v5 Developer FAQ

eSignAnyWhere offers you an SOAP and a REST interface. It uses authentication and XML for the datastructures. See our API documentation for the basic information about our interfaces. Most programming languages offer you simple SOAP/REST interfaces moreover for the beginning you may start with SoapUI, a webservice testing tool.
There are two possibilities to customize the UI

- eSignAnyWhere UI for Signers: you can customize it by changing the template
- eSignAnyWhere UI for Users (Backoffice): this can just be done in a private cloud instance or on premise. For more information contact us.

You can configure email templates and languages for your eSignAnyWhere signers. It may also helpful if you have a look in the user guide settings section.
There is an envelope XML guide available.
You can prefill PDF form field with values via API. The following API methods are supported:

- `SendEnvelope_v1`
- `SendEnvelopeFromTemplate_v1`
- `CreateDraft_v1`
- `CreateDraftFromTemplate_v1`

XML Configuration for prefill PDF forms:

```
<envelope>
   ...
   <overrideFormFieldValues>
      <document docRef="1">
         <textBox name="formId">
            <value>textBoxValue</value>
         </textBox>
         <listBox name="formId">
            <selectedItems>
               <selectedItemId>selectedItem1</selectedItemId>
               <selectedItemId>selectedItem2</selectedItemId>
            </selectedItems>
         </listBox>
         <radioButtonGroup name="formId">
            <selectedItemId>selectedItem</selectedItemId>
         </radioButtonGroup>
         <checkBox name="formId">
            <isChecked>true|false|0|1</isChecked>
         </checkBox>
         <comboBox name="formId">
            <value>comboBoxValue</value>
         </comboBox>
      </document>
   </overrideFormFieldValues>
   ...
</envelope>
```

Yes, just add two ore more documents within the tag in the API call SendEnvelope_v1:

```
 <sspFileIds>
           <string>First document</string>
           <string>Second document</string>
 </sspFileIds>
```

After this configuration you can send an envelope as usual but with multiple documents.
First you have to create a draft (`CreateDraft_v1`) and configure the option to allow an external designer (`allowAgentRedirect`).

XML Configuration

```
<draftOptions>
        ...
        <allowAgentRedirect>true</allowAgentRedirect>
        <iFrameWhiteList>http://172.16.17.256;http://foo.org</iFrameWhiteList>
        ...
</draftOptions>
```

The option `allowAgentRedirect` enables an anonymous designer integration (without eSignAnyWhere Login) and `iFrameWhiteList` extends the HTTP header with a list to integrate in your web application or portal (via `X-FRAME-OPTIONS`).

The designer can be embedded by modifing the following string:

`http://www.significant.com/AgentRedirect/index?draftid=#envelopeid#`

If the draft is finished you can start the envelope.
If you don't want to send an email to a specific recipient, you just have to add the following configuration to the envelope XML (`<disableEmail>True</disableEmail>`):

Simple Example:

```
...
<recipient>
        <languageCode>en</languageCode>
        <eMail>recipient@email.com</eMail>
        <firstName>Firstname</firstName>
        <lastName>Lastname</lastName>
        ...
        <disableEmail>True</disableEmail>
        ...
        <!-- optional authentication methods for this recipient -->
</recipient>
...
```

eSignAnyWhere it calling the Callback URL 30 times. With the timeout this should be enough to recover if the called system is down for a few minutes.
You can modify the message and also the language of the SMS OTP sent via API.

Following types can be defined (inclusive max character length - GSM-7 (3GPP TS 23.038 / GSM 03.38) standard):

- smsAuthTransactionCodeId (SMS Authentication Message)
  - Text (excluding placeholders) must not be longer than 146 characters. (If any character not-compliant to the standard will be used, the limit is reduced to 52!)
- disposableCertificateEnrolAndSignSmsText (Disposable Certificate Message)
  - Text (excluding placeholders) must not be longer than 150 characters. (If any character not-compliant to the standard will be used, the limit is reduced to 60!)
- remoteCertificateSignSmsText (Remote Certificate Message)
  - Text (excluding placeholders) must not be longer than 150 characters. (If any character not-compliant to the standard will be used, the limit is reduced to 60!)
- otpSignatureSmsText (SMS OTP-Signature Message)
  - Text (excluding placeholders) must not be longer than 146 characters. (If any character not-compliant to the standard will be used, the limit is reduced to 52!)
- swissComSign
  - Text must not be longer than 160 characters. (If any character not-compliant to the standard will be used, the limit is reduced by half!)
- bankIdSignText
  - Text must not be longer than 160 characters. (If any character not-compliant to the standard will be used, the limit is reduced by half!)

Please note the following: The tag {tId} (Transaction ID) and {Token} (Token) must be defined in the message. If you do not define a language attribute, this will be used as fallback or specified language or recipient does not exists.

Please see the following sample configuration for an otp signature:

```
{
  "SspFileIds": [
    "e686d325-1234-1234-1234-f33cc522f38c"
  ],
  "SendEnvelopeDescription": {
  "Name": "test",
  "DisplayedEmailSender": "",
  "EnableReminders": true,
  "FirstReminderDayAmount": 5,
  "RecurrentReminderDayAmount": 3,
  "BeforeExpirationDayAmount": 3,
  "ExpirationInSecondsAfterSending": 2419200,
  "CallbackUrl": "",
  "StatusUpdateCallbackUrl": "",
  "LockFormFieldsAtEnvelopeFinish": false,
  "Steps": [
    {
      "OrderIndex": 1,
      "Recipients": [
        {
```

```
      "Email": "##Email##",
      "FirstName": "##Name##",
      "LastName": "##Name##",
      "LanguageCode": "en",
      "EmailBodyExtra": "",
      "DisableEmail": false,
      "AddAndroidAppLink": false,
      "AddIosAppLink": false,
      "AddWindowsAppLink": false,
      "AllowDelegation": true,
      "AllowAccessFinishedWorkstep": false,
      "SkipExternalDataValidation": false,
      "AuthenticationMethods": [],
      "IdentificationMethods": [],
      "OtpData": {
        "PhoneMobile": "##PhoneNumber##"
      }
    }
  ],
  "EmailBodyExtra": "",
  "RecipientType": "Signer",
  "WorkstepConfiguration": {
    "WorkstepLabel": "test",
    "SmallTextZoomFactorPercent": 100,

    "TransactionCodeConfigurations": [
         {
        "Id": "otpSignatureSmsText",
        "HashAlgorithmIdentifier": "Sha256",
        "Texts": [
           {
        "Language": "en",
        "Value": "This is for testing porpuses only  with the transactionId {tId}. Your code is: {Token}"
      }
        ]
      }
    ],
    "SignatureConfigurations": [],
    "ViewerPreferences": {
      "FinishWorkstepOnOpen": false,
      "VisibleAreaOptions": {
        "AllowedDomain": "",
        "Enabled": false
      }
    },
    "ResourceUris": {
      "DelegationUri": "https://demo.esignanywhere.net/Resource/Delegate"
    },
    "AuditingToolsConfiguration": {
      "WriteAuditTrail": true
    },
    "Policy": {
      "WorkstepTasks": {
        "PictureAnnotationMinResolution": 0,
        "PictureAnnotationMaxResolution": 0,
        "PictureAnnotationColorDepth": "Color16M",
        "SequenceMode": "NoSequenceEnforced",
        "PositionUnits": "PdfUnits",
        "ReferenceCorner": "Lower_Left",
        "Tasks": [
          {
            "Texts": [
              {
                "Language": "en",
                "Value": "Agreement text"
              },
              {
                "Language": "*",
                "Value": "Agreement text"
              }
            ],
```

```
      "Headings": [
        {
          "Language": "en",
          "Value": "Agreement Subject"
        },
        {
          "Language": "*",
          "Value": "Agreement Subject"
        }
      ],
      "IsRequired": false,
      "Id": "ra",
      "DisplayName": "ra",
      "DocRefNumber": 1,
      "DiscriminatorType": "Agreements"
    },
    {
      "PositionPage": 1,
      "Position": {
        "PositionX": 58.0,
        "PositionY": 593.0
      },
      "Size": {
        "Height": 80.0,
        "Width": 190.0
      },
      "AdditionalParameters": [
        {
          "Key": "enabled",
          "Value": "1"
        },
        {
          "Key": "completed",
          "Value": "0"
        },
        {
          "Key": "req",
          "Value": "1"
        },
        {
          "Key": "isPhoneNumberRequired",
          "Value": "0"
        },
        {
          "Key": "trValidityInSeconds",
          "Value": "60"
        },
        {
          "Key": "fd",
          "Value": ""
        },
        {
          "Key": "fd_dateformat",
          "Value": "dd-MM-yyyy HH:mm:ss"
        },
        {
          "Key": "fd_timezone",
          "Value": "datetimeutc"
        }
      ],
      "AllowedSignatureTypes": [
        {
          "TrModType": "TransactionCodeSenderPlugin",
          "TrValidityInSeconds": 300,
          "TrConfId": "otpSignatureSmsText",
          "IsPhoneNumberRequired": false,
          "Ly": "simpleTransactionCodeSms",
          "Id": "84dc05ed-1234-1234-1234-fbc776faa439",
          "DiscriminatorType": "SigTypeTransactionCode",
          "Preferred": false,
          "SignaturePluginConfigurationId": ""
```

```
                  }
                ],
                "UseTimestamp": false,
                "IsRequired": true,
                "Id": "1#XyzmoDuplicateIdSeperator#Signature_f96c9889-1234-1234-4c9c-f774add0d46b",
                "DisplayName": "",
                "DocRefNumber": 1,
                "DiscriminatorType": "Signature"
              }
            ]
          }
        }
      },
      "DocumentOptions": [
        {
          "DocumentReference": "1",
          "IsHidden": false
        }
      ]
    }
  ],
  "AttachSignedDocumentsToEnvelopeLog": false
}
}
```

The easiest way to implement the TransactionCodeConfiguration in SOAP is to call the GetAdHocWorkstepConfiguration_v1 to receive a default workstep for the document or you can modify it directly via API (see sample below).

```
  <TransactionCodeConfigurations>
    <!--Single TransactionCodeConfiguration.-->
    <TransactionCodeConfiguration trConfId="smsAuthTransactionCodeId">
      <!--Message used to send a transaction code to the client. The message has to contain the placeholder
'{tId}' for the transactionId and the placeholder '{Token}' for the token.-->
      <Message>Please authenticate yourself for the access to the envelope with the transactionId {tId}. Your
code is: {Token}</Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="smsAuthTransactionCodeId" language="en">
      <Message>Please authenticate yourself for the access to the envelope with the transactionId {tId}. Your
code is: {Token}</Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="smsAuthTransactionCodeId" language="it">
      <Message>Con riferimento alla transazione {tId}, per autenticarsi si prega di inserire il seguente CODICE
{Token}</Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="disposableCertificateEnrolAndSignSmsText">
      <Message>Please confirm the issuance of your disposable certificate and signature, referenced by
transactionId {tId}, with the OTP: </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="disposableCertificateEnrolAndSignSmsText" language="en">
      <Message>Please confirm the issuance of your disposable certificate and signature, referenced by
transactionId {tId}, with the OTP: </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="disposableCertificateEnrolAndSignSmsText" language="it">
      <Message>Conferma l'emissione del tuo certificato disposable, con riferimento alla transazione {tId}, e
della firma con lÂ´OTP </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="remoteCertificateSignSmsText">
      <Message>Please sign the document, referenced by transactionId {tId}, using the OTP: </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="remoteCertificateSignSmsText" language="en">
      <Message>Please sign the document, referenced by transactionId {tId}, using the OTP: </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
    <TransactionCodeConfiguration trConfId="remoteCertificateSignSmsText" language="it">
      <Message>Firma il documento, con riferimento alla transazione {tId}, usando lÂ´OTP </Message>
      <hashAlgorithmIdentifier>Sha256</hashAlgorithmIdentifier>
    </TransactionCodeConfiguration>
  </TransactionCodeConfigurations>
```

If you have a finished envelope (all recipients finished) and the state of the evelope is still “in Progress”, the reason could be the post processing (the callback). The callback expects a HTTP 200 and if an error is retured it tries to call the callback after some time again (up to 30 times). This could delay the “finished” or “error” of the envelope.
The problem is, that the envelope is not yet in the correct status.
The current status is “Started”. This means, that the link for the first recipient is not yet being created.

You have to wait until the status is “InProgress”.

Another option would be to either…

- set “**suppressEmails**” on the recipient
- or to check “**Prevent emails from being sent :**” on the organization (using the Admin Web site)

Then the link will be generated immediately when calling **SendEnvelope_v1**.

If you use regular notifications (emails), the link is sent to the recipient as soon as the Workstep is ready.
You can define a reading task, so that the signer has to confirm the reading of the envelope. Details about the configuration you find in the Reading Task Guide.
Yes, you can store your own meta data in the envelopes.

```
<envelope>
   <metaData>
      <element>custom data</element>
   </metaData>
</envelope>
```

The `metaData` element allows you to store additional, non-eSAW-data (e.g. for archiving) directly in the envelope. You can retrieve this information via `getEnvelopeById` call. An example of metaData is to store data for the archiving system in the envelope. The callback-integrating solution then can download the files (PDF & Audit-Trail) and store them directly in the archive.

Envelope	Bulk

• Draft
  ◦ Not sent envelope
• Canceled
  ◦ Envelope which was canceled by the **sender**. (final state).
• Completed
  ◦ Envelope which does not need any more actions (final state).
• Expired
  ◦ Envelope reached expiration date. Can be restarted with a new expiration date.
• Rejected
  ◦ Rejected by one of the envelope **recipient**.
• Template
  ◦ Template which can be used to create envelopes with a given configuration.
• ActionRequired
  ◦ Waiting for your action.
• WaitingForOthers
  ◦ Waiting for actions of other recipients.
• ExpiringSoon
  ◦ The expiration date will soon the be reached.
• InProgress
  ◦ The envelope is in progress, waiting for next recipient in the order to do his/her action.
	• Draft
	  ◦ Not sent envelope
	• Canceled
	  ◦ Envelope which was canceled by the **sender**. (final state).
	• Completed
	  ◦ Envelope which does not need any more actions (final state).
	• Expired
	  ◦ Envelope reached expiration date. Can be restarted with a new expiration date.
	• Rejected
	  ◦ Rejected by one of the envelope **recipient**.
	• Template
	  ◦ Template which can be used to create envelopes with a given configuration.
	• Started
	  ◦ The envelope was started but needs to setup further metrics to the into status InProgress
	• CompletedWithWarnings
	  ◦ Warnings concerning long lived disposable
	• BulkCompleted
	  ◦ All envelopes of a bulk are completed
	• BulkPartlyCompleted
	  ◦ Not all envelopes of a bulk are completed
	• InProgress
	  ◦ The envelope is in progress, waiting for next recipient in the order to do his/her action.

eSAW supports to send out links for the SIGNificant products automatically via notifications. Therefore you just have to add to the recipient configuration (XML) the following parameters:

```
<envelope>
  <steps>
    <step>
      <recipients>
        <recipient>
          <addAndroidAppLink>0</addAndroidAppLink> <!-- 0 or 1 -->
          <addIosAppLink>0</addIosAppLink> <!-- 0 or 1 -->
          <addWindowsAppLink>0</addWindowsAppLink> <!-- 0 or 1 -->
        </recipient>
      </recipients>
    </step>
  </steps>
</envelope>
```

Otherwise you can connect one workstep of eSignAnyWhere with one of the SIGNificant Apps. First you call the `GetEnvelopeById_v1` with the `envelopeID` you got in sendEnvelope_v1. In the result you will find the `workstepRedirectionURL`. This URL forwards the SignAnywhere Viewer (Web-Client), but with the additional parameter `&responseType=returnWorkstepId` it returns the workstepId. Example:

```
https://demo.xyzmo.com/workstepredirector/sign?identifier=8WNDxmUVr5V/aV1AAN49xjKuVsHMQEIQVuM
/ktLNw1jOfWgaovF2mDg3uW9JJbp5Q/k7Yz92eoo=&responseType=returnWorkstepId
```

With this WorkstepId you can now connect the SIGNificant product to the document. If the document is finsihed the workflow continues automatically.

Other parameters are:

- `responseType=redirectToViewer` – redirects to SAW Viewer (default)
- `responseType=redirectToAndroidApp` – redirects to Android App
- `responseType=redirectToIOsApp` – redirects to iOS App
- `responseType=redirectToWindowsApp` – redirects to Windows App

- `responseType=returnWorkstepId` – returns the WorkstepId for other integration types

If you want to get a callback on specific events, e.g. when a signer rejects the agreement, you can use the following guide.
If you upload a PDF/A document to eSignAnyWhere it stays through the workflow a PDF/A valid document. If you are starting with a non-PDF/A document, the final document will be also a non-PDF/A document.
Yes, you may use our SigStrings to place signature fields in documents. You just have to type a string (the simplest version: `` `sig` ``) in the document and eSignAnyWhere is placing a signature field for you automatically. Here you get more information about our placeholders.

If you want to use more complex tags (e.g. for form fields, radio buttons, etc.) you may be interested in our advanced tags. This can be found in our Placeh older Use Case and moreover a look into the *PrepareSendEnvelopeSteps_v1* function. This function can be helpful for integrations.
This is typically caused by an outdated Adobe Reader with not update-to-date certificates. Please install a new version or perform an update of the certificates (Settings > Trust Manager > Update AATL/EUTL).