string | The postal code of this debtor/creditor. |
| city [M] | string | The city of this debtor/creditor. |
| country [M] | string | The country of this debtor/creditor. |

## 4.9 TransactionResponse

The `TransactionResponse` data structure is the response sent after a request for a transaction. This structure is used as a return value only, and will never be instantiated by users.

| Key | Type | Comments |
|---|---|---|
| url | string | The URL to the web interface for the first signer. |
| id | string | This transaction id. |

## 4.10 TransactionInfo

The `TransactionInfo` data structure describes the status of a transaction. This structure is used as a return value only, and will never be instantiated by users.

| Key | Type | Comments |
|---|---|---|
| status | string | The status of the transaction. The existing statuses are:<br>**ready** Signers can connect and sign.<br>**expired** The transaction has been requested more than 14 days ago. It will no more be available to signers.<br>**canceled** A signer has canceled the transaction. Signers will no more be able to connect to the service.<br>**failed** An error occurred during a signature. The signers won't be able to connect to the service.<br>**completed** All signers have successfully sign, the requester can retrieve the documents. |
| signerInfos | SignerInfo | A list of bean containing information about the signers and their progression in the signature process. |
| currentSigner | int | The index of current signer if the status of transaction is ready or who ended the transactions for other status. |
| creationDate | date | The creation date or last relaunch date of this transaction. |
| description | String | The description of the Transaction. |
| initiatorInfo | InitiatorInfo | A bean containing information about the requester of a transaction. |
| eachField | Boolean | whether the transaction was requested with requesting handwritten signature for each signature field or not. |
| customerId | String | This id can be specified when creating the transaction request and is used as additional information to identify the transaction. |
| transactionId | String | This id is generated when creating the transaction request and is the unique identifier of this transaction. |

## 4.11 SignerInfo

The `SignerInfo` data structure describes the status of a signer. This structure is used as a return value only, and will never be instantiated by users.

| Key | Type | Comments |
|---|---|---|
| status | string | The status of the signer. The existing statuses are:<br>**waiting** The signer has not yet been invited to sign. Others signers must sign prior to this user.<br>**ready** The signer has been invited to sign, but has not tried yet.<br>**accessed** The signer has accessed the signature service.<br>**code-sent** The signer agreed to sign and has been sent an OTP.<br>**signed** The signer has successfully signed.<br>**pending-id-docs** The signer has successfully signed and should send the `documents` to complete his registration.<br>**pending-validation** The signer has successfully signed and is pending RA validation.<br>**canceled** The signer refused to sign, or one of the previous signers canceled or failed its signature.<br>**failed** An error occurred during the signature. In this case, `error` is set.<br>The status of the observer. The existing statuses are:<br>**waiting** The observer has not yet been invited to access the document(s). Others signers must sign prior to this user.<br>**ready** The observer has been invited to access the document(s), but has not tried yet.<br>**accessed** The observer has accessed the document(s). |
| error | string | The error message in case of failure. |
| certificateInfo | CertificateInfo | A bean containing information about the certificate the signer used (or attempt to) to sign. If the signer has not signed yet or in some cases if an error occurs during the signature, an empty struct will be set for his certificatei. |
| url | string | The URL of the signature page. |
| email | string | The signer's email. |
| firstName | string | The signer's firstname. |
| lastName | string | The signer's lastname. |
| actionDate | date | the action date (signature, cancel or other). |
| refusedDocs | Integer[] | List of refused docs indexes. |

## 4.12 InitiatorInfo

The `InitiatorInfo` data structure describes the requester of a transaction. This structure is used as a return value only, and will never be instantiated by users.

| Key | Type | Comments |
|---|---|---|
| email | string | The requester's email. |
| firstName | string | The requester's firstname. |
| lastName | string | The requester's lastname. |

## 4.13 CertificateInfo

The `CertificateInfo` struct contains information about a certificate. This structure is used as a return value only, and will never be instantiated by users.

| Key | Type | Comments |
|---|---|---|
| subject | string | The certificate subject DN |
| issuer | string | The certificate issuer DN |
| serial | string | The certificate serial number |

## 4.14 TransactionFilter

**Warning: this object has an experimental structure. While it is considered production quality, it is subject to change in future versions.**

The `TransactionFilter` struct is a filter on transactions.

| Key | Type | Comments |
|---|---|---|
| requesterEmail | string | The returned list will contains only transactions whose requester is the UNIVERSIGN user identified by this e-mail address. If the current requester is not an admin, this field will be defaulted to his own e-mail address (i.e. a non-admin user can only list his own transactions). |
| profile | string | The name of the profile used to create the matching transactions. |
| notBefore | date | The matching transactions will have been created after this date. |
| notAfter | date | The matching transactions will have been created before this date. |
| startRange | int | An index, used together with **stopRange**, to define which subset of the result will be returned. Default value is **0**. |
| stopRange | int | An index, used together with **startRange**, to define which subset of the result will be returned. Its default and maximum value is **startRange + 1000**. |
| signerId | String | The matching transactions will contain a signer whose id matches this value. |
| notBeforeCompletion | date | The matching transactions that have been completed after this date. |
| notAfterCompletion | date | The matching transactions that have been completed before this date. |
| status | int | The matching transactions will contain one of those statuses. The existing statuses are:<br>**0** Ready.<br>**1** Expired.<br>**2** Completed.<br>**3** Canceled.<br>**4** Failed.<br>**5** Pending RA Validation. |
| withAffiliated | Boolean | The matched transactions of the affiliated organizations. If this filter is set to true, the own transactions won't be listed. |

## 4.15 StandaloneRegistrationRequest

The `StandaloneRegistrationRequest` data structure contains the information about the signer to register and allows to specify a signature profile to use.

| Key | Type | Comments |
|---|---|---|
| profile [O] | string | The name of the signature profile to use for the customization of the registration page. Signature profiles are set up by the UNIVERSIGN team. The default value is "default". |
| signer [M] | TransactionSigner | The signer that will be registered. |

# Error codes

**73002** An error occurred when signing the PDF document.

**73004** An error occurred when retrieving a stored private key.

**73010** The login and/or password are invalid.

**73011 & 73013** The requesting user doesn't have a valid signing account or its signing account is inactive.

**73014** When retrieving the document, the transaction id is missing or wrong.

**73020** Malformed request. Typically, an invalid value has been set, the `TransactionRequest` misses some mandatory elements or has some incompatible elements. The message attached to the RPC fault contain more accurate information.

**73025** The used transaction id or custom id is invalid.

**73027** Requesting an action on a transaction which status doesn't allow this action (e.g requesting the documents of an unfinished transaction).

**73040** Internal server error.

**73070** The requested document is being retrieved asynchronously from archival, it may take few hours. A further request will return the document.

# References

[1] *Draft ETSI TS 102 778*, PDF Advanced Electronic Signatures (PAdES), v0.0.17. Technical Specification, May 2009.

[2] *HTTP Authentication: Basic and Digest Access Authentication*
http://www.ietf.org/rfc/rfc2617.txt

[3] *Document Management - Portable Document Format - Part 1: PDF 1.7*
http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf