API Documentation

Attention

Please note that this documentation and the links refer to the api v6. For more information please see the migration guide and the documentation related to v5.

REST API /v3 and /v4 DEPRECATION: The 23.76 (published March 2024) will be the last LTS version that includes these API versions. By early June 2024, the REST API routes to v3/v4 will be deactivated on DEMO. Early December 2024, the REST API routes to v3/v4 will be removed from feature stream releases. Note that there is no date communicated yet to discontinue REST APIv5 (and where v5 refers to v4 routes, these will still remain); however we recommend to use the /v6 API specification already.

Introduction

On this page you will find the *eSAW API* description. First we start with a basic overview of the API. If you are looking for examples we recommend the Postman Sample, Envelope structure and our Stories and Examples.

- Introduction
- Principles of Api v6
- Overview and references
- Resources
- General conncepts
  - Object validation
  - Authorization
  - Authorization
    - Format Specification
- Callbacks
  - Callback Types
    - Envelope Callback
    - Envelope Status Callback
    - Workstep Event Callbacks
      - Available Event Types
    - Draft Callbacks

Principles of Api v6

Detailed information about all changes between api v5 and api v6 can be found here: migration guide.

- Reduced HTTP verbs
  - Only HTTP GET and POST
- Consistent naming and symmetric structures within api v6
- Consistency between Web User Interface (WebUI) and api v6 in features but also naming
  - Positive wording (e.g. prevent sharing with team changed to share with team)
- Simplified terminology and structures
- No abstract types any more

Changes has been made on api method level as well as the JSON structures and the envelope status values changed.

Overview and references

The API is for developers, who want to integrate eSignAnyWhere into their application and for administrators, who want to script interactions with eSignAnyWhere (e.g. user synchronization).

Quick Overview: eSAW uses REST (with JSON) as API. The basic workflow is to upload a document and then send the envelope with a envelope configuration. Optional, before sending the envelope, it is possible to prepare the envelope to get the workstep configuration for sending the envelope. For more information about the envelope configuration please also have a look at the Envelope Structure. The configuration consists out of the envelope part (workflow configuration) and for each action a definition and a signing configuration (workstep configuration). The workstep configuration is the description (as JSON for REST) of tasks for signer (e.g. Signature Fields, Form-Fields) and additional document configurations.

The easiest way to start is enabling the DeveloperMode for a user. As developer (and power user) you can send envelopes via eSignAnywhere in the UI and download the complete envelope configuration (including the workstep configurations). So eSAW can be a seen as configuration designer, where you can easily prepare the envelope configuration. After you downloaded the configuration you just have to replace the recipient information and configuration.

Resources

REST API Reference (Swagger) | >= 3.1 |	https://demo.esignanywhere.net/Api
REST tutorials	This turorials help getting familiar with the API technology and the most common tools to do first tests of API calls already before implementing your own integration code. visit REST tutorial using Postman
Tutorial: Hello World*	This tutorial allows to dig into the API integration of eSignAnyWhere a bit deeper. It focuses on audience already familiar with tools to run API calls, such as Postman or SoapUI. visit Hello World Tutorial
Developer mode*	visit developer mode
Sample Code in Java	Here you can find the java sample: Download. (Contains example with REST, developed with JavaSE-12)
SignAnyWhere Viewer 2019 Redesign v 3.5	visit SignAnyWhere Viewer 2019 Information
SignAnyWhere Viewer Extended Customization	visit SignAnyWhere Viewer Extended Customization
Integration & Use Cases	visit Integration & Use Cases
Developer FAQ	visit Developer FAQ
eSAW Error Codes	visit eSAW Error Codes

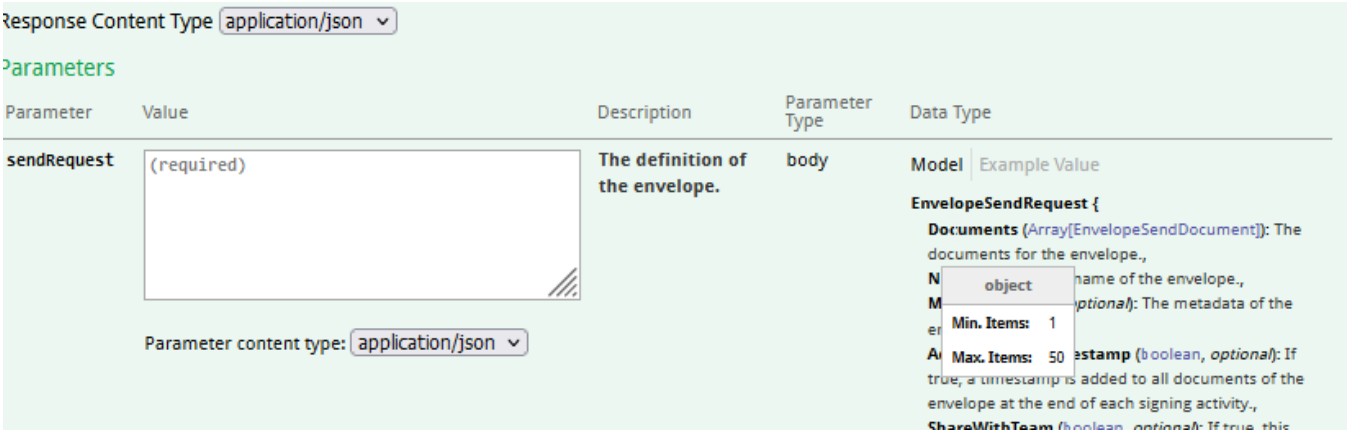
*** Recommended**

The User Guide, Signer Guide and Administration Guide (for on-premise customers) can be also helpful.

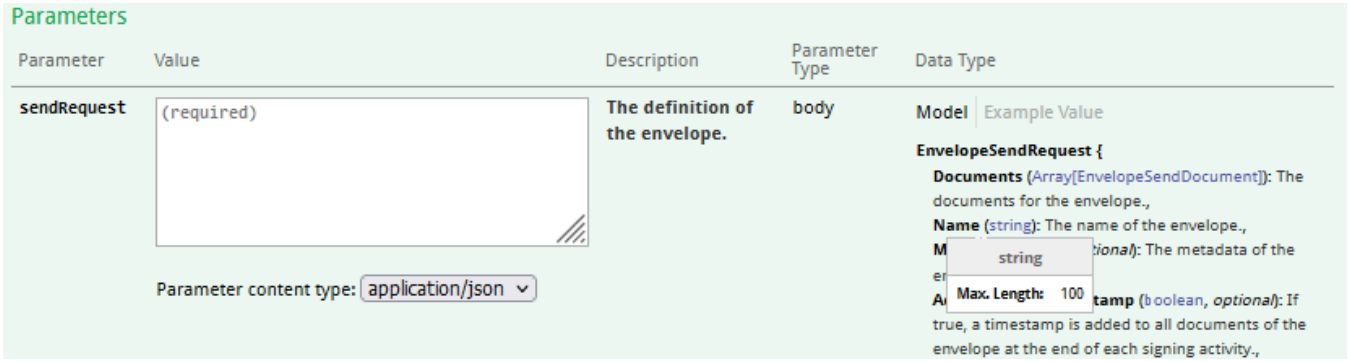
General conncepts

Object validation

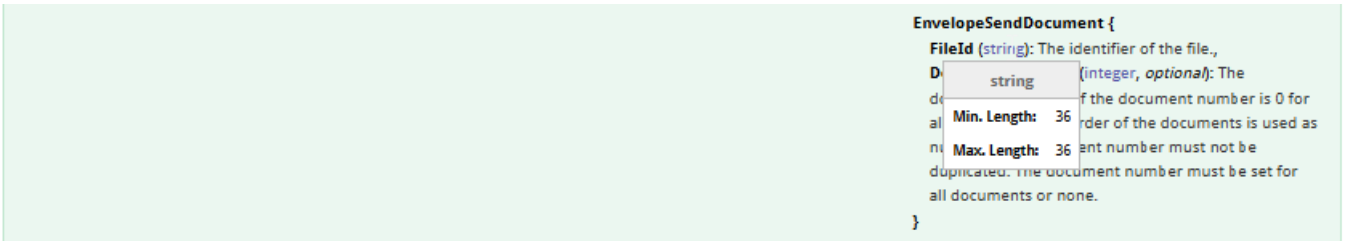
String and array objects are validated. If hovering over an object in the ***model*** section on e.g. https://demo.esignanywhere.net/Api/swagger/ui/index you can see which validation is performed. In the following sample the validation for the ***Documents*** array is such that at least one document must be added and maximum of 50 documents are allowed.



For string objects, the length is validated. In the following sample, a maximum of 100 characters is checked for the string ***Name.***



Ids have a fixed length, therefore the minimum and maximum values are the same:



Authorization

This section covers the authorization options for REST-API integrations. For the authorization you have different options with REST API; as described in the next chapters. If you are authorized you will get a HTTP/200 Ok info. Otherwise you will get a 401 Unauthorized error.

Authorization

We recommend to use user-specific API tokens. Therefore, each user can create several tokens for different application integrations. The apiToken has to be provided as HTTP Header. v 20.42

Please see the next sample authorization (Bearer token):

Key	Value
"Authorization"	e.g. "Bearer asdfngtmvv8pfmsuaxpzz85zux3e63dd9zttrwitx9mln6qka6tds83du3p3lroe"

Please see the next sample authorization (api token):

Key	Value
"ApiToken"	"asdfngtmvv8pfmsuaxpzz85zux3e63dd9zttrwitx9mln6qka6tds83du3p3lroe"

Such an user api token can be created in SettingsAPI Tokens and Apps; Section "API Tokens".

Tokens created by eSAW are currently 64-digit alphanumeric strings - but the length and set of allowed characters may be changed with future product versions.

Format Specification

Note that the key can be any 64 digit alphanumeric value; not necessarily following the GUID format! The length and set of allowed characters may be changed with future product versions.

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.

Time of Retry	Total time after t0	
	0 min	T0: Initial Callback Event will retry (see next row) if no HTTP 2xx response, or in case of timeout
T1 = T0 + 5 min	5 min	2nd Attempt (= 1st Retry) will retry (see next row) if no HTTP 2xx response, or in case of timeout
T2 = T1 + 10 min	15 min	3rd Attempt
T3 = T2 + 15 min	30 min	...
T4 = T3 + 20 min	50 min	...
...	...	...
T9 = T8 + 40 min	180 min = 3h	10th Attempt if still no HTTP 2xx response => listed as "warning" in errors view (assuming default value "10" configured in _global.xml for "notificationErrorThreshold")
...	...	...
T29 = T28 + 145 min	2175 min = 36.25h	30th Attempt if still no HTTP 2xx response => listed as "error" in errors view & permanent give-up (assuming default value "30" configured in _global.xml for "notificationMaximumRetries"); but can be triggered from UI / errors view) again

Callback Types

- CallbackConfiguration
- Draft Callbacks

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.

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 parameter for that envelope (e.g. internal references). Moreover, on our shared SaaS environments only HTTPS callbacks (port 443, and 1025-65535) are allowed.

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

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 parameter for that envelope (e.g. internal references). Moreover, on our shared SaaS environments only HTTPS callbacks (port 443, and 1025-65535) are allowed.

Sample:

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

Sample with custom parameter "*internalid*":

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

Workstep Event Callbacks

The workstep event callbacks are specific event callbacks fired on events caused by the underlying "SIGNificant Server Platform" component, but also routed through the notification system of eSignAnyWhere. Those callbacks inform in a way more detailled way about the workstep events - but note that those events are not necessarily time-synced to the envelope events. To trigger activities on the eSignAnyWhere API, always consider the envelope or envelope status callbacks.

Detailed callbacks on specific events

Note: You can configure a proxy for all callbacks. Please see the next sample:

```
<callbackProxySettings>
      <!-- Enable or disable the use of proxy for all callbacks. Values 1 (enabled) or 0 (disabled)-->
      <enabled>0</enabled>
      <!-- Address of the proxy server-->
      <address></address>
      <!-- Send all callbacks to local addresses without using proxy. Values 1 (bypass for local) or 0 (always
use proxy)-->
      <!-- Local requests are identified by the lack of a period (.) in the URI, as in http://webserver/, or
access the local server, including http://localhost, http://loopback, or http://127.0.0.1-->
      <bypassProxyOnLocal>0</bypassProxyOnLocal>
      <networkCredentials>
        <!-- Domain for Credentials-->
        <domain></domain>
        <!-- Username for Credentials-->
        <username></username>
        <!-- Password for Credentials. If password is not encrypted then remove the attribute enc-->
        <password enc="sec2"></password>
      </networkCredentials>
    </callbackProxySettings>
```

The following placeholders are defined:

- ##WorkstepId## – workstep of current action
- ##EventType## – type of event (see list of types below)
- ##Source## – internal (eSAW) or external (Viewer)
- ##Time## – time when the action occurred
- ##Description## – textual description of the event
- ##RecipientEmail## – emailadress of current recipient
- ##EnvelopeId## – current envelope id
- ##RecipientOrder## – index of current recipient

Please also see the available event types which can be added in the section "**CallbackConfiguration**". If one or more event types are enabled in the configuration the result is equivalent to the api v5 configuration for whitelist.

Please see the following sample configuration:

```
"CallbackConfiguration": {
  "CallbackUrl": "string",
  "StatusUpdateCallbackUrl": "string",
  "ActivityActionCallbackConfiguration": {
    "Url": "string",
    "ActionCallbackSelection": {
      "ConfirmTransactionCode": true,
      "DefaultEventType": true,
      "AgreementAccepted": true,
      "AgreementRejected": true,
      "RequestPrepareAuthenticationInformationSuccess": true,
      "PrepareAuthenticationSuccess": true,
      "AuthenticationFailed": true,
      "AuthenticationRejected": true,
      "AuthenticationSuccess": true,
      "ReAuthenticationFailed": true,
      "AuditTrailRequested": true,
      "AuditTrailXmlRequested": true,
      "CalledPage": true,
      "WhoIsInformation": true,
      "DocumentDownloaded": true,
      "FlattenedDocumentDownloaded": true,
      "AddedAnnotation": true,
      "AddedAttachment": true,
      "AppendedDocument": true,
      "FormsFilled": true,
      "ConfirmReading": true,
      "PageViewChanged": true,
      "SendTransactionCode": true,
      "PrepareSignWorkstepDocument": true,
      "SignWorkstepDocument": true,
      "UndoAction": true,
      "WorkstepCreated": true,
      "WorkstepFinished": true,
      "WorkstepRejected": true,
      "DisablePolicyAndValidityChecks": true,
      "EnablePolicyAndValidityChecks": true,
      "AppendFileToWorkstep": true,
      "AppendTasksToWorkstep": true,
      "SetOptionalDocumentState": true,
      "PreparePayloadForBatch": true
    }
  }
},
```

Please note that if all event types are disabled the URL which was configured in the configuration will not get any callbacks.

These events are fired by the Workstep Controller (internal component) and are fired before the data in eSAW is complete updated (some post-processing is required). Therefore this event callbacks are used only in rare integrations. For more information please see https://demo.esignanywhere.net/Api/swagger/ui/index#!/Envelope/Envelope_Send section *EnvelopeSendActionCallbackSelection*

Available Event Types

Type	Description
ConfirmTransactionCode	A transaction code was sent
AgreementAccepted	The user accepted the agreement
AgreementRejected	The user rejected the agreement
PrepareAuthenticationSuccess	The prepare authentication process succeeded
AuthenticationFailed	The user failed to authenticate
AuthenticationSuccess	The user succeeded to authenticate

AuditTrailRequested	The audittrail was requested
AuditTrailXmlRequested	The audittrail XML was requested
CalledPage	The viewer site was requested
DocumentDownloaded	The document download was requested
FlattenedDocumentDownloaded	The flattened document download was requested
AddedAnnotation	An annotation was added
AddedAttachment	An attachment was added
AppendedDocument	A document was appended
FormsFilled	A form field was filled
ConfirmReading	A reading task was completed
PageViewChanged	Note: This event is only used for the audit trail, no notification is sent to the configured URL. The user changed the page view (e.g. by scrolling through the document).
SendTransactionCode	This event is raised, when a TransactionCode for a signature with type TransactionCode has been sent using the IdentityServer or the TransactionCodeSenderPlugin
PrepareSignWorkstepDocument	A signature is prepared for signing
SignWorkstepDocument	Try to sign a signature
UndoAction	An action was undone
WorkstepCreated	A workstep was created
WorkstepFinished	A workstep was finished
WorkstepRejected	A workstep was rejected
DisablePolicyAndValidityChecks	The policy and validity checks have been disabled.
EnablePolicyAndValidityChecks	The policy and validity checks have been enabled.
AppendFileToWorkstep	A file was appended to the workstep
AppendTasksToWorkstep	A task was added to the workstep
SetOptionalDocumentState	A optional document became either active or inactive
PreparePayloadForBatch	The payload is getting prepared for batch signing

Draft Callbacks

Draft callbacks are fired, if a draft is used or deleted. The draft callback is set in the “CreateDraftOptions” (“AfterSendCallbackUrl”: “”), via the following call: https://demo.esignanywhere.net/Api/swagger/Draft/Draft_Create

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

Sample:

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