

v5 Api Reference - Introduction REST

On this page you will find the eSAW API description. First 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 [Postman Sample](#), [REST Guide](#) and our [Stories and Examples](#).

- [Authorization](#)
 - [UserKey Header Authorization](#)
 - [Bearer token Authorization](#)
 - [OrganizationKey and UserLogin Header](#)
 - [Creating an organization Key](#)
 - [Format Specification](#)
- [Callbacks](#)
 - [Callback Types](#)
 - [Envelope Callback](#)
 - [Envelope Status Callback](#)
 - [Workstep Event Callbacks](#)
 - [Blacklist-Definition](#)
 - [Whitelist-Definition](#)
 - [Available Event Types](#)
 - [Draft Callbacks](#)
- [Error](#)

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.

UserKey Header 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. ≥20.42

Header	Description	Example Value
apiToken	The user specific API token	e.g. "asdfngtmvv8pfmsuaxpzz85zux3e63dd9ztttrwix9mln6qka6tds83du3p3lroe"

Such an organization key can be created in SettingsAPI Tokens and Apps; Section "API Tokens".

Tokens created by eSAW are currently 66-digit alphanumeric strings - but the length and set of allowed characters may be changed with future product versions.

The User Key can also be retrieved, for integration scenarios, by implementing an OAuth Authorization Code Flow.

Bearer token Authorization

Currently available for just some API methods (e.g. sspfile/uploadtemporary).
The same key as used for the userKey header authorization can be used as bearer token.

OrganizationKey and UserLogin Header

Authentication can be done also using the userlogin name and an organization-wide organization key in the HTTP headers. We recommend to *avoid* using organization key authorization in scenarios where the organization key has to be shared with users, as it may be misused to gain access to other sender's envelopes.

Creating an organization Key



To create organization keys in new organizations where the organization was created with software version 21.16 or newer, it is required to enable the Feature Flag "OrganizationApiToken". On a SaaS environment, Namirial staff will enable that feature per organization on request. The request needs to come from a user who is granted the Administrator permission in that organization.

Such an organization key can be created in SettingsAPI Tokens and Apps; Section "Organization API Tokens". Note that the option to create an organization key will be invisible, if the instance was set up with a newer product version and the feature flag was not enabled.

Organization API Tokens

ADD NEW API TOKEN

You will be asked to name the API token. The name of the API token has no functional behavior, it just helps to manage different tokens. We recommend to create independent tokens for different API integrations. This will allow you to invalidate a token easily in case one external application did e.g. publish the token by mistake.

Add new API Token

Please enter a name for the new token

Token for Integration XYZ

CANCEL

ADD


Once completed, the token will be listed, together with other created tokens:

Organization API Tokens

☒ DETAILS 

☐ DETAILS 

Token for Integration XYZ

☒ DETAILS 

ADD NEW API TOKEN

The list view allows you to (temporarily) disable a token with the slider, open a details view, or delete a token permanently.

To read the token value, open the Details view:

Token for Integration XYZ

DETAILS

Name

Token for Integration XYZ

Organization Token

lrl9gur5v3de741e41gyf6pwi0os308s

SAVE

ADD NEW API TOKEN

With the copy-button next to the "Organization Token" field, the token can be copied to the Windows Clipboard and inserted in your business application or integration configuration.

Be careful, handle the token like an organization wide password! Keep in mind that the token allows wide access to your organization's configuration and documents.

Format Specification

Note that the key can be any 32digit alphanum value; not necessarily following the GUID format! The length and set of allowed characters may be changed with future product versions.

Header	Description	Example Value
OrganizationKey	The organization wide token	e.g. "1234abcd-1a2b-fedc-01a3-9876ba12cdef" or "abxdz1m8a805lhq4awnfkx8jsbrlqsup"
UserLoginName	The user name (=email address) of the user who has access to the data (e.g.: sender of an envelope)	john.doe@example.com

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.

In general: eSignAnyWhere is 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.

- 1st callback sent
- 2nd callback after 5min (only previous fails of course, e.g. does not get a "200" back)
- 3rd callback after 10min after the previous one (so 15min after the 1st)
- 4th callback after 15min after the previous one (so 30min after the 1st)
-
- xth callback after 30min after the previous one

Callback Types

- Envelope Callback
- Envelope Status Callback
- Workstep Event Callback
- 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.

Placeholder

- **##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 paramter for that envelope (e.g. internal references). Moreover, on our shared SaaS environments only HTTPS (port 443) callbacks are allowed.

Placeholder 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 paramter for that envelope (e.g. internal references). Moreover, on our shared SaaS environments only HTTPS (port 443) callbacks 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 detailed 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>
```

You can forward all eventtypes to your callback url or use the following:

- blacklist: all events, except the events in the blacklist, are fired
- whitelist: only the events in the whitelist are fired
- empty blacklist/whitelist: all events are fired

Do not use blacklist and whitelist at the same time! If you only want to use the event callbacks, use an empty envelope callback in the configuration (`<callbackUrl />`)

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 for the blacklist and whitelist definitions below.

Definition without black-/whitelist:

```
"WorkstepEventCallback": {
  "Url": "http://www.mycallback.at?
  envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##"
},
```

```

<envelope>
  ...
  <workstepEventCallback>
    <url>http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##</url>
  </workstepEventCallback>
  <steps>
    ...
  </steps>
</envelope>

```

Blacklist-Definition

```

"StatusUpdateCallbackUrl": "string",
"WorkstepEventCallback": {
  "Url": "http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##",
  "Blacklist": [
    "string"
  ]
},

```

```

<workstepEventCallback>
  <url>http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##</url>
  <blacklist>
    <event>SomeEventName</event>
    <event>SomeDifferentEventName</event>
  </blacklist>
</workstepEventCallback>

```

Whitelist-Definition

```

"StatusUpdateCallbackUrl": "string",
"WorkstepEventCallback": {
  "Url": "http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##",
  "WhiteList": [
    "string"
  ]
},

```

```

<workstepEventCallback>
  <url>http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##</url>
  <whitelist>
    <event>SomeEventName</event>
    <event>SomeDifferentEventName</event>
  </whitelist>
</workstepEventCallback>

```

Please also see the following complete configuration:

```

{
  "SspFileIds": [
    "##FileId##"
  ],
  "SendEnvelopeDescription": {

```

```

    "Name": "test",
    "EmailSubject": "Please sign the enclosed envelope",
    "EmailBody": "Dear #RecipientFirstName# #RecipientLastName#\n\nPersonalMessage#\n\nPlease sign the
envelope #EnvelopeName#\n\nEnvelope will expire at #ExpirationDate#",
    "DisplayedEmailSender": "",
    "EnableReminders": true,
    "FirstReminderDayAmount": 5,
    "RecurrentReminderDayAmount": 3,
    "BeforeExpirationDayAmount": 3,
    "DaysUntilExpire": 28,
    "CallbackUrl": "",
    "StatusUpdateCallbackUrl": "",
    "WorkstepEventCallback": {
        "Url": "http://www.mycallback.at?
envelopeId=##EnvelopeId##&recipientEmail=##RecipientEmail##&recipientOrder=##RecipientOrder##",
        "WhiteList": [
            "string"
        ]
    },
    "Steps": [
        {
            "OrderIndex": 1,
            "Recipients": [
                {
                    "Email": "##EMAIL##",
                    "FirstName": "##NAME##",
                    "LastName": "##NAME##",
                    "LanguageCode": "en",
                    "EmailBodyExtra": "",
                    "DisableEmail": false,
                    "AddAndroidAppLink": false,
                    "AddIosAppLink": false,
                    "AddWindowsAppLink": false,
                    "AllowDelegation": false,
                    "AllowAccessFinishedWorkstep": false,
                    "SkipExternalDataValidation": false,
                    "AuthenticationMethods": [
                        {
                            "Method": "Pin",
                            "Parameter": "1234"
                        }
                    ]
                }
            ]
        }
    ],
    "EmailBodyExtra": "",
    "RecipientType": "Signer",
    "WorkstepConfiguration": {
        "WorkstepLabel": "test",
        "SmallTextZoomFactorPercent": 100,
        "FinishAction": {
            "ServerActions": [],
            "ClientActions": []
        },
    },
    "ReceiverInformation": {
        "UserInformation": {
            "FirstName": "##NAME##",
            "LastName": "##NAME##",
            "Email": "##EMAIL##"
        },
        "TransactionCodePushPluginData": []
    },
    "SenderInformation": {
        "UserInformation": {
            "FirstName": "##NAME##",
            "LastName": "##NAME##",
            "Email": "##EMAIL##"
        }
    },
    "TransactionCodeConfigurations": [
        {

```

```

        "Id": "smsAuthTransactionCodeId",
        "HashAlgorithmIdentifier": "Sha256",
        "Texts": [

        ]
    },
    ],
    "SignatureConfigurations": [],
    "ViewerPreferences": {
        "FinishWorkstepOnOpen": false,
        "VisibleAreaOptions": {
            "AllowedDomain": "*",
            "Enabled": false
        }
    },
    "ResourceUris": {
        "SignatureImagesUri": "string"
    },
    "AuditingToolsConfiguration": {
        "WriteAuditTrail": false,
        "NotificationConfiguration": {}
    },
    "Policy": {
        "GeneralPolicies": {
            "AllowSaveDocument": true,
            "AllowSaveAuditTrail": true,
            "AllowRotatingPages": false,
            "AllowEmailDocument": true,
            "AllowPrintDocument": true,
            "AllowFinishWorkstep": true,
            "AllowRejectWorkstep": true,
            "AllowRejectWorkstepDelegation": false,
            "AllowUndoLastAction": false,
            "AllowAdhocPdfAttachments": false,
            "AllowAdhocSignatures": false,
            "AllowAdhocStampings": false,
            "AllowAdhocFreeHandAnnotations": false,
            "AllowAdhocTypewriterAnnotations": false,
            "AllowAdhocPictureAnnotations": false,
            "AllowAdhocPdfPageAppending": false
        },
        "WorkstepTasks": {
            "PictureAnnotationMinResolution": 0,
            "PictureAnnotationMaxResolution": 0,
            "PictureAnnotationColorDepth": "Color16M",
            "SequenceMode": "NoSequenceEnforced",
            "PositionUnits": "PdfUnits",
            "ReferenceCorner": "Lower_Left",
            "Tasks": [
                {
                    "Texts": [
                        {
                            "Language": "*",
                            "Value": "Signature Disclosure Text"
                        },
                        {
                            "Language": "en",
                            "Value": "Signature Disclosure Text"
                        }
                    ]
                },
                {
                    "Language": "*",
                    "Value": "Signature Disclosure Subject"
                },
                {
                    "Language": "en",
                    "Value": "Signature Disclosure Subject"
                }
            ]
        },
        "IsRequired": false,

```



```

        "Id": "ra",
        "DisplayName": "ra",
        "DocRefNumber": 1,
        "DiscriminatorType": "Agreements"
    },
    {
        "PositionPage": 1,
        "Position": {
            "PositionX": 63.0,
            "PositionY": 603.0
        },
        "Size": {
            "Height": 80.0,
            "Width": 190.0
        },
        "AdditionalParameters": [
            {
                "Key": "enabled",
                "Value": "1"
            },
            {
                "Key": "positioning",
                "Value": "onPage"
            },
            {
                "Key": "req",
                "Value": "1"
            },
            {
                "Key": "fd",
                "Value": ""
            },
            {
                "Key": "fd_dateformat",
                "Value": "dd-MM-yyyy HH:mm:ss"
            },
            {
                "Key": "fd_timezone",
                "Value": "datetimeutc"
            },
            {
                "Key": "spcId",
                "Value": "tLevelId"
            }
        ],
        "AllowedSignatureTypes": [
            {
                "AllowedCapturingMethod": "Click2Sign",
                "Id": "679dd763-6e25-4a68-929d-cblcel3dac7e",
                "DiscriminatorType": "SigTypeClick2Sign",
                "Preferred": false,
                "StampImprintConfiguration": {
                    "DisplayExtraInformation": true,
                    "DisplayEmail": true,
                    "DisplayIp": true,
                    "DisplayName": true,
                    "DisplaySignatureDate": true,
                    "FontFamily": "Times New Roman",
                    "FontSize": 11.0
                }
            }
        ],
        "UseTimestamp": false,
        "IsRequired": true,
        "Id": "1#XyzmoDuplicateIdSeperator#Signature_ale940eb-bcd5-2222-9777-f3570faedf3f",
        "DisplayName": "",
        "DocRefNumber": 1,
        "DiscriminatorType": "Signature"
    }
}
]
}

```

```

    },
    "Navigation": {
      "HyperLinks": [],
      "Links": [],
      "LinkTargets": []
    }
  },
  "DocumentOptions": [
    {
      "DocumentReference": "1",
      "IsHidden": false
    }
  ],
  "UseDefaultAgreements": true
},
{
  "OrderIndex": 2,
  "Recipients": [
    {
      "Email": "##EMAIL##",
      "FirstName": "##NAME##",
      "LastName": "##NAME##",
      "LanguageCode": "en",
      "EmailBodyExtra": "",
      "DisableEmail": false,
      "AddAndroidAppLink": false,
      "AddIosAppLink": false,
      "AddWindowsAppLink": false,
      "AllowDelegation": false,
      "SkipExternalDataValidation": false,
      "AuthenticationMethods": []
    }
  ],
  "EmailBodyExtra": "",
  "RecipientType": "Cc",
  "DocumentOptions": [],
  "UseDefaultAgreements": false
}
],
"AddFormFields": {
  "Forms": {}
},
"OverrideFormFieldValues": {
  "Forms": {}
},
"AttachSignedDocumentsToEnvelopeLog": false
}
}

```

These events are fired by the Workstep Controller (internal component) and are fired before the data in eSAW is complete updated (some postprocessing is required). Therefore this event callbacks are used only in rare integrations.

Available Event Types

ConfirmTransactionCode – A transaction code was sent
 DefaultEventType – Not specially defined event type
 AgreementAccepted – The user accepted the agreement
 AgreementRejected – The user rejected the agreement
 RequestPrepareAuthenticationInformationSuccess – The request for additional authentication infos was requested
 PrepareAuthenticationSuccess – The prepare authentication process succeeded
 AuthenticationFailed – The user failed to authenticate
 AuthenticationRejected – The user rejected the authentication process
 AuthenticationSuccess – The user succeeded to authenticate
 ReAuthenticationFailed – The reauthentication process failed
 AuditTrailRequested – The audittrail was requested
 AuditTrailXmlRequested – The audittrail XML was requested
 CalledPage – The viewer site was requested
 WholsInformation
 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 – The user scrolled
 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
 EnablePolicyAndValidityChecks
 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
 StartBatch – A batch signing process started
 EndBatch – A batch signing process ended
 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.xyzmo.com/Api/v4.0/envelope/create>

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

Sample:

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

Error

In general, our REST endpoint returns different HTTP status codes:

- 200 OK
- 204 NoContent (response is empty (e.g. download files))
- 40x return an error code and error info
 - 400 BadRequest (envelope description is incorrect)
 - 401 Unauthorized (User is not authorized)
 - 404 NotFound
 - 415 UnsupportedMediaType
- or HTTP 500 for generic server errors