API Reference - SOAP

deprecated

SOAP API is deprecated

Please note that we announced deprecation of the SOAP API already a while ago in the product release notes. This documentation is therefore marked as DEPRECATED.
See Api Reference - Introduction REST for the REST API reference which should be considered as the successor of the SOAP API.

Repeating here also how SOAP discontinuation was published in the release notes:
We declared SOAP as deprecated and therefore SOAP will not be included in versions after 21.76 (already postponed by one year, initially the 20.76 was announced). Latest release including SOAP API for eSAW will be 21.76, released in spring 2022 and with the software maintenance on 21.76 until spring 2024.
Therefore, we recommend REST technology for integration. Please see also the migration guide.

The Api Reference Guide explains the basic about the SOAP API. On this page you can find information about the authorization, envelope callbacks and errors. Moreover, you can find the documentation about the following two eSAW SOAP API: eSAW API description and eSAW UserManagement API description.

Your eSignAnyWhere benefits

Introduction

On this page you will find the *eSAW API* description and the *eSAW UserManagement API* description. Morverover we start with a basic overview of the API.

Before you use the API Reference, we recommend you to read the API Documentation, to get an overview about our programming interface, data types and basic concepts. If you are looking for examples, we recommend the Hello World Tutorial, the SoapUI Sample, Postman Sample and our Stories and Examples (with all XML calls).

- The API can be used using SOAP and JSON/REST
- SOAP: All complex types are passed as an XML string
- SOAP: The API does NOT use SoapFaults / Exceptions, instead the result contains information if the request was successful or not
- Authorization is required (Application Key)
- SOAP: All method results have the same basic scheme:
    - Success

```
<apiResult version="1.2.3.4">
  <baseResult>ok</baseResult>
  <okInfo>
    <!-- actual method result -->
  </okInfo>
</apiResult>
```

    - Error

```
<apiResult version="1.2.3.4">
  <baseResult>failed</baseResult>
  <errorInfo>
    <error>ERR....</error>
    <errorMsg>error message</errorMsg>
  </errorInfo>
</apiResult>
```

SOAP calls with XML / REST with JSON

SOAP uses XML, which have to be encoded via HTML special charaters (“<” in “<” and “>” in “>”). Higher programming languages takes automatically care of the conversion, so just in lower languages it is required (or also SoapUI). If this encoding is not done you will receive a *HTTP 400 Bad Request* error.

REST uses JSON. We recommend to have a look in the sample code first, otherwise Swagger UI is available for integration, which is available under /api (demo.xyzmo.com/api or esignanywhere.net/api).

> We recommend to have a look into the Hello World Tutorial, the SoapUI Sample (SOAP), Postman Sample (REST) and our Stories and Examples (with all XML calls). Moreover our eSignAnyWhere_SampleCode_.zip (depricated, SOAP-only) contains helpful integrations.

Authorization

For authentication, you'll need an api token. You can find this information in Settings / Api Tokens and Apps. Note that all API methods require authentication. Start with the Hello World, SoapUI Tutorial or REST tutorial using Postman (REST) and you might have a look in the envelope XML guide. Please note, that the email is automatically the user, who sends the envelope.

This section covers a SOAP-integration. The REST/JSON API is similar and you might have a look into Swagger and the Postman Sample. As written in the API Documentation, you have to access API calls with an authorization information. The following code shows a api token authorization. ≥20.42

```
<authorization>
<apiToken >hizit4enf8ellb6b5h12345bgt8mzf1abcedf27jys91hetvesabcd9fdb312348bk</apiToken >
</authorization>
```

The API token is a user specific secret which should not be shared with other users. We recommend to create different API keys for different application integrations, to avoid configuring the same key in various integration systems. This allows, e.g. in case of sharing a key by mistake, to disable one key while keeping other integrations working with their existing configuration. Note that the Authorization XML is passed on as string parameter, not as directly embedded XML – so encode the the < with < and > with > when directly testing e.g. in a SOAP test environment (no extra encoding required when your SOAP library encodes the parameters automatically, e.g. in .NET or Java)

> You can also authorize with the organizationKey and the userLoginName. Please see the following configuration:
>
> Note: The organizationKey can also be found in Settings / Api Tokens and Apps
>
> ```
> <authorization>
>   <organizationKey><!-- actual organization API key --></organizationKey>
>   <userLoginName><!-- actual login e-mail address --></userLoginName>
> </authorization>
> ```

Callbacks

The API allows the definition of several callbacks.

Please note, that only the envelope callback (directly from eSignAnyWhere) is fired, when the envelope is in a final state. The *status update callback* is fired by a sub-component and you may require to wait a post-processing time that the envelope reaches its final state.

Note: Please send back the HTTP 200 immediately! If you are not returning the HTTP 200 immediately, eSignAnyWhere tries to recall the URL.
Attention: After some attempts the envelope will not be finished!

> Please note the following information which applies to all callbacks:
> Consider, that our system expects the full callback url, including the parameter list you expect, with the placeholders that should be replaced by values at runtime. You can also add your own paramter for that envelope (e.g. internal references). Moreover, on our shared SaaS and private SaaS environments only HTTPS (default port 443) callbacks are allowed.

Envelope Callback

This is the basic callback (<CallbackUrl />), which is fired if the envelope reaches a final state (completed, rejected). If you integrate eSAW, please have a look at the *Envelope Status Callback* (directly below documented), because it might deliver more details about the envelope and might so be more useful for integrating.

Placehoder

- **##EnvelopeId##**
- **##Action##**
  - envelopeFinished : when an envelope was finished (completed or rejected)

Sample:

```
https://www.mycallback.at?envelope=##EnvelopeId##
```

Envelope Status Callback

Envelopes status callbacks (<statusUpdateCallbackUrl />) are fired, based on envelope events/actions. There are also detailed callbacks available based on events (see this guide).

Placehoder for callback URL:

- **##EnvelopeId##**
- **##Action##**
  - workstepFinished : when the workstep was finished
  - workstepRejected : when the workstep was rejected
  - workstepDelegated : whe the workstep was delegated
  - workstepOpened : when the workstep was opened
  - sendSignNotification : when the sign notification was sent
  - envelopeExpired : when the envelope was expired
  - workstepDelegatedSenderActionRequired : when an action from the sender is required because of the delegation

Sample:

```
https://www.mycallback.at?envelope=##EnvelopeId##&action=##Action##
```

Sample with custom parameter "*internalid*":

```
https://www.mycallback.at?envelope=##EnvelopeId##&action=##Action##&internalid=1234
```

Draft Callbacks

Draft callbacks are fired, if a draft is used or deleted. The draft callback is set in the *DraftOptionsXML (<AfterSendCallbackUrl />)* via *CreateDraft_v1*.

- **##DraftId##**
- **#Action##**
  - draftDiscarded
  - draftSent

Sample:

```
https://www.mycallback.at?draft=##DraftId##
```

Errors

The SOAP response is a XML datastructure with the state of the response and the response itself. In **`baseResult`** you see the state of the call (ok/failed) and in **`errorInfo`** or **`okInfo`** the response of your request. In case of an error a helpful message is in the **`errorInfo`**. A list of error codes is also available.

In general, our SOAP endpoint return a HTTP success (HTTP/200) even in case of errors, when your call reached the server. The response then has baseResult false, and contains an errorInfo with error (the error code) and errorMsg (human readable error information).

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soap:Body>
      <GetAdHocWorkstepConfiguration_v1Response xmlns="http://www.eSignAnyWhere.com/">
         <GetAdHocWorkstepConfiguration_v1Result><![CDATA[<apiResult version="2.1.260.5831">
  <baseResult>failed</baseResult>
  <errorInfo>
    <error>ERR0009</error>
    <errorMsg>ERR0009: Authorization of user '...' failed</errorMsg>
  </errorInfo>
</apiResult>]]></GetAdHocWorkstepConfiguration_v1Result>
      </GetAdHocWorkstepConfiguration_v1Response>
   </soap:Body>
</soap:Envelope>
```

or:

```
<apiResult version="2.2.458.6616">
  <baseResult>failed</baseResult>
  <errorInfo>
    <error>ERR0011</error>
    <errorMsg>ERR0011: Parameter apiAuthorizationXml is incorrect: Failed to parse</errorMsg>
  </errorInfo>
</apiResult>
```

Some reasons for errors:

- ERR0008: File '<file id>' is missing
- ERR0009: Authorization of user '…' failed
- ERR0011: Parameter envelopeDescriptionXml is incorrect
- ERR0013: Action on envelope '…' is not allowed for envelope state 'Expired'; You cannot download an expired envelope. Check if you use a wrong envelope id, or if the envelope lifetime was too short.
- ERR0014: User '…' is not allow to perform this action: Document count of organization with id '…' exceeded: XX used Documents / 1 to be added / XX Limit. Check if you should upgrade your license, or if you already got a new license voucher which you didn't issue yet. Check if you are still using the Evaluation License. You get an HTTP/1.1 400 Bad Request when your call does not reach the SOAP endpoint. Most common reasons therefore:
    - The string parameters, which contain xml code, did not encode the < to < and so your SOAP request is invalid (e.g. the authorizationXML)

eSAW API description for SOAP

SOAP API endpoint is https://www.significant.com/api.asmx
WSDL can be found here: https://www.significant.com/api.asmx?WSDL

Methods for sending an envelope or to create a draft of an envelope

An envelope can consist of multiple documents and can contain multiple recipients for signing (steps). Your backend application will provide the documents for the envelope and the configuration of the workflow and worksteps. Once the envelope has been completed or rejected, your backend will be notified from eSignAnyWhere about this (callback). Therefore you can provide a callback url, so you don't need to poll the eSAW system.

The pattern for sending an envelope

Upload all documents for this envelope (`UploadTemporarySspFile_v1`)
Optional: Analyze all documents to get a default configuration. (`GetAdHocWorkstepConfiguration_v1`)
Optional: Prepare configuration for every signing step in the envelope, by adjusting the xml accordingly.
Start the workflow of the envelope by providing the general configuration of the envelope and the configuration for every step (`SendEnvelope_v1`)

The pattern for creating a draft of an envelope:

Upload all documents for this envelope (`UploadTemporarySspFile_v1`)
Create the draft by providing the general configuration of the envelope (`CreateDraft_v1`)
You can redirect the user to the designer of eSignAnyWhere to manually position and assign fields for the recipients pf this envelope.

Methods for downloading the documents of an completed envelope

Once you get the notification from eSignAnyWhere that an envelope has been completed, you can download the signed documents and the log of the envelope.

The pattern for downloading documents of an completed envelope

Check the status of that Envelope, to be sure it is completed and not rejected. (`GetEnvelopeById_v1`)

Download all the signed documents and the envelope log. (`DownloadCompletedDocument_v1`)

Methods for managing an envelope

`SendReminders_v1`: Send reminders to all recipients that currently can sign the envelope
`CancelEnvelope_v1`: Cancels the envelope
`DeleteEnvelope_v1`: Delets an envelope or a draft of an envelope
`RestartEnvelope_v1`: Restarts an expired envelope

Reference eSAW API

The "cancel" operation is related to the workflow. Cancel means, an already sent signing process will be stopped and canceled. The envelope is not deleted (the same applies for a draft, as it hasn't been started yet).
For this call you just need the templateId. The response of this call is a new SspFileId and the workstep configuration of the template. Therefore, with this call you get all information (workstep configuration and the document) to send an envelope afterwards.
Creates a draft with the given parameters.

Parameters

sspFileIds: Reference to files already uploaded to the SSP server e.g. using the uploadTemporarySspFileV1 call

envelopeDescriptionXml: The envelope configuration (<envelope>…</envelope>). In a draft, defining a workstep configuration is not supported.

draftOptionsXml: With the draft options, the redirect url and the callback url can be specified. See also callbacks.

```
<draftOptions>
    <afterSendRedirectUrl>http://www.mycallback.at/DraftAfterSendRedirect?
envelope=##EnvelopeId##&action=##Action##</afterSendRedirectUrl>
    <afterSendCallbackUrl>http://www.mycallback.at/DraftAfterSendCallback?
envelope=##EnvelopeId##&action=##Action##</afterSendCallbackUrl>
    <redirectPolicy>ToDesigner</redirectPolicy>
</draftOptions>
```

Result

DraftId
Creates the Draft on the server. See Implementation for information how to integrate the draft in your environment. For more details of parameter **envelope DescriptionXml** see: The Envelope XML
After an envelope has been completed (i.e. all signers signed, and the result was downloaded and stored in any kind of document management system), implementors should take care about deleting the envelope from the SignAnyWhere servers. This should be done as general system cleanup best-practise, and also for security issues (there were links sent to recipients containing the envelope reference, why do you want to keep this links alive when documents were moved to a document management system where you probably have a different permission strategy)
Removes the previously uploaded document (using UploadTemporarySspFile_v1) to the SIGNificant Server Platform again.
The file id has to be retrieved from *GetEnvelopeById_v1*. Attention: just documents of finished envelopes can be downloaded. The returned file id from the uploading call will cause an error (invalid file id).
Please use the FindEnvelopes_v2 function.

```
<findEnvelopesDescriptor>
  <!-- all search parameters are optional -->
  <status>Draft|Started|InProgress|Canceled|Completed|Expired|Rejected|Template|ActionRequired|
WaitingForOthers|ExpiringSoon|Active</status> <inStatusSinceDays>30</inStatusSinceDays>
  <sentAfterDate>2017-02-08T12:18:37.2415657Z</sentAfterDate>
  <!-- all dates can also be "date only" -->
  <sentBeforeDate>2017-05-10T11:18:37.2415657Z</sentBeforeDate>
  <!-- search text searches for email subject, email body, envelope name, envelope description, sender user or
envelope recipient; use with care because of high system impact -->
  <searchText>some text</searchText>
  <hasSender>
    <eMail>Marius.Gheorghe@eflowauto.test</eMail>
  </hasSender>
  <hasSigner>
    <eMail>Marius.Wenny@eflowauto.test</eMail>
  </hasSigner>
  <hasRecipient>
    <eMail>Fabian.Wenny@eflowauto.test</eMail>
    <!-- if just one recipient is searched for, you can leave the "eMail" tag because of backwards
compatibility -->
  </hasRecipient>
  <waitingForRecipient>Fabian.Jordan@eflowauto.test</waitingForRecipient>
</findEnvelopesDescriptor>
```

resultFormat:
empty, “OnlyIdentifiers”, “Extended”

Important note: searchText should be used with care, because it will cause a high load on the system (string based search).
Sends you back an adhoc workstep configuration for a given document.

Usage

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:esig="http://www.
eSignAnyWhere.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <esig:GetAdHocWorkstepConfiguration_v1>
         <!--Optional:-->
         <esig:authorizationXml>
          <authorization>
```

```
          <organizationKey>##ORGANIZATION-KEY##</organizationKey>
          <userLoginName>##USER-LOGON##;/userLoginName>
         </authorization>
         </esig:authorizationXml>
         <!--Optional:-->
         <esig:sspFileIds>
            <!--Zero or more repetitions:-->
            <esig:string>##FILE-ID##</esig:string>
         </esig:sspFileIds>
         <!--Optional:-->
         <esig:adHocWorkstepConfiguration>
         <AdhocWorkstepConfiguration>
    <!-- not relevant-->
    <WorkstepLabel>workstepLabel</WorkstepLabel>
    <!-- not relevant-->
    <SmallTextZoomFactorPercent>100</SmallTextZoomFactorPercent>
    <!-- not relevant-->
    <WorkstepTimeToLiveInMinutes>0</WorkstepTimeToLiveInMinutes>
    <!--Configure the actions done by the server and the by the clients when the workstep is finished.-->
    <FinishAction>
        <!-- not relevant-->
        <ServerAction callSynchronous="0"></ServerAction>
        <!--A client action specifies the redirect, when a recipient clicks on finish.-->
        <ClientAction clientName="SIGNificant SignAnywhere" closeApp="0" RemoveDocumentFromRecentDocumentList="
0" CallClientActionOnlyAfterSuccessfulSync="1">https://www.significant.com</ClientAction>
    </FinishAction>

    <!--Configure the adhoc workstep creation-->
    <NoSequenceEnforced>0</NoSequenceEnforced>

    <!-- Define default properties of signature fields / tasks-->
    <SigTemplate>
        <!--The elements width in points-->
        <width>50.5</width>
        <!--The elements height in points-->
        <height>100.5</height>
        <!--Parameter defining the signature type. Possible values: 'BiometricSignature', 'LocalCertificate',
'Picture', 'TransactionCode', 'TransactionCodeAndBiometricSignature', 'TransactionCodeAndLocalCertificate',
'TransactionCodeBiometricSignatureAndLocalCertificate' and 'BiometricSignature_and_LocalCertificate'. -->
        <param name="sigType">Picture</param>
        <!--Parameter that refines sigType "Picture": a list, seperated by "," from these values: Draw2Sign,
Type2Sign,Click2Sign -->
        <param name="allowedCapturingMethods">Draw2Sign,Type2Sign</param>

    </SigTemplate>
    <!--Configuration for parsing the form fields. Possible values: '1' parse the form fields, '0' do not parse
form fields-->
    <!--Attribute 'mapRequiredFieldsToRequiredTask': set the form filling task required when some of the fields
are required. Possible values: '1' required forms lead to required tasks, '0' required fields do not enforce
the task to be required-->
    <!--Attribute 'formsGrouping': Specify how the parsed forms should be grouped into tasks. Possible values:
'PerPage' all forms on one page are grouped to one forms group, 'PerDocument' all forms of one document are
grouped to one forms group, 'PerEnvelope' all forms of all documents inside the envelope are grouped to one
forms group-->
    <ParseFormFields mapRequiredFieldsToRequiredTask="0" formsGrouping="PerDocument">1</ParseFormFields>
    <!--If the workstep is not generated by hand but automatically generated by the Workstep Controller
AdhocPolicies are specified-->
    <AdhocPolicies>
        <!-- not relevant-->
        <AllowModificationsAfterSignature>1</AllowModificationsAfterSignature>
    </AdhocPolicies>
    <!--Configurate the signatures for this workstep. One default configuration has to be defined. The default
configuration is used for flatten signatures, adhoc signatures and signature fields which do not reference a
special signature plugin configuration. The default configuration does not contain the attribute 'spcId'. If
the attribute 'spcId' is defined the signature plugin configuration does only apply to signature fields
referencing the configuration by specifiying <param name="spcId">id</param>.-->
    <signaturePluginConfiguration>
        <!--Configurate the signature properties-->
        <PdfSignatureProperties_V1>
            <!--Should the signatures be pdfA conformant. Note this setting does not convert a document into
pdfA, it only keeps it pdfA conformant if it already is. Possible values: '1' sign pdfA conformant - in this
```

```
case the file size will be bigger than without pdfA, '0' do not sign pdfA conformant.-->
            <PdfAConformant>0</PdfAConformant>
            <!--Defines if the signature should be PAdES part 4 compliant. Possible values: '1' sign the
document PAdES part 4 compliant, '0' sign the document with standard pdf signature. Default value: '0'-->
            <PAdESPart4Compliant>1</PAdESPart4Compliant>
            <!--Defines if the certificate chain for the signing certificate should be embedded into the
signature. Possible values: '1' include the certificate chain, '0' do not include the certificate chain.
Default value: '0'-->
            <IncludeSigningCertificateChain>1</IncludeSigningCertificateChain>
            <!--Defines if and how the revocation information for the signing certificate chain should be
embedded. Possible values: 'DoNotInclude' no revocation information is included, 'Include' the revocation
information has to be included, if not possible the signature throws an exception, 'TryToInclude' if the
revocation information can be fetched, it should be included, if not the signature is done without revocation
information. Information about the signatures where the revocation information could not be included is saved
into the WorkstepStatus, 'CheckRevocationIncludeOcsp' the revocation information has to be included when it is
an OCSP, if checking of the revocation (OCSP or CRL) fails an exception is thrown. Information about the
signatures where the revocation information could not be included is saved into the WorkstepStatus-->
            <SigningCertificateRevocationInformationIncludeMode>Include<
/SigningCertificateRevocationInformationIncludeMode>
        </PdfSignatureProperties_V1>
        <!--Configurate the cryptographic data-->
        <PdfSignatureCryptographicData_V1>
            <!--The hash algorithm used for the signatures. Possible values: 'Sha1', 'Sha256', 'Sha512'-->
            <SignatureHashAlgorithm>Sha256</SignatureHashAlgorithm>
            <!--The description of the signing certificate. More than one SigningCertificateDescriptor can be
defined by adding this node more than once. If more SigningCertificateDescriptors are present, these
configurations are used as backup if the selected SigningCertificateDescriptor is not working. For example if
no revocation information can be retrieved although it should be included into the signature.-->
            <SigningCertificateDescriptor>
                <!--The certificates identifier-->
                <Identifier>3b777446a35fca027cbed5f69e24995945a611cb</Identifier>
                <!--The certificate identifier type. Possible values: 'Subject', 'Sha1Thumbprint'-->
                <Type>Sha1Thumbprint</Type>
                <!--The cryptographic service provider to locate the certificate. Possible values: 'default'
uses the servers certificate store, 'custom' uses the custom signature action-->
                <Csp>Default</Csp>
            </SigningCertificateDescriptor>
        </PdfSignatureCryptographicData_V1>
    </signaturePluginConfiguration>

    <!--Configure the signature string parsing pattern: Text in the document will be parsed for this pattern
and if found, a signature task is generated. -->
    <SigStringParsingConfiguration>
        <!--Defines a signature string to parse. Tag can be present more than once-->
        <SigStringsForParsing>
            <!--The start pattern of the signature string if it has a start and end pattern. Otherwise the
whole word to parse-->
            <StartPattern>`sig</StartPattern>
            <!--End pattern if needed, otherwise empty-->
            <EndPattern>`</EndPattern>
            <!--Define if the signature strings should be cleared from the document. Possible values: '1'
remove the signature strings from the document, '0' do not change the document-->
            <ClearSigString>1</ClearSigString>
            <!--Define if only the entire word should be searched. For example if start pattern is 'signature'
only 'signature' but not 'signaturepad' is found. This option does only effect signature string without end
patterns. Possible values: '1' search only the entire word, '0' search words containing the pattern as well.-->
            <SearchEntireWordOnly>1</SearchEntireWordOnly>
        </SigStringsForParsing>
    </SigStringParsingConfiguration>
    <!--Defines general policies for this workstep-->
    <GeneralPolicies>
        <!--Is the client allowed to save the workstep document-->
        <AllowSaveDocument>1</AllowSaveDocument>
        <!--Is the client allowed to save the audittrail document-->
        <AllowSaveAuditTrail>1</AllowSaveAuditTrail>

        <!-- not relevant-->
        <AllowFinishWorkstep>1</AllowFinishWorkstep>
        <!--Is the client allowed to reject the workstep-->
        <AllowRejectWorkstep>1</AllowRejectWorkstep>
```

```
        <AllowAdhocPdfAttachments>1</AllowAdhocPdfAttachments>
    </GeneralPolicies>

    <ViewerPreferences>
        <ShowVersionNumber>1</ShowVersionNumber>
        <EnableThumbnailDisplay>1</EnableThumbnailDisplay>
        <EnableWarningPopupOnLeave>1</EnableWarningPopupOnLeave>
        <WarningPopupDisplayAfter>Always</WarningPopupDisplayAfter>
        <GuidingBehavior>GuideRequiredAndOptionalTasks</GuidingBehavior>

    </ViewerPreferences>

</AdhocWorkstepConfiguration>
         </esig:adHocWorkstepConfiguration>
      </esig:GetAdHocWorkstepConfiguration_v1>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <GetAdHocWorkstepConfiguration_v1Response xmlns="http://www.eSignAnyWhere.com/">
         <GetAdHocWorkstepConfiguration_v1Result><![CDATA[<apiResult version="2.1.260.5831">
  <baseResult>ok</baseResult>
  <okInfo>
    <workstepConfiguration>
      <WorkstepLabel>workstepLabel</WorkstepLabel>
      <SmallTextZoomFactorPercent>100</SmallTextZoomFactorPercent>
      <WorkstepTimeToLiveInMinutes>0</WorkstepTimeToLiveInMinutes>
      <FinishAction>
        <ServerAction callSynchronous="0" />
        <ClientAction clientName="SIGNificant SignAnywhere" closeApp="0" RemoveDocumentFromRecentDocumentList="
0" CallClientActionOnlyAfterSuccessfulSync="1">https://www.significant.com</ClientAction>
      </FinishAction>
      <signatureTemplate>
        <version>1.2.0.2</version>
        <positionUnits>PdfUnits</positionUnits>
        <positionReferenceCorner>Lower_Left</positionReferenceCorner>
      </signatureTemplate>
      <pdfForms isEditingAllowed="1" />
      <ViewerPreferences>
        <ShowVersionNumber>1</ShowVersionNumber>
        <EnableThumbnailDisplay>1</EnableThumbnailDisplay>
        <EnableWarningPopupOnLeave>1</EnableWarningPopupOnLeave>
        <WarningPopupDisplayAfter>Always</WarningPopupDisplayAfter>
        <GuidingBehavior>GuideRequiredAndOptionalTasks</GuidingBehavior>
        <FormFieldsGuidingBehavior>AllowSubmitAlways</FormFieldsGuidingBehavior>
        <FinishWorkstepOnOpen>0</FinishWorkstepOnOpen>
      </ViewerPreferences>
      <Policy version="1.0.0.2">
        <GeneralPolicies>
          <AllowSaveDocument>1</AllowSaveDocument>
          <AllowSaveAuditTrail>1</AllowSaveAuditTrail>
          <AllowRotatingPages>1</AllowRotatingPages>
          <AllowAppendFileToWorkstep>0</AllowAppendFileToWorkstep>
          <AllowAppendTasksToWorkstep>0</AllowAppendTasksToWorkstep>
          <AllowEmailDocument>0</AllowEmailDocument>
          <AllowPrintDocument>0</AllowPrintDocument>
          <AllowFinishWorkstep>1</AllowFinishWorkstep>
          <AllowRejectWorkstep>1</AllowRejectWorkstep>
          <AllowUndoLastAction>0</AllowUndoLastAction>
          <AllowColorizePdfForms>0</AllowColorizePdfForms>
          <AllowAdhocPdfAttachments>1</AllowAdhocPdfAttachments>
          <AllowAdhocSignatures>0</AllowAdhocSignatures>
          <AllowAdhocStampings>0</AllowAdhocStampings>
          <AllowAdhocFreeHandAnnotations>0</AllowAdhocFreeHandAnnotations>
          <AllowAdhocTypewriterAnnotations>0</AllowAdhocTypewriterAnnotations>
          <AllowAdhocPictureAnnotations>0</AllowAdhocPictureAnnotations>
```

```
          <AllowAdhocPdfPageAppending>0</AllowAdhocPdfPageAppending>
        </GeneralPolicies>
        <WorkstepTasks SequenceMode="SequenceOnlyRequiredTasks" originalSequenceMode="
SequenceOnlyRequiredTasks" />
        <AdhocPolicies>
          <AllowModificationsAfterSignature>1</AllowModificationsAfterSignature>
        </AdhocPolicies>
      </Policy>
      <timeCreated>2017-01-17T09:55:13.2332816Z</timeCreated>
      <signaturePluginConfiguration>
        <PdfSignatureProperties_V1>
          <PdfAConformant>0</PdfAConformant>
          <PAdESPart4Compliant>1</PAdESPart4Compliant>
          <IncludeSigningCertificateChain>1</IncludeSigningCertificateChain>
          <SigningCertificateRevocationInformationIncludeMode>Include<
/SigningCertificateRevocationInformationIncludeMode>
        </PdfSignatureProperties_V1>
        <PdfSignatureCryptographicData_V1>
          <SignatureHashAlgorithm>Sha256</SignatureHashAlgorithm>
          <SigningCertificateDescriptor>
            <Identifier>##SIGNATURE-HASH##</Identifier>
            <Type>Sha1Thumbprint</Type>
            <Csp>Default</Csp>
          </SigningCertificateDescriptor>
        </PdfSignatureCryptographicData_V1>
      </signaturePluginConfiguration>
      <TransactionCodeConfigurations>
        <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="de">
          <Message>Bitte authentifizieren Sie sich für den Zugriff auf die Dokumentenmappe der Transaktion
{tId}. Ihre TAN lautet: {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>
      </TransactionCodeConfigurations>
    </workstepConfiguration>
  </okInfo>
</apiResult>]]></GetAdHocWorkstepConfiguration_v1Result>
      </GetAdHocWorkstepConfiguration_v1Response>
   </soap:Body>
</soap:Envelope>
```


Get the status of an envelope by its envelopeId.

The status of an envelope can be:

- Draft
- Started
- InProgress
- Canceled
- Completed
- Expired
- Rejected
- Template
- CompletedWithWarnings

The status of an recipient can be

- NotSigned
- Signed
- Rejected
- Delegated

Usage

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <GetEnvelopeById_v1Response xmlns="http://www.eSignAnyWhere.com/">
         <GetEnvelopeById_v1Result><![CDATA[<apiResult version="2.2.452.6541">
  <baseResult>ok</baseResult>
  <okInfo>
    <envelopeStatus>
      <id>62fe2dcd-811d-4ac2-9602-f146e18075c1</id>
      <name>API-Test draft to envelope</name>
      <status>Expired</status>
      <sendDate>2017-01-19T08:14:24.223Z</sendDate>
      <expirationDate>2017-01-21T08:14:24.223Z</expirationDate>
      <bulkRecipients>
        <bulkRecipient eMail="">
          <status>Expired</status>
          <recipients>
            <recipient>
              <orderIndex>1</orderIndex>
              <eMail>recipient1@email.com</eMail>
              <status>NotSigned</status>
              <signedDate></signedDate>
              <recipientType>Signer</recipientType>
              <workstepRedirectionUrl></workstepRedirectionUrl>
            </recipient>
            <recipient>
              <orderIndex>2</orderIndex>
              <eMail>recipient2@email.com</eMail>
              <status>NotSigned</status>
              <signedDate></signedDate>
              <recipientType>Signer</recipientType>
              <workstepRedirectionUrl></workstepRedirectionUrl>
            </recipient>
            <recipient>
              <orderIndex>3</orderIndex>
              <eMail>copyReceiver@somewhere.com</eMail>
              <status>NotSigned</status>
              <signedDate></signedDate>
              <recipientType>Cc</recipientType>
              <workstepRedirectionUrl></workstepRedirectionUrl>
            </recipient>
          </recipients>
          <completedDocuments>
            <logDocumentId></logDocumentId>
          </completedDocuments>
        </bulkRecipient>
      </bulkRecipients>
    </envelopeStatus>
  </okInfo>
</apiResult>]]></GetEnvelopeById_v1Result>
      </GetEnvelopeById_v1Response>
   </soap:Body>
</soap:Envelope>
```

This request will return you the current status of your organization’s license. Please note that for on premise the data model is a bit different.

```
<licenseInformation>
 <type>Per Number of Documents</type>
 <expirationDateUtc>2019-05-30T00:00:00</expirationDateUtc>
 <documents>
        <total>99999</total>
        <used>13</used>
 </documents>
 <users>
        <total>999999</total>
        <used>12</used>
 </users>
</licenseInformation>
```

Fastest call to the SignAnyWhere server. Can be used to check if server is reachable, or to check the server version.

Result

Result is an apiResult with baseResult type OK, but **without** an okInfo element. The server version is an attribute of the response's document root, as in every SOAP response of the eSignAnyWhere API.

```
<apiResult version="2.2.458.6616">
  <baseResult>ok</baseResult>
</apiResult>
```

This request doesn't require an authentication.
This function should be used in combination with the advanced tags in documents. See the Plaeholder Use Case for more information about the tags.

Input: list of files, ad-hoc workstep, additional descriptor

```
<prepareSendEnvelopeStepsDescriptor>
  <clearFieldMarkupString>0|1</clearFieldMarkupString>
</prepareSendEnvelopeStepsDescriptor>
```

Result: an ad-hoc workstep configuration and list of steps for SendEnvelope_v1 (prefilled).

```
<prepareSendEnvelopeStepsResult>
 <workstepConfiguration skipThirdPartyChecks="0">
   <!-- ad hoc config as in "GetAdHocWorkstepConfiguration_v1", contains existing form elements and sig strings
in the PDF which can additionally be assigned by the integration code -->
 </workstepConfiguration>
 <steps> <!-- one step for each signerX in the markup -->
    <step>
       <emailBodyExtra>
       </emailBodyExtra>
       <orderIndex>1</orderIndex>
       <recipientType>Signer</recipientType>
       <workstepConfiguration skipThirdPartyChecks="0">
         <!-- filled out configuration for the recipient, contains all tasks for "signer1" in the field markup
-->
       </workstepConfiguration>
       <recipients>
          <recipient>
             <eMail>placeholder1@placeholder1.com</eMail>
             <emailBodyExtra>
             </emailBodyExtra>
             <firstName>placeholder1</firstName>
             <lastName>placeholder1</lastName>
             <languageCode>en</languageCode>
             <disableEmail>false</disableEmail>
             <addAndroidAppLink>false</addAndroidAppLink>
             <addIosAppLink>false</addIosAppLink>
             <addWindowsAppLink>false</addWindowsAppLink>
             <allowDelegation>true</allowDelegation>
             <authentications />
          </recipient>
       </recipients>
    </step>
 </steps>
 <addFormFields>
  <!-- all add form field descriptions which are defined in the field markup -->
 </addFormFields>
</prepareSendEnvelopeStepsResult>
```

Usage:

- upload PDFs *UploadTemporarySspFile_v1*
- call *PrepareSendEnvelopeSteps_v1*
    - in the *prepareSendEnvelopeStepsResult*: fill in recipient data (firstname, lastname, email)
    - optional: in the *prepareSendEnvelopeStepsResult*: assign unassigned elements in the “ad hoc” config, set advanced recipient properties, …
- create envelope XML with data from *PrepareSendEnvelope_v1*

```
<envelope>
  ...
  <steps> <!-- use data from prepareSendEnvelopeStepsResult -->
  </steps>
  ...
  <addFormFields> <!-- use data from prepareSendEnvelopeStepsResult -->
  </addFormFields>
</envelope>
```

- call *SendEnvelope_v1* to send Envelope

You can replace a recipient via API as long as it is not the recipient’s turn. You also can replace the workstep configuration of a recipient, if the envelope was created via API and the recipient is a signer.

The Override-Recipient-XML:

```
<recipientOverride>
   <recipientLookUp>
      <eMail>recipient@xyzmo.com</eMail>
      <orderIndex>2</orderIndex>
   </recipientLookUp>
   <recipient>
      <eMail>replacedByApi@eflowauto.test</eMail>
      <emailBodyExtra>some extra text</emailBodyExtra>
      <firstName>api</firstName>
      <lastName>replacetest</lastName>
      <languageCode>de</languageCode>
      <disableEmail>false</disableEmail>
      <addAndroidAppLink>false</addAndroidAppLink>
      <addIosAppLink>false</addIosAppLink>
      <addWindowsAppLink>false</addWindowsAppLink>
      <allowDelegation>false</allowDelegation>
      <authentications>
         <authentication>
            <method>Pin</method>
            <parameter>42</parameter>
         </authentication>
      </authentications>
      <disposableCertificateAdditionalInformation>
         <countryResidence>AT</countryResidence>
         <phoneMobile>+43(676)111-1111</phoneMobile>
         <documentType>CI</documentType>
         <documentIssuedBy>Some Authority</documentIssuedBy>
         <documentIssuedOn>2010-09-01T00:00:00Z</documentIssuedOn>
         <documentExpiryDate>2020-09-01T00:00:00Z</documentExpiryDate>
         <socialSecurityNumber>TEST123TEST456</socialSecurityNumber>
         <documentNumber>docnr</documentNumber>
      </disposableCertificateAdditionalInformation>
   </recipient>
   <workstepConfiguration>
   ... <!-- optional; only valid when API generated + recipient is a signer -->
   </workstepConfiguration>
</recipientOverride>
```

You can restart only expired envelopes with this function. Just use the envelope id to restart it.
Send a specific template. Therefore the templateId and the overrideOptionsXML is required.

OverrideOptionsXML:

```
<overrideOptions>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <name>new name</name>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <eMailSubject>new subject</eMailSubject>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <eMailBody>new body</eMailBody>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <!-- example: http://172.16.17.138:57550/default.aspx?EnvelopeId=##EnvelopeId## -->
   <callbackUrl></callbackUrl>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <!-- example: http://172.16.17.78:57550/default.aspx?
envelopeId=##EnvelopeId##&action=##Action##&myParamForMe=1234 -->
   <statusUpdateCallbackUrl></statusUpdateCallbackUrl>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <enableReminders>true</enableReminders>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <firstReminderDayAmount>1</firstReminderDayAmount>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <recurrentReminderDayAmount>2</recurrentReminderDayAmount>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <beforeExpirationReminderDayAmount>1</beforeExpirationReminderDayAmount>
   <!-- if node is present, value will be overridden, if missing, value from template is used -->
   <daysUntilExpire>20</daysUntilExpire>

   <recipientOverrides>
      <recipientOverride>
                <recipientLookUp>
                       <!-- in case a placeholder for a recipient is used: <eMail>Placeholder:
NameOfPlaceholder</eMail> -->
            <eMail>template1recip@email.com</eMail>
            <orderIndex>1</orderIndex>
                 </recipientLookUp>
         <recipient>
            <eMail>new.email@gmail.com</eMail>
            <emailBodyExtra>some extra text</emailBodyExtra>
            <firstName>Hello</firstName>
            <lastName>World</lastName>
            <languageCode>en-US</languageCode>
            <authentications>
               <authentication>
                  <method>Pin</method>
                  <parameter>42</parameter>
               </authentication>
            </authentications>
         </recipient>
      </recipientOverride>

          <recipientOverride>
              <recipientLookUp>
            <eMail>template2.recip@email.com</eMail>
            <orderIndex>2</orderIndex>
                 </recipientLookUp>
         <recipient>
            <eMail>new.recip2@gmail.com</eMail>
            <emailBodyExtra>some extra text</emailBodyExtra>
            <firstName>Hello</firstName>
            <lastName>World</lastName>
            <languageCode>en-US</languageCode>
            <authentications>
               <authentication>
                  <method>Pin</method>
                  <parameter>4812</parameter>
               </authentication>
            </authentications>
         </recipient>
      </recipientOverride>

   </recipientOverrides>
</overrideOptions>
```

Creates the Envelope on the server and sends notification mails to the recipients (in default configuration). See Implementation for information how to integrate the envelope worksteps in your environment. For more details of parameter **envelopeDescriptionXml** see: The Envelope XML

Usage

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:esig="http://www.
eSignAnyWhere.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <esig:SendEnvelope_v1>
         <!--Optional:-->
         <esig:authorizationXml>
        <authorization>
          <organizationKey>##ORGANIZATION-KEY##</organizationKey>
          <userLoginName>##USER-LOGON##</userLoginName>
         </authorization>
         </esig:authorizationXml>
         <!--Optional:-->
         <esig:sspFileIds>
            <!--Zero or more repetitions:-->
            <esig:string>##FILE-ID##</esig:string>
         </esig:sspFileIds>
         <!--Optional:-->
         <esig:envelopeDescriptionXml>
<envelope>
   <name>API-Test draft to envelope</name>
   <eMailSubject>hello world</eMailSubject>
   <eMailBody>Dear #RecipientFirstName#!

Please sign this document.</eMailBody>
   <enableReminders>True</enableReminders>
   <firstReminderDayAmount>1</firstReminderDayAmount>
   <recurrentReminderDayAmount>1</recurrentReminderDayAmount>
   <beforeExpirationReminderDayAmount>1</beforeExpirationReminderDayAmount>
   <daysUntilExpire>2</daysUntilExpire>

   <!-- callback to your backend system on a completed envelope
   <callbackUrl>http://172.16.17.78:57550/default.aspx?EnvelopeId=##EnvelopeId##&myParamForMe=1234<
/callbackUrl>
   -->
   <callbackUrl></callbackUrl>

   <steps>
      <!-- define recipient and tasks of every step in this envelope -->
      <!-- 1st step -->
      <step>
         <emailBodyExtra></emailBodyExtra>
         <orderIndex>1</orderIndex>
         <recipientType>Signer</recipientType>
         <recipients>
            <recipient>
               <languageCode>en-us</languageCode>
               <eMail>recipient1@email.com</eMail>
               <firstName>First</firstName>
               <lastName>Signer</lastName>
                <!-- optional authentication methods for this recipient -->
            </recipient>
         </recipients>
         <!-- Workstep config for this step -->
         <!--##WorkstepConfigRecipient1##-->
<workstepConfiguration>
      <WorkstepLabel>workstepLabel</WorkstepLabel>
      <SmallTextZoomFactorPercent>100</SmallTextZoomFactorPercent>
      <WorkstepTimeToLiveInMinutes>0</WorkstepTimeToLiveInMinutes>
      <FinishAction>
        <ServerAction callSynchronous="0" />
        <ClientAction clientName="SIGNificant SignAnywhere" closeApp="0" RemoveDocumentFromRecentDocumentList="
0" CallClientActionOnlyAfterSuccessfulSync="1">https://www.significant.com</ClientAction>
      </FinishAction>
      <signatureTemplate>
```

```
        <version>1.2.0.2</version>
        <positionUnits>PdfUnits</positionUnits>
        <positionReferenceCorner>Lower_Left</positionReferenceCorner>
      </signatureTemplate>
      <pdfForms isEditingAllowed="1" />
      <ViewerPreferences>
        <ShowVersionNumber>1</ShowVersionNumber>
        <EnableThumbnailDisplay>1</EnableThumbnailDisplay>
        <EnableWarningPopupOnLeave>1</EnableWarningPopupOnLeave>
        <WarningPopupDisplayAfter>Always</WarningPopupDisplayAfter>
        <GuidingBehavior>GuideRequiredAndOptionalTasks</GuidingBehavior>
        <FormFieldsGuidingBehavior>AllowSubmitAlways</FormFieldsGuidingBehavior>
        <FinishWorkstepOnOpen>0</FinishWorkstepOnOpen>
      </ViewerPreferences>
      <Policy version="1.0.0.2">
        <GeneralPolicies>
          <AllowSaveDocument>1</AllowSaveDocument>
          <AllowSaveAuditTrail>1</AllowSaveAuditTrail>
          <AllowRotatingPages>1</AllowRotatingPages>
          <AllowAppendFileToWorkstep>0</AllowAppendFileToWorkstep>
          <AllowAppendTasksToWorkstep>0</AllowAppendTasksToWorkstep>
          <AllowEmailDocument>0</AllowEmailDocument>
          <AllowPrintDocument>0</AllowPrintDocument>
          <AllowFinishWorkstep>1</AllowFinishWorkstep>
          <AllowRejectWorkstep>1</AllowRejectWorkstep>
          <AllowUndoLastAction>0</AllowUndoLastAction>
          <AllowColorizePdfForms>0</AllowColorizePdfForms>
          <AllowAdhocPdfAttachments>1</AllowAdhocPdfAttachments>
          <AllowAdhocSignatures>0</AllowAdhocSignatures>
          <AllowAdhocStampings>0</AllowAdhocStampings>
          <AllowAdhocFreeHandAnnotations>0</AllowAdhocFreeHandAnnotations>
          <AllowAdhocTypewriterAnnotations>0</AllowAdhocTypewriterAnnotations>
          <AllowAdhocPictureAnnotations>0</AllowAdhocPictureAnnotations>
          <AllowAdhocPdfPageAppending>0</AllowAdhocPdfPageAppending>
        </GeneralPolicies>
        <WorkstepTasks SequenceMode="SequenceOnlyRequiredTasks" originalSequenceMode="
SequenceOnlyRequiredTasks" />
        <AdhocPolicies>
          <AllowModificationsAfterSignature>1</AllowModificationsAfterSignature>
        </AdhocPolicies>
      </Policy>
      <timeCreated>2017-01-17T10:25:02.7013041Z</timeCreated>
      <signaturePluginConfiguration>
        <PdfSignatureProperties_V1>
          <PdfAConformant>0</PdfAConformant>
          <PAdESPart4Compliant>1</PAdESPart4Compliant>
          <IncludeSigningCertificateChain>1</IncludeSigningCertificateChain>
          <SigningCertificateRevocationInformationIncludeMode>Include<
/SigningCertificateRevocationInformationIncludeMode>
        </PdfSignatureProperties_V1>
        <PdfSignatureCryptographicData_V1>
          <SignatureHashAlgorithm>Sha256</SignatureHashAlgorithm>
          <SigningCertificateDescriptor>
            <Identifier>##SIGNATURE-HASH##</Identifier>
            <Type>Sha1Thumbprint</Type>
            <Csp>Default</Csp>
          </SigningCertificateDescriptor>
        </PdfSignatureCryptographicData_V1>
      </signaturePluginConfiguration>
      <TransactionCodeConfigurations>
        <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="de">
          <Message>Bitte authentifizieren Sie sich für den Zugriff auf die Dokumentenmappe der Transaktion
{tId}. Ihre TAN lautet: {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>
      </TransactionCodeConfigurations>
    </workstepConfiguration>

        </step>
        <!-- 2nd step -->
        <step>
         <emailBodyExtra></emailBodyExtra>
         <orderIndex>2</orderIndex> <!-- for parallel steps set the same order index -->
         <recipientType>Signer</recipientType>
         <recipients>
            <recipient>
               <languageCode>en-us</languageCode>
               <eMail>recipient2@email.com</eMail>
               <firstName>Second</firstName>
               <lastName>Signer</lastName>
                <!-- optional authentication methods for this recipient -->
            </recipient>
         </recipients>
         <!-- Workstep config for this step -->
         <!--##WorkstepConfigRecipient2##-->
<workstepConfiguration>
      <WorkstepLabel>workstepLabel</WorkstepLabel>
      <SmallTextZoomFactorPercent>100</SmallTextZoomFactorPercent>
      <WorkstepTimeToLiveInMinutes>0</WorkstepTimeToLiveInMinutes>
      <FinishAction>
        <ServerAction callSynchronous="0" />
        <ClientAction clientName="SIGNificant SignAnywhere" closeApp="0" RemoveDocumentFromRecentDocumentList="
0" CallClientActionOnlyAfterSuccessfulSync="1">https://www.significant.com</ClientAction>
      </FinishAction>
      <signatureTemplate>
        <version>1.2.0.2</version>
        <positionUnits>PdfUnits</positionUnits>
        <positionReferenceCorner>Lower_Left</positionReferenceCorner>
      </signatureTemplate>
      <pdfForms isEditingAllowed="1" />
      <ViewerPreferences>
        <ShowVersionNumber>1</ShowVersionNumber>
        <EnableThumbnailDisplay>1</EnableThumbnailDisplay>
        <EnableWarningPopupOnLeave>1</EnableWarningPopupOnLeave>
        <WarningPopupDisplayAfter>Always</WarningPopupDisplayAfter>
        <GuidingBehavior>GuideRequiredAndOptionalTasks</GuidingBehavior>
        <FormFieldsGuidingBehavior>AllowSubmitAlways</FormFieldsGuidingBehavior>
        <FinishWorkstepOnOpen>0</FinishWorkstepOnOpen>
      </ViewerPreferences>
      <Policy version="1.0.0.2">
        <GeneralPolicies>
          <AllowSaveDocument>1</AllowSaveDocument>
          <AllowSaveAuditTrail>1</AllowSaveAuditTrail>
          <AllowRotatingPages>1</AllowRotatingPages>
          <AllowAppendFileToWorkstep>0</AllowAppendFileToWorkstep>
          <AllowAppendTasksToWorkstep>0</AllowAppendTasksToWorkstep>
          <AllowEmailDocument>0</AllowEmailDocument>
          <AllowPrintDocument>0</AllowPrintDocument>
          <AllowFinishWorkstep>1</AllowFinishWorkstep>
          <AllowRejectWorkstep>1</AllowRejectWorkstep>
          <AllowUndoLastAction>0</AllowUndoLastAction>
          <AllowColorizePdfForms>0</AllowColorizePdfForms>
          <AllowAdhocPdfAttachments>1</AllowAdhocPdfAttachments>
          <AllowAdhocSignatures>0</AllowAdhocSignatures>
          <AllowAdhocStampings>0</AllowAdhocStampings>
```

```
          <AllowAdhocFreeHandAnnotations>0</AllowAdhocFreeHandAnnotations>
          <AllowAdhocTypewriterAnnotations>0</AllowAdhocTypewriterAnnotations>
          <AllowAdhocPictureAnnotations>0</AllowAdhocPictureAnnotations>
          <AllowAdhocPdfPageAppending>0</AllowAdhocPdfPageAppending>
        </GeneralPolicies>
        <WorkstepTasks SequenceMode="SequenceOnlyRequiredTasks" originalSequenceMode="
SequenceOnlyRequiredTasks" />
        <AdhocPolicies>
          <AllowModificationsAfterSignature>1</AllowModificationsAfterSignature>
        </AdhocPolicies>
      </Policy>
      <timeCreated>2017-01-17T10:25:02.7013041Z</timeCreated>
      <signaturePluginConfiguration>
        <PdfSignatureProperties_V1>
          <PdfAConformant>0</PdfAConformant>
          <PAdESPart4Compliant>1</PAdESPart4Compliant>
          <IncludeSigningCertificateChain>1</IncludeSigningCertificateChain>
          <SigningCertificateRevocationInformationIncludeMode>Include<
/SigningCertificateRevocationInformationIncludeMode>
        </PdfSignatureProperties_V1>
        <PdfSignatureCryptographicData_V1>
          <SignatureHashAlgorithm>Sha256</SignatureHashAlgorithm>
          <SigningCertificateDescriptor>
            <Identifier>##SIGNATURE-HASH##</Identifier>
            <Type>Sha1Thumbprint</Type>
            <Csp>Default</Csp>
          </SigningCertificateDescriptor>
        </PdfSignatureCryptographicData_V1>
      </signaturePluginConfiguration>
      <TransactionCodeConfigurations>
        <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="de">
          <Message>Bitte authentifizieren Sie sich für den Zugriff auf die Dokumentenmappe der Transaktion
{tId}. Ihre TAN lautet: {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>
      </TransactionCodeConfigurations>
    </workstepConfiguration>

        </step>
        <!-- define a copy receiver -->
        <step>
         <emailBodyExtra></emailBodyExtra>
         <orderIndex>3</orderIndex>
         <recipientType>CC</recipientType>
         <recipients>
            <recipient>
               <languageCode>en-us</languageCode>
               <eMail>copyReceiver@somewhere.com</eMail>
               <firstName>Copy Receiver Name</firstName>
               <lastName>Thunderbolt</lastName>
            </recipient>
         </recipients>
        </step>
    </steps>
```

```
</envelope>
        </esig:envelopeDescriptionXml>
      </esig:SendEnvelope_v1>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:
Body><SendEnvelope_v1Response xmlns="http://www.eSignAnyWhere.com/"><SendEnvelope_v1Result><apiResult version="
2.1.260.5831">
  <baseResult>ok</baseResult>
  <okInfo>
    <envelopeId>##ENVELOPE-ID##</envelopeId>
  </okInfo>
</apiResult></SendEnvelope_v1Result></SendEnvelope_v1Response></soap:Body></soap:Envelope>
```

In case you don't want to wait on the automatic reminders for a recipient, you can manually send the reminder. Just use the envelope-id with this function and it will automatically remind the current recipient about his or her signing task. Please note, that you can only resend reminders every 12 hours (default value), to prevent a spaming of the recipient. This value can be changed on private SaaS or on premise instances.
This call only requires an envelope id to unlock a recipient in a parallel use case. So if a signer has opened his document in the parallel case and didn't finish it, it can be reverted and the others can open it. It is the same functionality as in the UI to unlock the envelope.
Uploads a file for using with an new envelope. Returns a FileId. The FileId is at least valid for 10 min (time-to-live, actual live time depends on clean up service load). Full HTTP Request: https://www.significant.com/api.asmx?op=UploadTemporarySspFile_v1

Parameters

file:

```
<file><data>##fileContent##</data><name>##filename##</name></file>
```

The file content itself is Base64 encoded.

Result

Returns a file id which is required for further calls referring to this file. Store the generated file id in your application.
This call uploads a document to the users clipboard. This is also used for the MS Word plugin.
This call only verifies your authentication (email and API-key).

Reference Usermanagement API

Usermanagement API can be used by user managers. It allows you to manage users and teams. It should be also used to populate SAML attributes for the users.

userDescription:

```
<userDescription>
   <eMail>test.email@flowtest.com</eMail>
   <firstName>Test</firstName>
   <lastName>Email</lastName>
   <jobTitle></jobTitle> <!-- optional -->
   <phoneNumber></phoneNumber> <!-- optional -->
   <role>None|PowerUser|RegisteredSigner</role>
   <isUserManager>0|1|true|false</isUserManager>
   <defaultSubject></defaultSubject> <!-- optional -->
   <defaultMessage></defaultMessage> <!-- optional -->
   <notifyWhenActionNeeded>0|1|true|false</notifyWhenActionNeeded>
   <notifyWhenRecipientSigned>0|1|true|false</notifyWhenRecipientSigned>
   <notifyWhenRecipientRejected>0|1|true|false</notifyWhenRecipientRejected>
   <notifyWhenDeliveryFailed>0|1|true|false</notifyWhenDeliveryFailed>
   <notifyWhenRecipientOpenedDocument>0|1|true|false</notifyWhenRecipientOpenedDocument>
   <authentications> <!-- for SAML Authenticators -->
      <samlAuthentication> <!-- array -->
         <providerName>AD FS My</providerName> <!-- SAML (AD FS, authentity provider with SAML) -->
         <userIdentifierAttibuteValue>adam</userIdentifierAttibuteValue> <!-- SAML  (AD FS) -->
      </samlAuthentication>
   </authentications>
</userDescription>
```

userActivationDescriptor:

```
<userActivationDescriptor>
  <sendInvitationMail>0|1</sendInvitationMail>
</userActivationDescriptor>
```

All fields are optional. If they are not set, the original value is being kept. Attention: an empty tag will update the value to empty!

```
<userDescription>
   <eMail>test.email@flowtest.com</eMail>
   <firstName>Test</firstName>
   <lastName>Email</lastName>
   <jobTitle></jobTitle>
   <phoneNumber></phoneNumber>
   <role>None|PowerUser|RegisteredSigner</role>
   <isUserManager>0|1|true|false</isUserManager>
   <defaultSubject></defaultSubject>
   <defaultMessage></defaultMessage>
   <notifyWhenActionNeeded>0|1|true|false</notifyWhenActionNeeded>
   <notifyWhenRecipientSigned>0|1|true|false</notifyWhenRecipientSigned>
   <notifyWhenRecipientRejected>0|1|true|false</notifyWhenRecipientRejected>
   <notifyWhenDeliveryFailed>0|1|true|false</notifyWhenDeliveryFailed>
   <notifyWhenRecipientOpenedDocument>0|1|true|false</notifyWhenRecipientOpenedDocument>
   <authentications> <!-- for SAML Authenticators -->
      <samlAuthentication> <!-- array -->
         <providerName>AD FS My</providerName> <!-- SAML (AD FS, authentity provider with SAML) -->
         <userIdentifierAttibuteValue>adam</userIdentifierAttibuteValue> <!-- SAML  (AD FS) -->
      </samlAuthentication>
   </authentications>
   <AutomatedDelegationSettings>
    <Reason>The reason for delegation</Reason>
    <EndDate>2019-10-20T12:54:53.217Z</EndDate>
    <DelegateeUserId>##USERID##</DelegateeUserId>
    <UtilizeAlsoOnCopyRecipients>1</UtilizeAlsoOnCopyRecipients>
  </AutomatedDelegationSettings>
</userDescription>
```

reassignEnvelopesDescriptor:

```
<reassignEnvelopesDescriptor>
  <newRecipient>some@guy.com</newRecipient>
  <reassignDrafts>0|1</reassignDrafts>
  <reassignTemplates>0|1</reassignTemplates>
  <reassignClipboard>0|1</reassignClipboard>
  <reassignAddressBook>0|1</reassignAddressBook>
</reassignEnvelopesDescriptor>
```

Returns the user information by email.
findUserDescriptor:

```
<findUsersDescriptor>
  <role>None|PowerUser|RegisteredSigner</role> <!-- optional -->
  <isUserManager>0|1|true|false</isUserManager> <!-- optional -->
</findUsersDescriptor>
```

The call is identical to FindUsers_v1, but the result is a list of users with several details. This list can then be used to to calculate required user settings with regards to authorization.

```
<findUsersDescriptor>
  <role>None|PowerUser|RegisteredSigner</role> <!-- optional -->
  <isUserManager>0|1|true|false</isUserManager> <!-- optional -->
</findUsersDescriptor>
```

Result:

```
 <extendedFindUsersResult>
        <user>
               <id>c6926bdf-a2ee-492a-8ff1-d451f28de7bb</id>
               <eMail>Ronald.Wenny@eflowauto.test</eMail>
               <firstName>Marius</firstName>
               <lastName>Petereder</lastName>
               <sid>MySid</sid>
               <userName>
               </userName>
               <role>PowerUser</role>
               <isUserManager>true</isUserManager>
               <isEnabled>true</isEnabled>
               <authentications>
                  <samlAuthentication>
                         <providerName>QA AD FS</providerName>
                         <userIdentifierAttibuteValue>Ronald.Wenny@eflowauto.test</userIdentifierAttibuteValue>
                  </samlAuthentication>
               </authentications>
        </user>
        <user>
               <id>734fdfcb-1fd2-4e9d-84c6-d9f720d70b7b</id>
               <eMail>mak.joan@xyzmo.com</eMail>
               <firstName>Mak</firstName>
               <lastName>Joan</lastName>
               <sid>MySIDasdflaskdjföalskdfjöasldfkjöalsdkfjaskldöfju</sid>
               <userName>administrator</userName>
               <role>PowerUser</role>
               <isUserManager>true</isUserManager>
               <isEnabled>true</isEnabled>
        </user>
 </extendedFindUsersResult>
```

GetTeam returns as response the same structure as required for SetTeams.

GetTeam returns as response the same structure as required for SetTeams.

teamXML:

```
<teams>
        <team>
           <name>test team</name>
           <allowEnvelopeSharingWithinTeam>false</allowEnvelopeSharingWithinTeam>
           <allowTemplateSharingWithinTeam>false</allowTemplateSharingWithinTeam>
           <teamMember>
              <eMail>knorze@xyzmo.com</eMail>
              <teamMembers>
                 <teamMember>
                    <eMail>ronald.korze@significant.com</eMail>
                    <teamMembers />
                 </teamMember>
              </teamMembers>
           </teamMember>
        </team>
     </teams>
```