

Table of Contents

eSignatures 5.2

Release Note

1. Release Notes 5.2.0
2. Release Notes 5.2.1
3. Release Notes 5.2.2
4. Release Notes 5.2.3
5. Release Notes 5.2.4
6. Release Notes 5.2.5
7. Release Notes 5.2.6
8. Release Notes 5.2.7
9. Release Notes 5.2.8
10. Release Notes 5.2.9
11. Release Notes 5.2.10

User Documentation

1. Introduction
2. WebPortal users
3. End users

API Documentation

Revisions

1. Introduction
2. Authentication
3. Quick Overview
4. Error Handling Responses
5. Available Package Services
6. Miscellaneous Services
7. Audit proofs
8. Queuing Mechanism
9. Signing Types
10. PDF Signing locations
11. Error Code Descriptions

Release Note

This section contains the Release Notes of eSignatures 5.2.

- [eSignatures 5.2.0](#)
- [eSignatures 5.2.1](#)
- [eSignatures 5.2.2](#)
- [eSignatures 5.2.3](#)
- [eSignatures 5.2.4](#)
- [eSignatures 5.2.5](#)
- [eSignatures 5.2.6](#)
- [eSignatures 5.2.7](#)
- [eSignatures 5.2.8](#)
- [eSignatures 5.2.9](#)
- [eSignatures 5.2.10](#)

1. Release Notes 5.2.0

Release date: 2019-02-06

1.1 New features in eSignatures 5.2.0

New supported input file format: .xml

In eSignatures 5.2 you can upload .xml documents through the API. Signers are able to verify the content of the .xml files they are asked to sign, just like with other supported file types. This way, signers can effectively see what they sign. If necessary, a .pdf representation of the .xml file can also be uploaded through the API, for signers to view their document in a more easily readable format.

New API call: Get Enabled Signing Types

This new API call returns the signing types that have been enabled in the eSignatures Configuration Index. This call is useful for integrators who need to know which signing types are currently enabled in a given eSignatures installation, and prevents them from needing to duplicate the entire list of signing types in their configuration database.

Mandated signing on itsme

In eSignatures 5.2, the mandated signing feature is also available for itsme signing. You are able to determine who exactly is mandated to sign during a particular session, based on their name and birth date.

Asynchronous signing

In eSignatures 5.2, asynchronous signing is supported. Thanks to asynchronous signing, signers no longer have to wait until all documents are signed before they may close the signing session. This comes in handy especially when signing high volumes of (large) documents. As soon as the signing has started, signers may click Close to close the signing session and continue working on other things. The signing will simply run in the background.

Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (through OpenID Connect) and Biometric.

Attention: when documents are sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.

New interface language: Spanish

The eSignatures WebPortal interface is now available in Spanish.

Configurable back button

The URL to which the back-button points when using a mobile device can now be configured in the Configuration Index. This URL used to be hardcoded to <https://www.esignatures.eu/>.

Phone number confirmation and Email address confirmation can be disabled or enabled

In the Configuration Index you can choose to disable or enable the phone number confirmation and email address confirmation.

When phone number confirmation is disabled, signers won't need to complete their phone number. Instead, they receive the sms code directly.

When email address confirmation is disabled, signers won't need to complete their email address. Instead they receive the email code directly.

Terms of use, Privacy policy and cookie policy added to the signing session

Clients can integrate their terms of use, privacy policy and cookie policy through the Configuration Index. Note that on-premise

clients are required to do so. The default Connective terms and policies are void for on-premise clients.

Signers can consult the applicable terms of use, privacy policy and cookie policy in their signing session.

1.2 Improvements

New supported regions for SMS OTP signing

A new series of countries and regions such as Mayotte and Reunion are supported for SMS OTP signing.

1.3 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-2578	/	Users who only have view rights on a document group can no longer use the Notify button.
CEP-3334	/	The test PKI tool now shows the correct message when loading eid_valid_none.xml.
CEP-3374	/	It is no longer possible to send multiple reminders immediately.
CEP-4242	26971	Fully signed documents no longer remain in Pending state after switching to the All tabs view.
CEP-4233	/	When removing signature fields defined by markers, eSignatures sometimes considered the removed fields as still requiring a signature.
CEP-4473	/	Terminology inconsistencies between WebPortal and API have been resolved.
CEP-4581	/	WYSIWYS language was not always displayed in default user language
CEP-4690	/	When canceling the OpenID Connect signing session, the signing sometimes continued to run in the background, while the package had the status 'pending' and its documents had the status 'signed'.
CEP-4691	/	When signing with OpenID Connect, eSignatures always tried to quicksign.
CEP-4698	27886	In CallbackURI with https://* the requests were not sent to https://* due to the fact that text/json was used instead of application/json
CEP-4765	/	Complex signing combined with itsme used to throw legal notice errors.
CEP-4828	/	OpenID Connect and iDIN signing could not finish face to face signing when there is a single signer on the document
CEP-4889	/	When the system logged you off automatically, the login page was shown without buttons or text.
CEP-4908	/	Corrected SMTP parameters in Installation Guide

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4933	/	Zooming issues while manual signing are fixed in Firefox
CEP-5005	/	When using API V2 to create a package and adding a document to the package with a targetType, the document was not added and an error was thrown.
CEP-5067	/	An Unsupported Media Type error was thrown when trying to use OpenID Connect signing type
CEP-5127	/	It is no longer possible to add a legal notice in the WebPortal or in the API when the legal notice setting is set to disabled in the Configuration Index.
CEP-5130	/	Legal notices on package level were not always saved on signer level
CEP-5184	28820	Some signed documents did not contain all signatures after downloading them. This was caused by the fact dots were used in text field or signature field names, which is currently not supported.
CEP-5185, CEP-5332	/	Special characters were not blocked in phone number fields.
CEP-5186	/	Error message when signing with an expired certificate has been improved.
CEP-5217	28938	Incorrect/incomplete download routes were mentioned in the External API v3 Swagger.
CEP-5238	29018	Some non-unique OperationIDs were used in the External API v3 Swagger.
CEP-5276	/	The language of the WYSIWYS on mobile could not be changed.
CEP-5289	/	When a DefaultTargetFormat is set in the Configuration Index and you add a document to a package in API v3 or create an instant package, both without using the targetType parameter, an error used to be thrown.
CEP-5293	/	Callback URL didn't work in V2 templates
CEP-5325	/	There used to be a signing field multiplication issue in the WebPortal when adding multiple signing fields for the same signer.
CEP-5361	/	API v2 "Single" Document Status call returned "Signed" for some documents but the documents couldn't be.
CEP-5375	/	Packages containing templates with user groups could not be created.

1.4 Known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome
/	/	Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
CEP-2230	/	You can place a signing field partially outside the page, but the signature will still be valid.
CEP-3779	/	Signing method is not selectable
/	/	German installation of Chrome can generate errors during signing.

1.5 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations

apply when uploading PDF documents that contain **text fields**.

- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.

- They append ?fromEmail=true to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.0, consult the **Connective - eSignatures 5.2.0 - Installation Documentation** to learn how to upgrade it to version 5.2.0.

1. Release Notes 5.2.1

Release date: 2019-02-13

1.1 New features in eSignatures 5.2.1

eSignatures 5.2.1 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5421	29329	The Package Status information for API v2 used to return the API v3 Status response text, instead of the v2 Status response text.
CEP-5422	/	The API v2 status Failed is now returned in the Get Package List and Get Document List.
CEP-5423	/	The API v3 Get Package List now sorts on status Failed and Draft.
CEP-5424	/	The API v3 Get Package List now returns the correct statuses.
CEP-5381	/	The Asynchronous signing pop-up now has a Cancel button instead of a Close button.

1.3 Known issues

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome
/	/	Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
CEP-2230	/	You can place a signing field partially outside the page, but the signature will still be valid.
CEP-3779	/	Signing method is not selectable
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.1, consult the **Connective - eSignatures 5.2.1 - Installation Documentation** to learn how to upgrade it to version 5.2.1.

1. Release Notes 5.2.2

Release date: 2019-02-26

1.1 New features in eSignatures 5.2.2

eSignatures 5.2.2 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5496	/	Signing multiple documents parallel no longer encounters deadlocks
CEP-5495	29408	The API v3 ExternalPackageReference no longer gets lost after updating the package in the portal
CEP-5478	/	Canceling a signing session now does a correct rollback
CEP-5467	/	Deleting files when using azurefile storage no longer gives errors
CEP-5402	27979	Signing a signingfield that is outside of the pdf now signs the field
CEP-4722	/	Complex pdf's upload performance enhanced when using the API

1.3 Known issues

eSignatures 5.2.2

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5489	/	Revoking a package while the last signer is finishing his signing process gives package status discrepancies.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome
/	/	Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

eSignatures 5.1.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5500	/	Can't download audit proofs when using a dss with a tenant setup and the main node has no default tenant.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
CEP-2230	/	You can place a signing field partially outside the page, but the signature will still be valid.
CEP-3779	/	Signing method is not selectable
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of

elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.2, consult the **Connective - eSignatures 5.2.2 - Installation Documentation** to learn how to upgrade it to version 5.2.2.

1. Release Notes 5.2.3

Release date: 2019-03-08

1.1 New features in eSignatures 5.2.3

eSignatures 5.2.3 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5529	/	Sequential and complex signing mails received
CEP-5528	29151	Size check on signing fields configurable in web.config
CEP-5521	/	Audit proofs can be downloaded using the document correlationid
CEP-5500	/	Can't download audit proofs when using a dss with a tenant setup and the main node has no default tenant.
CEP-5536	29524	Callbacks are triggered on status changes
CEP-5489	/	Revoking a package while the last signer is finishing his signing process no longer gives package status discrepancies.

1.3 Known issues

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed

individually.

- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be show. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.3, consult the **Connective - eSignatures 5.2.3 - Installation Documentation** to learn how to upgrade it to version 5.2.3.

1. Release Notes 5.2.4

Release date: 2019-03-27

1.1 New features in eSignatures 5.2.4

eSignatures 5.2.4 is a hotfix version and doesn't contain new features.

1.2 Improvements

Improved asynchronous signing

The asynchronous signing process has been improved by removing an unnecessary confirmation step, which speeds up the signing.

A missing callback feature has also been added. The additional callback can be done when the signing process is started, i.e. when a package enters the Command queue.

1.3 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5581	29562	The signing modal sometimes showed a "Something went wrong" error in Internet Explorer and Edge after successful signing and before redirecting.
CEP-5666, CEP-5476	/	Idin signing issues have been resolved.
CEP-5660	/	Issue when changing the document group at step 1 of the upload has been resolved.
CEP-5335	29535	Issue when parallel uploading documents to package has been resolved.
CEP-5281	/	BeLawyer mandated signing didn't work when dashes were used in the MandatedSignerIds.
CEP-5544	/	Unnecessary confirmation step in asynchronous signing has been removed.
CEP-5502	/	Missing callback in asynchronous signing has been added.
CEP-5413	/	Improved visibility feature of PDF signing fields.

1.4 Known issues

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.5 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.

- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be show. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append ?fromEmail=true to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.4, consult the **Connective - eSignatures 5.2.4 - Installation Documentation** to learn how to upgrade it to version 5.2.4.

1. Release Notes 5.2.5

Release date: 2019-04-09

1.1 New features in eSignatures 5.2.5

eSignatures 5.2.5 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-5611	/	Improved wording of the agreement statement in the WYSIWYS
CEP-5627	/	Unnecessary window was briefly shown during redirecting
CEP-5634	/	Failed packages signed with SMS or Mail OTP can be requeued from the poison queue and be successfully signed

1.3 Known issues

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be show. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.5, consult the **Connective - eSignatures 5.2.5 - Installation Documentation** to learn how to upgrade it to version 5.2.5.

1. Release Notes 5.2.6

Release date: 2019-04-29

1.1 New features in eSignatures 5.2.6

eSignatures 5.2.6 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-5379	/	Itsme scrolling issue on iPhone mobile browser has been resolved.
CEP-5686	/	Worker performance issue has been resolved.
CEP-5693	29872	When QuickSigning multiple documents with SMS OPT, the signing process kept indicating step 1, instead of steps 2 and 3.
CEP-5702	29886	Connection issue to SMTP with NTLM has been resolved.
CEP-5807	/	XML signature used to contain incorrect MIME Mediatype: application/pdf instead of application/xml.
CEP-5948	/	The database load has been reduced now the base64 of PDFs is stored in a new 'Proofs' folder in the repository instead of in the database.

1.3 Known issues

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.

- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be

upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.6, consult the **Connective - eSignatures 5.2.6 - Installation Documentation** to learn how to upgrade it to version 5.2.6.

1. Release Notes 5.2.7

Release date: 2019-05-24

1.1 New features in eSignatures 5.2.7

eSignatures 5.2.7 is a hotfix version and doesn't contain new features.

1.2 Improvements

For performance and scalability purposes, a Callback timeout has been introduced and is set to **100 seconds**. This way, if the driving application doesn't respond to eSignatures' callback, a timeout will be forced, and the rest of the flow will be finished as if the expected **200 OK** message were received. If eSignatures were to wait indefinitely for a response to finish the package, its performance would drop drastically.

For this reason, it's highly recommended that the client's callback service is developed in such a way that it sends its response as soon as possible. Any other actions done by the callback service must not depend on the response being sent but should function asynchronously.

1.3 Handled issues

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-5985	/	A Callback timeout has been set on 100 seconds.
CEP-5982	30119	WYSIWYS scrolling/zoom issues on mobile devices

1.4 Known issues

eSignatures 5.2.7

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6118	/	When enabling Audit Proofs, the Proofs folder is not automatically created for Azure file storage.
Android-42	/	The browser zooming works on Android, but not in the Connective app.

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.5 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.

- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be show. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append ?fromEmail=true to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.7, consult the **Connective - eSignatures 5.2.7 - Installation Documentation** to learn how to upgrade it to version 5.2.7.

1. Release Notes 5.2.8

Release date: 2019-06-05

1.1 New features in eSignatures 5.2.8

eSignatures 5.2.8 is a hotfix version and doesn't contain new features.

1.2 Improvements

For performance and scalability purposes, a Callback timeout has been introduced and is set to **100 seconds**. This way, if the driving application doesn't respond to eSignatures' callback, a timeout will be forced, and the rest of the flow will be finished as if the expected **200 OK** message were received. If eSignatures were to wait indefinitely for a response to finish the package, its performance would drop drastically.

For this reason, it's highly recommended that the client's callback service is developed in such a way that it sends its response as soon as possible. Any other actions done by the callback service must not depend on the response being sent but should function asynchronously.

1.3 Handled issues

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-5985	/	A Callback timeout has been set on 100 seconds.
CEP-5982	30119	WYSIWYS scrolling/zoom issues on mobile devices

1.4 Known issues

eSignatures 5.2.7

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6118	/	When enabling Audit Proofs, the Proofs folder is not automatically created for Azure file storage.
Android-42	/	The browser zooming works on Android, but not in the Connective app.
Android-43	/	Samsung tablets: browser opens in desktop mode.
CEP-6149	/	IDIN, WYSIWYS Visual progress incorrect after redirect from issuer website.

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5370	/	When you set the MandatedSigningType parameter to matchid for eID signing in the Configuration Index, it is currently not possible to upload templates.
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.5 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml files must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.

- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the `MandatedSigningType` is set to `nameandbirthdate` in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.
- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be show. To avoid this, clients can take the following actions:

- Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
- They append ?fromEmail=true to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.8, consult the **Connective - eSignatures 5.2.8 - Installation Documentation** to learn how to upgrade it to version 5.2.8.

1. Release Notes 5.2.9

Release date: 2019-07-24

1.1 New features in eSignatures 5.2.9

eSignatures 5.2.9 is a hotfix version and doesn't contain new features.

1.2 Improvements

eSignatures 5.2.9 offers a number of performance improvements.

1.3 Handled issues

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-6453	/	In some cases, packages were incorrectly in status 'Finished', while not all underlying documents were signed.
CEP-6449	30810	NotificationCallback error has been fixed.
CEP-6417	/	Out of memory issue when downloading Audit Proofs has been resolved.
CEP-6380	/	DSS keypairs for signing types are now loaded correctly in the Config Index after upgrading from a 5.2.x version.
CEP-6252	30281	NotificationCallbackUrl invalid payload issue has been resolved.
CEP-6147	/	Poison queue API call could not be used in legacy mode.
CEP-5912	/	F2F signing issue has been resolved.
CEP-5598	/	Worker uninstall could not be done without rebooting the server.
CEP-5572	29584	Document/Package incorrect status issue has been solved.
CEP-5499	/	Retry issue in Manual signing field has been resolved.
CEP-5425	28937	Manual signing issue on tablet after rotation has been solved.
CEP-5400	/	Asynchronous signing issue where you could still reject a document after you signed it and closed the modal has been solved.

JIRA CODE	ISSUES CODE	DESCRIPTION
CEP-5370	/	Template upload issues when settings the MandatedSigningType parameter to matchid for eID signing has been solved.
CEP-5230	/	Unreleased settings have been hidden in the Config Index.
CEP-4977	28476	Incorrect .dms extension issue when downloading PDFs from Safari on macOS has been resolved.
CEP-4765	/	Complex signing issue when using itsme has been resolved.

1.4 Known issues

eSignatures 5.2.9

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5961	/	A contact creation issue currently creates 2 signers instead of 1 when creating a contact during the upload process.
CEP-6151	/	Tapping a WYSIWYS link in a Samsung browser on Android tablets opens in Desktop mode.

eSignatures 5.2.7

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6118	/	When enabling Audit Proofs, the Proofs folder is not automatically created for Azure file storage.
Android-43	/	Samsung tablets: browser opens in desktop mode.
Android-42	/	The browser zooming works on Android, but not in the Connective app.
CEP-6149	/	IDIN, WYSIWYS Visual progress incorrect after redirect from issuer website.

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.5 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml file must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of

package is determined by the first uploaded document.

- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (< 1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.9, consult the **Connective - eSignatures 5.2.9 - Installation Documentation** to learn how to upgrade it to version 5.2.9.

1. Release Notes 5.2.10

Release date: 2020-08-28

1.1 New features in eSignatures 5.2.10

eSignatures 5.2.10 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8178	33451	Marker detection issue when the marker is not on the first page has been fixed.

1.3 Known issues

eSignatures 5.2.9

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5961	/	A contact creation issue currently creates 2 signers instead of 1 when creating a contact during the upload process.
CEP-6151	/	Tapping a WYSIWYS link in a Samsung browser on Android tablets opens in Desktop mode.

eSignatures 5.2.7

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6118	/	When enabling Audit Proofs, the Proofs folder is not automatically created for Azure file storage.
Android-43	/	Samsung tablets: browser opens in desktop mode.
Android-42	/	The browser zooming works on Android, but not in the Connective app.
CEP-6149	/	IDIN, WYSIWYS Visual progress incorrect after redirect from issuer website.

eSignatures 5.2.5

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5676	/	A revoked package with status "in progress" cannot be deleted.

eSignatures 5.2.4

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5564	/	When a package contains both an asynchronous and synchronous signing method, and the asynchronous signing fails, the signing session cannot be recovered.

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5605	/	When signing f2f with idin and multiple signers, only the first signer is able to sign. When not signing f2f, a signer is only able to sign the first signing field.

eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4396	/	Rebranding improvements
CEP-4817	/	Display bugs when using complex signing on Safari and Chrome

eSignatures 5.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-4719	/	The timeout of the session is currently absolute. Users are logged out even if they have been active.

Older known issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-3467	/	eID field with name+birthday validation & belaywer field (regardless of validation) incompatible
RFC-520	/	When a package contains multiple documents that must each be signed by a different user, and User A rejects the documents assigned to him, while User B accepts the documents assigned to him, the rejection details are not visible on Package level, nor on Document level for the user who accepted the documents. The rejection details are only visible on Document level to the user who rejected the documents. These details will be completed in a future version of eSignatures.
/	/	German installation of Chrome can generate errors during signing.

1.4 Known limitations

General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A package may contain a maximum of 15 documents.
- An .xml file must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- Documents whose physical dimensions exceed 3.99 m by 3.99 m are **not** supported.
- Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A.
- When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the **signature field names** only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain **text fields**.
- Adding multiple initiators within a single package is not supported.
- Combining eID signing and BeLawyer signing as choice of signing is not always supported: when the MandatedSigningType is set to nameandbirthdate in the Configuration Index, these two signing methods cannot be combined.
- Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of

package is determined by the first uploaded document.

- The default groups 'administrators and default user group' can't be used as a signer in templates.
- Inserting a tab in front of a Text Marker in Word is not supported. Instead of using tabs, use tables, columns or text boxes. If you still want to use tabs, convert your Word document to PDF before uploading it.
- When using Safari: after you upgrade from an older version, you will be prompted to quit your browser. When reopening your browser and your previous tabs don't open automatically, do not reuse your original link to the signing page. Instead, go to **History > Reopen All Windows From Last Session**.
- Native design applications for DTP, CAD, etc. can generate documents with a high level of complexity (very high number of elements, vectors, images, ...). This may result in the fact that the application cannot prepare the document in a reasonable timeframe. Consequently, it will be impossible to add these files to the Signing environment (not through the API nor in the DocumentPortal) due to timeout. It is not possible to know in advance if a complex document will generate a timeout or not as it depends on too many factors. The applications generating these kinds of PDFs usually have a setting to create a PDF that is suitable for online usage. We strongly recommend using this setting to reduce the complexity of the document before upload.

Itsme signing

- When using itsme as signing method, the target type of your documents must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages through the WebPortal, each document within the package must be signed individually.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Audit Proofs

- The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.
- The Audit proof feature also has an impact on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (< 1MB) do not seem to impact the signing speed.

API v2 to v3 migration

Legacy Mode

If clients are not able to upgrade to API v3, because they're not ready to support one-time URLs or access tokens, they can use the Legacy mode setting combined with version 2 of the API. Legacy mode is a transitional measure that supports 'old' URLs and allows you to complete unfinished documents. Please be aware that this is a deprecated feature and will be removed in version 4 of the API.

The following limitations apply:

- When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.
- When Legacy mode is set to disabled, the download URLs retrieved from API v2 can only be used once. These URLs should not be sent directly to the end user in an email, as the second time the URL is opened, a non-user-friendly error message will be shown. To avoid this, clients can take the following actions:
 - Either they email an URL to their application so that they always redirect from their application to the newest download URL retrieved from API v2 just before redirecting.
 - They append `?fromEmail=true` to the URL retrieved from API v2 so that we can show an error screen asking the end user to enter their email address. If that email address corresponds with a signer or receiver and the package / document exists they should get an email from Connective with a new, working URL.

2. Upgrade Information

For customers signing with physical tokens (eID, biometric using Wacom, etc.), the Connective Browser Package must be upgraded to version 2.0.6. The upgrade should be prompted automatically when trying to sign a document.

If you have a version of eSignatures installed prior to version 5.2.10, consult the **Connective - eSignatures 5.2.10 - Installation Documentation** to learn how to upgrade it to version 5.2.10.

User Documentation

This section contains the User Documentation of eSignatures 5.2.

1. Introduction

Welcome to the eSignatures 5.2 User documentation.

This documentation consists of two parts: one intended for **WebPortal users** and one intended for **End users**.

In the **WebPortal** part you'll learn how to use the eSignatures WebPortal: how to upload documents and packages, add signers and receivers, configure the signing methods, and sign documents in all possible ways. You'll also learn how to manage contacts, manage users and create templates.

In the **End user** part end users will learn how to sign the documents and packages received by email.

At the end of each part you'll find an **FAQ** and **Troubleshooting** section.

Important note: since eSignatures now offers the same functionalities for documents and packages, from now on the term "document" refers to both documents and packages to improve the readability of the documentation. Only where the behavior is specific to packages do we use the term "package".

1.1 Revisions

DATE	OWNER	TOPIC
2018-05-07	DGI	Document Creation
2018-05-15	DGI	Improved layout
2018-05-25	DGI	Release version
2018-06-08	DGI	Final release 5.0
2018-07-09	DGI	Added info on Compatibility View
2018-07-24	DGI	Added info on password policy + FAQ on logging in
2018-08-08	DGI	Version 5.1.0
2018-08-20	DGI	Removed references to QRConnect
2018-08-23	DGI	Combination of sequential and parallel signing
2018-09-03	DGI	Added info on regex
2018-09-19	DGI	Legal notice on signer level
2018-09-19	DGI	itsme signing
2018-10-23	DGI	Finalization
2018-11-06	DGI	Move Browser package info to separate section
2018-11-07	DGI	Version 5.1.1: Microsoft Edge Supported
2019-01-07	DGI	Added info on phone number format
2019-02-06	DGI	Version 5.2
2019-02-11	DGI	Updated supported OS
2019-03-26	DGI	Update to version 5.2.4
2019-04-09	DGI	Update to version 5.2.5
2019-04-29	DGI	Update to version 5.2.6
2019-05-14	DGI	Update to version 5.2.7

1.2 Copyright and legal notices

5.2.7-14052019-01

This documentation is provided for informational purposes only, and Connective and its suppliers make no warranties, either express or implied, in this documentation. Information in this documentation, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this documentation remains with the user.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this documentation may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Connective.

© 2019 Connective. All rights reserved.

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apple and Mac are trademarks of Apple, Inc., registered in the U.S. and other countries.

Wacom is a registered trademark of Wacom Co., Ltd.

Any additional company and product names mentioned in this documentation may be trademarked and/or registered as trademarks. Mention of third-party products constitutes neither an endorsement nor a recommendation. Connective assumes no responsibility with regard to the performance or use of these products.

2. WebPortal users

The **WebPortal users** section is intended for all WebPortal users. In this section you'll learn how to use the eSignatures WebPortal: how to upload documents and packages, add signers and receivers, configure the signing methods, and sign documents in all possible ways. You'll also learn how to manage contacts, manage users and create templates.

2.1 What is eSignatures?

Connective eSignatures is a fast, user-friendly and secure platform to upload, sign and distribute digital documents. It is an end-to-end solution that includes a wide range of signing methods from eID to SMS and allows you to configure the order in which signers need to sign.

Workflow

Preparing documents for signing is very simple: you upload your documents, determine who should sign, which signing method they should use, and then send the documents via secured email. Optionally, you can also determine who should receive a copy of the fully signed documents.

You can also send documents to multiple signers and define the order in which they should sign. This way, your documents end up with the right signers at the right time. The signers, on their end, can sign their documents anywhere, anytime and on any device. They simply click the secured link in their email to open the documents for signing, or log in to the Signer Portal using a web browser. A rebrandable Connective app for iOS and Android is also available.

User roles

The eSignatures workflow contains 3 user roles:

Initiator: the eSignatures user who uploads the documents, determines who should sign them and which signing method they should use.

Signer: the person who needs to sign documents.

Receiver: the person who needs to receive a copy of the fully signed documents.

Note: documents must always be uploaded via the Document Portal. The Document Portal can be accessed from a computer or from a tablet, it cannot be accessed from a smartphone. Also note that the Connective app is not intended to upload documents.

2.2 What's new in eSignatures 5.2?

2.2.1 New features

New supported input file format: .xml

In eSignatures 5.2 you can upload .xml documents through the API. Signers are able to verify the content of the .xml files they are asked to sign, just like with other supported file types. This way, signers can effectively see what they sign. If necessary, a .pdf representation of the .xml file can also be uploaded through the API, for signers to view their document in a more easily readable format.

New API call: Get Enabled Signing Types

This new API call returns the signing types that have been enabled in the eSignatures Configuration Index. This call is useful for integrators who need to know which signing types are currently enabled in a given eSignatures installation, and prevents them from needing to duplicate the entire list of signing types in their configuration database.

Mandated signing on itsme

In eSignatures 5.2, the mandated signing feature is also available for itsme signing. You are able to determine who exactly is mandated to sign during a particular session, based on their name and birth date.

Asynchronous signing

In eSignatures 5.2, asynchronous signing is supported. Thanks to asynchronous signing, signers no longer have to wait until all documents are signed before they may close the signing session. This comes in handy especially when signing high volumes of (large) documents. As soon as the signing has started, signers may click Close to close the signing session and continue working on other things. The signing will simply run in the background.

Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (through OpenID Connect) and Biometric.

Attention: when documents are sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.

New interface language: Spanish

The eSignatures WebPortal interface is now available in Spanish.

Configurable back button

The URL to which the back-button points when using a mobile device can now be configured in the Configuration Index. This URL used to be hardcoded to <https://www.esignatures.eu/>.

Phone number confirmation and Email address confirmation can be disabled or enabled

In the Configuration Index you can choose to disable or enable the phone number confirmation and email address confirmation.

When phone number confirmation is disabled, signers won't need to complete their phone number. Instead, they receive the sms code directly.

When email address confirmation is disabled, signers won't need to complete their email address. Instead they receive the email code directly.

Terms of use, Privacy policy and cookie policy added to the signing session

Clients can integrate their terms of use, privacy policy and cookie policy through the Configuration Index. Note that on-premise clients are required to do so. The default Connective terms and policies are void for on-premise clients.

Signers can consult the applicable terms of use, privacy policy and cookie policy in their signing session.

2.2.2 Improvements

New supported regions for SMS OTP signing

A series of countries and regions such as Mayotte and Reunion are supported for SMS OTP signing.

2.3 Accessing my account

2.3.1 How do I sign up for an eSignatures account?

2.3.2 How do I log in to my account?

2.3.3 How do I log out?

2.3.4 How do I change my profile settings?

2.3.1 How do I sign up for an eSignatures account?

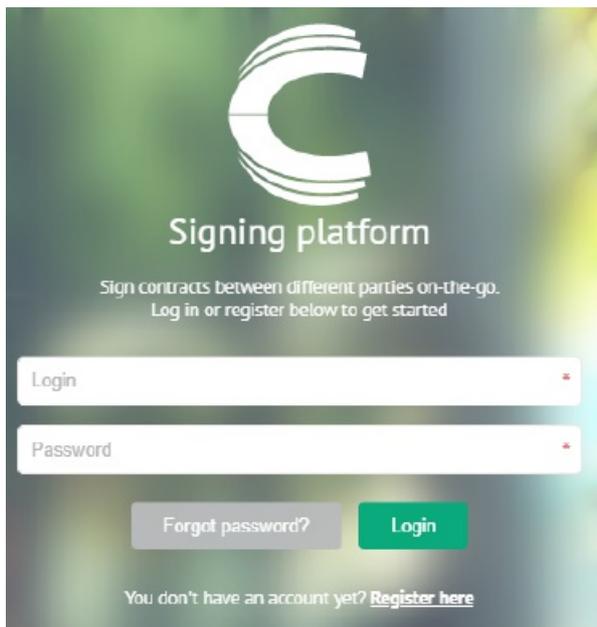
Whether you need to sign up for an eSignatures account depends on the configuration of the eSignatures solution.

If you already received credentials from your administrator, you don't need to sign up. You can log in directly. See [How do I log in to my account?](#) for more information.

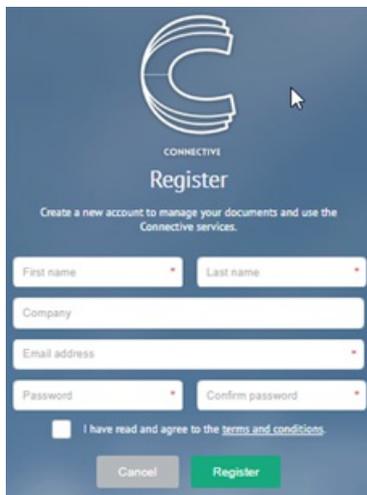
To sign up for an eSignatures account:

- Go to the link you received from your administrator. In our standard demo environment this link is <https://www.esignatures.eu>.
- Click **Register here** at the bottom of the page.

Note: if **Register here** is not displayed at the bottom of the page, this means you can't register yourself. Contact your administrator to obtain your login credentials.



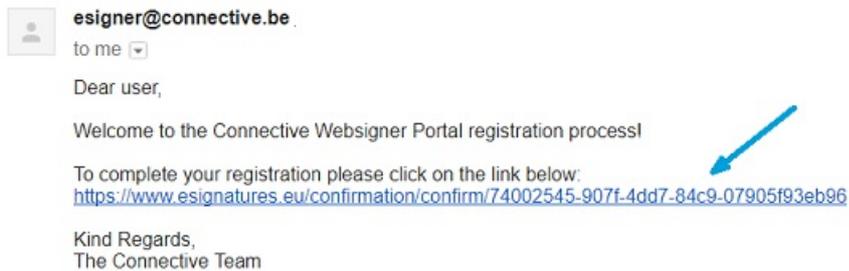
- Fill out your personal information:
 - First name
 - Last name
 - Company (Optional)
 - Email address
- Enter a password of your choice and confirm the password. The password must meet the following requirements:
 - Contain at least 6 characters, and no more than 127 characters.
 - Contain at least 1 uppercase letter.
 - Contain at least 1 lowercase letter.
 - Contain at least 1 number.
 - Contain at least 1 special character (e.g. whitespace, symbol, etc.)
- Read the terms and conditions, and then select **I have read and agree to the terms and conditions**.



The image shows a registration form for Connective. At the top, there is a logo consisting of a stylized 'C' made of horizontal lines, with the word 'CONNECTIVE' underneath it. Below the logo, the word 'Register' is displayed in a large font. Underneath 'Register', there is a smaller line of text: 'Create a new account to manage your documents and use the Connective services.' The form itself consists of several input fields: 'First name' and 'Last name' (both with dropdown arrows), 'Company', 'Email address' (with a dropdown arrow), 'Password' (with a dropdown arrow), and 'Confirm password' (with a dropdown arrow). Below these fields, there is a checkbox labeled 'I have read and agree to the terms and conditions'. At the bottom of the form, there are two buttons: 'Cancel' and 'Register'.

- Click **Register**. An email will be sent to your email account.
- Check your email and open the email you received from your eSignatures solution.
In our standard demo environment you receive an email from esigner@connective.be.
Tip: if you don't see the email in your Inbox, check your Spam or Junk folder.
- Click the link in the email to complete the registration.

Registration Process Confirmation Inbox x

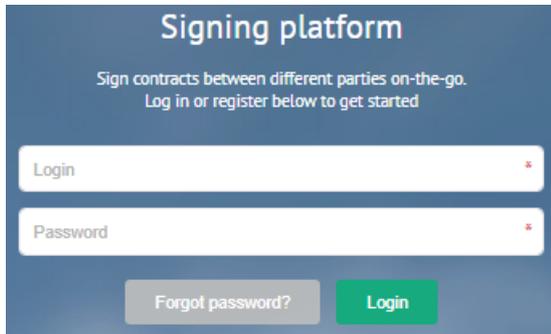


- In the window that opens, click **Confirm registration**.
- Congratulations, you're now registered as a new user and you can [log in](#) to your account.

2.3.2 How do I log in to my account?

- Go to the link you received from your administrator. In our standard demo environment this link is <https://www.esignatures.eu>.
- Enter your email address in the **Login** field.
- Enter your password in the **Password** field.

Tip: the password is the one you've chosen when [signing up](#) for an eSignatures account, or that was assigned to you by your administrator.



- Click **Login**.

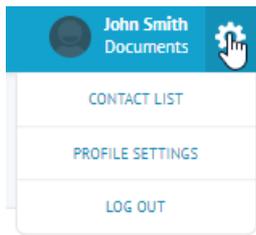
Tip: in case you forgot your password, click **Forgot Password?** Then enter your email address and click **Reset password**. A email will be sent to reset your password.

Note: in case you don't have an account yet, see [How do I sign up for an eSignatures account?](#)

Tip: if you're using the standard demo environment of eSignatures, you can also download the Connective app (iOS and Android). The [Connective app](#) is required on iOS and Android devices when using any signing method that needs additional hardware (smart card reader, signature pad, etc.) A rebrandable app may be available for your company's eSignatures solution. Contact your administrator for more information.

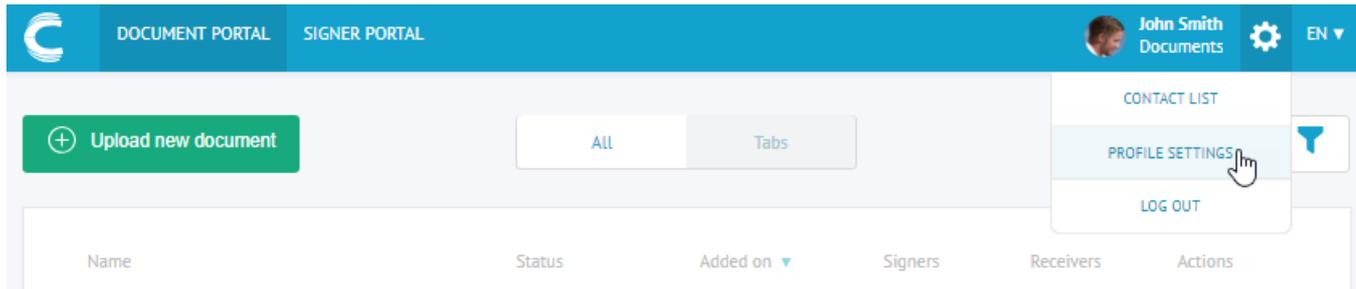
2.3.3 How do I log out?

- Click the settings icon in the top toolbar.
- Click **Log out**.



2.3.4 How do I change my profile settings?

- [Log in](#) to your account.
- Click the settings icon  in the top toolbar, and click **profile settings**.



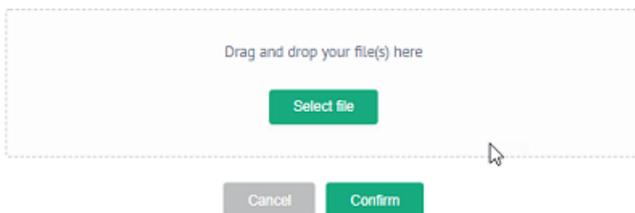
In profile settings you can do the following actions:

- Upload/change your profile picture
- Change your personal information
- Choose your language settings
- Reset your password
- Change your email address
- Unregister your account

These actions are described below.

Upload/change your profile settings

- Click the settings icon  in the top toolbar, and click **profile settings**.
- Click the picture icon to upload a picture.
- Click **Select file** to browse for a picture. Or drag and drop a picture to the dotted frame.



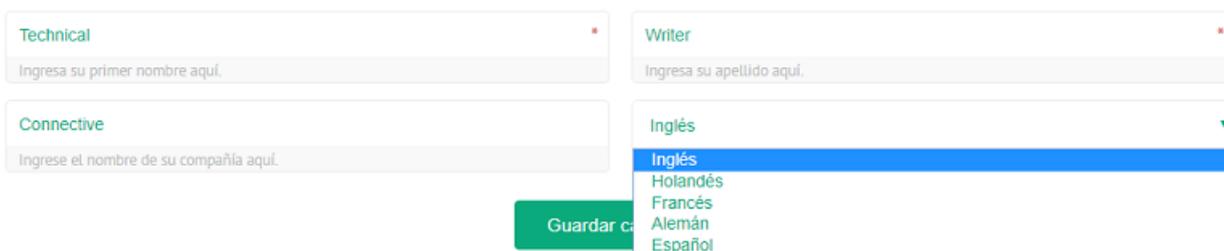
- Click **Confirm**.

Change your personal information

- Click the settings icon  in the top toolbar, and click **profile settings**.
- Click inside the fields to change your first name, last name and company.
- Click **Save changes**.

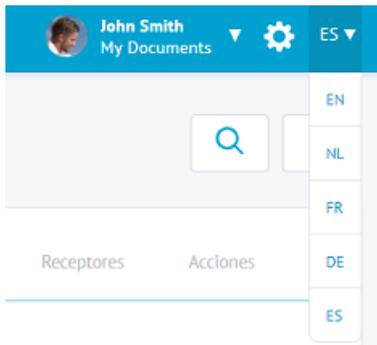
Choose your language settings

- Click the settings icon  in the top toolbar, and click **profile settings**.
- Select the preferred language from the language drop-down list: English, Dutch, French, German or Spanish. The emails you will receive are written in this language.

The image shows a portion of the profile settings form. It contains four input fields: 'Technical' (with a red asterisk), 'Connective', 'Writer' (with a red asterisk), and a language dropdown menu. The language dropdown is open, showing a list of languages: Inglés, Holandés, Francés, Alemán, and Español. A green button labeled 'Guardar cambios' is visible at the bottom.

- Click **Save changes**.

Note: changing these language settings does not change the interface language. To change the interface language, click the language list in the top toolbar and select the language of your choice, as displayed in the image below.



Reset your password

- Click the settings icon  in the top toolbar, and click **profile settings**.
- Click **Reset password**.
- Enter your current password.
- Enter your new password and confirm the new password.
- Click **Confirm**.
- You now need to log in again, using your new password.

Change your email address

- Click the settings icon in the top toolbar and click **profile settings**.
- Click **Change email address**.
- Insert your new email address and current password. **Note:** enter the password of your eSignatures account, not the password of the new email address.
- Click **Confirm**. An email is sent to your email address.

 A screenshot of a web form titled 'Change your current email address.' The form contains two input fields: 'Email address' and 'Current password'. Below the 'Email address' field is a placeholder text 'Enter a valid email address.' Below the 'Current password' field is a placeholder text 'Fill in your current password.' At the bottom of the form are two buttons: 'Cancel' and 'Confirm'.

- Now log out from your account.
- Open the email you received from your eSignatures solution. In our standard demo environment you receive an email from **esigner@connective.be** with the subject line **Change Email Confirmation**.
- Click the link inside the email to complete the process. You can now log in using your new email address.

Unregister your account

Attention: unregistering an account is irreversible. Once you've unregistered an account you need to register it again - or your administrator has to do it for you - in case you want to use your account again.

- Click the settings icon  in the top toolbar and click **profile settings**.
- Click **Unregister**.
- Enter the email address you want to unregister.
- Click **Confirm**.

Are you sure you want to unregister your user account?

Enter your email address to confirm you want to unregister your account.
Unregistering a user account cannot be undone.

Email address

Enter a valid email address.

Cancel

Confirm

2.4 Uploading documents

[2.4.1 Document Portal overview](#)

[2.4.2 How to prepare documents for upload](#)

[2.4.3 How do I upload documents?](#)

[2.4.4 How do I view my uploaded documents?](#)

[2.4.5 How do I edit a draft document?](#)

[2.4.6 How do I send a notification to the signer?](#)

[2.4.7 How do I revoke a document?](#)

[2.4.8 How do I delete a document?](#)

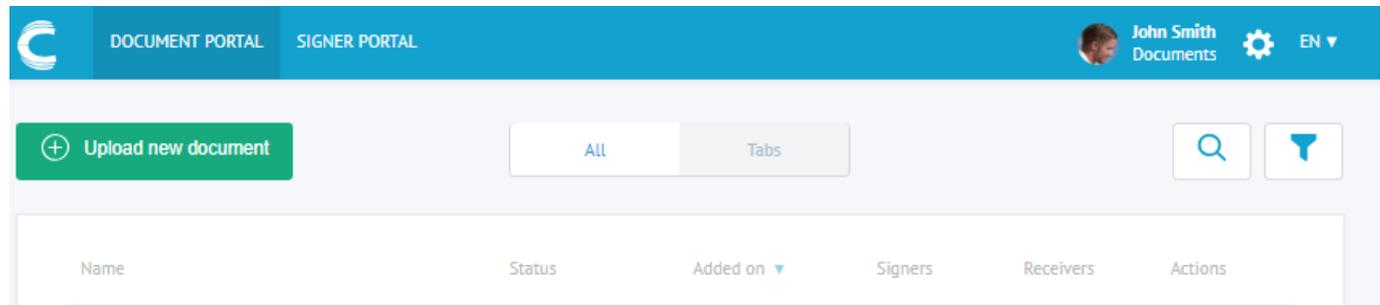
[2.4.9 How do I extend the expiration date?](#)

[2.4.10 How do I download a document?](#)

2.4.1 Document Portal overview

The Document Portal is where you upload documents that need to be signed, determine who should sign them, which signing method they should use, and then send the documents via secured email. Optionally, you can also determine who should receive a copy of the fully signed documents.

In the Document Portal you have an overview of all the documents you uploaded. You can see their status, when they have been added, to whom they have been sent, and which actions can be done.



When you use the Document Portal for the first time, the document list is empty.

- To start uploading documents, click **Upload new document**.

Important: before you upload a document, read the section **How to prepare documents for upload?** for some guidelines and best practices.

- To view the **document groups** you have access to, click the down list next to your user name. When you've selected a document group, only the documents belonging to the selected document group are displayed.
- To search for specific documents or filter search results, use the search  and filter  buttons. See **How do I view my uploaded documents?**
- To change the settings, click the settings icon . In a default configuration you have access to the **Contact List** and to your **Profile Settings**. As administrator you also have access to the **User Management** settings.
- To change the interface language, click the language drop-down list and select another language.
- To log out, click the settings icon  and click **Log out**.

Tip: also check the **video tutorial**:

2.4.2 How to prepare documents for upload?

Before you try to upload documents to eSignatures, please take into account the following limitations and guidelines.

Limitations

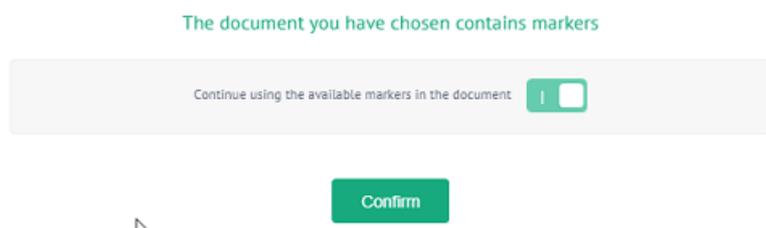
- A package containing multiple documents must not exceed 150 MB.
- A package may contain a maximum of 15 documents.
- A single document must not exceed 30 MB.
- The physical dimensions of a document must not exceed 3.99 m by 3.99 m.

We recommend you do *not* upload files that exceed these limits. Depending on the internet connection, large documents may affect user experience and signing performance. Documents that exceed the specified limitations are officially not supported.

Guidelines

- You can upload the following file formats: .doc, .docx, .txt and .pdf. Make sure the documents you upload comply with the standards of these file formats. Uploading poor quality documents will result in poor output. Note however that not all file formats may have been enabled in your eSignatures solution depending on the configuration.
- When you're uploading documents that already contain digital signatures - whether they were signed in eSignatures or in another signing application - make sure you don't select *any* option that alters the documents, like selecting a different output format for instance. Any alteration to the documents will render the existing signatures invalid.
- When your documents already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - you can choose to sign them in eSignatures or sign them in another application after the eSignatures signing flow has ended. To sign them in eSignatures, keep the option **Continue using the available markers in the document** selected when prompted. To sign them in another application, disable this option.

Important: make sure the signature field names only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported.



- When you upload multiple documents inside a package, make sure all documents have the same document language. Otherwise problems may occur during signing. If you need to upload documents in different languages, do so in separate packages.

Text fields

Pay attention if the PDF documents you upload contain **editable text fields**. If the name of a such a text field corresponds to the **Text Field Format** (regex) you or your administrator configured in the **Configuration Index**, the text field will be converted to a signature field in eSignatures, and the original text field will not be displayed. This is intended behavior. So if you want to upload PDF documents with editable text fields, and prevent eSignatures from converting them into signature fields, make sure the name of the text field you create in your PDF Solution does not correspond to the **Text Field Format** that has been defined in the Configuration Index of eSignatures. The default regex format used in the Configuration Index is `#[a-zA-Z]+(?:\d*)?\d+`

Important: make sure the text field names only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported.

Regex explanation

- The leading # means the Text Field must start with a hashtag
- `[a-zA-Z]` matches any character from a to z, in small or capital letters

- ?: means the preceding item is optional and matches at most once
- \d is a metacharacter that matches any digit, which is identical to [0-9]
- '*' means 0 or more instances of the preceding token
- . matches the . character
- ? is an occurrence indicator denoting 0 or 1 occurrence (i.e. optional)
- The + indicates 1 or more occurrences of the previous sub-expression

Note that Text Fields can't be added in eSignatures itself, but must be added to the document before upload.

Text Markers

Instead of adding signature fields one by one in eSignatures, you can choose to add **Text Markers** to the documents you upload. Text Markers are pieces of code that indicate where the **signature fields** must be placed and which dimensions they must have. This can be useful if all your signature fields must be placed at the same location and require a specific size.

Tip: the location in the document where you put the Text Marker is where the signature field will be placed.

Notes:

- Text Markers must correspond to the **Text Marker Format** (regex) defined in the Configuration Index. Otherwise the signature field won't be placed correctly, or not placed at all. The default format for searching Text Markers in a document (regex) is `#[a-zA-Z]+(?:\d)?\d+_(?:\d)?\d+_(?:\d*)?\d+#`

Regex explanation

- The leading and trailing # means the Text Marker must start and end with a hashtag
- [a-zA-Z] matches any character from a to z, in small or capital letters
- The + indicates 1 or more occurrences of the previous sub-expression
- ?: means the preceding item is optional and matches at most once
- \d is a metacharacter that matches any digit, which is identical to [0-9]
- '*' means 0 or more instances of the preceding token
- . matches the . character
- ? is an occurrence indicator denoting 0 or 1 occurrence (i.e. optional)
- _ means the regex must contain an underscore at the indicated positions

In practice this means a Text Marker should always be in the following format `#SIGidentifier_Height_Width#`.

E.g. `#SIG01_100_200#`. In this example case a signature field of 100 pixels high and 200 pixels wide will be placed. When adding a Text Marker, always put the Height value (minimum 70 pixels) in front of the Width value (minimum 112 pixels).

Note: it is recommended to use slightly higher values than the minimum ones. E.g. 75 pixels x 120 pixels. Using the absolute minimum values may lead to round-off errors during conversions.

- Text Markers should not be combined with rotated PDFs because detected signature fields won't be rotated automatically to match the PDF text direction. They'll be placed near the text marker on a best-effort approach.
- Text Markers cannot be added in eSignatures itself, but must be added to the document before you upload it.

2.4.3 How do I upload documents?

Step 1: [upload documents](#)

Step 2: [define signing fields](#)

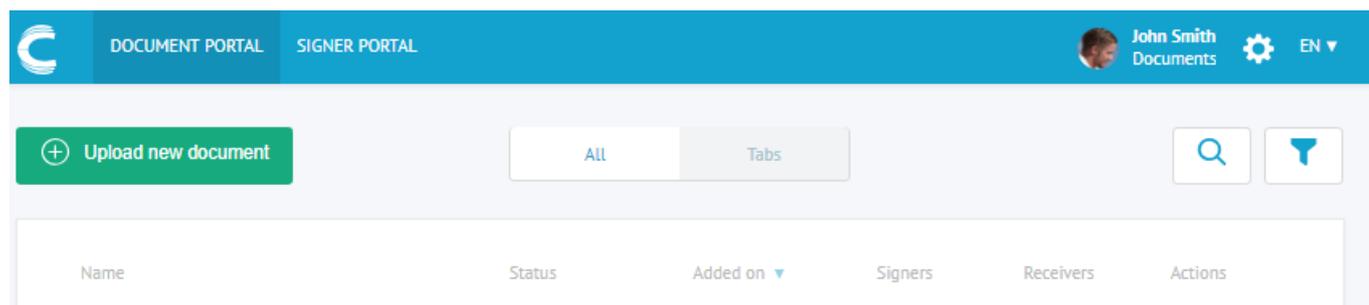
Step 3: [send the documents](#)

Step 1: upload documents

Important: before you upload documents, read the section [How to prepare documents for upload?](#) for some guidelines and best practices.

Tip: also check the [video tutorial](#) on how to upload documents:

To start uploading documents, click **Upload new document** in the Document Portal.



Select the files you want to upload

In a default configuration, files can be uploaded in 2 ways: through the file explorer and by drag-and-drop. If the corresponding setting has been enabled in the Configuration Index, you can also import documents from a Cloud account (Dropbox, Google Drive and OneDrive).

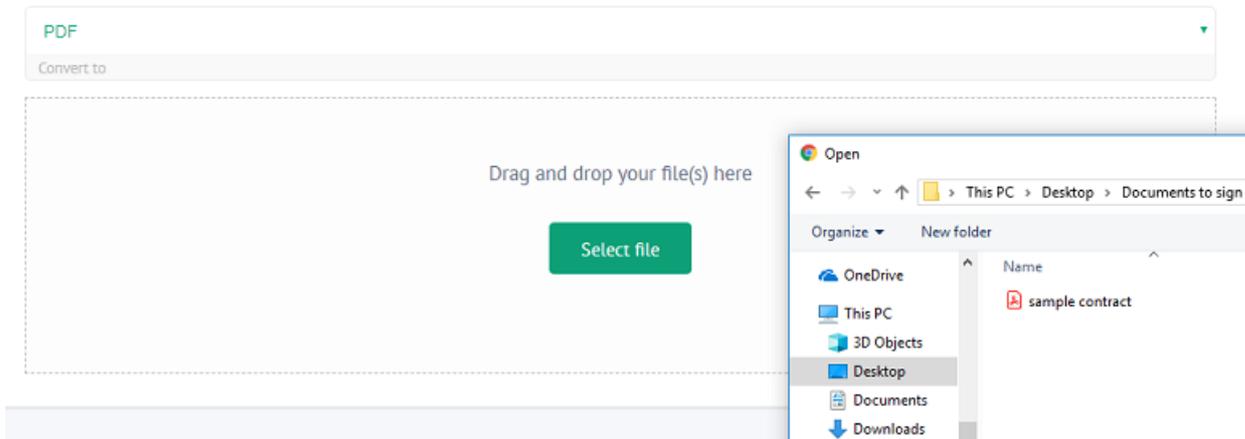
If any of these methods are not available in your solution, it means your eSignatures environment has been configured that way.

The file formats that can be uploaded are: .doc, .docx, .txt and .pdf. Note that not all these file types may be available in your eSignatures solution, depending on its configuration.

Select files through the file explorer

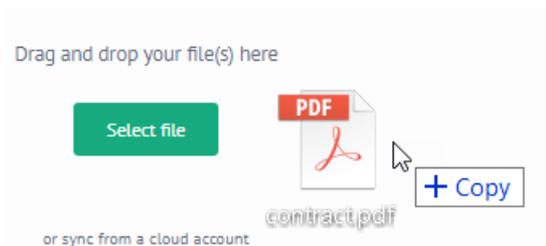
- Click **Select file**.
- In the window that opens, browse for the documents you want to upload.
- Ctrl-click or Shift-click the required files and click **Open**.

Important: we recommend you don't upload more than 15 documents inside a package.



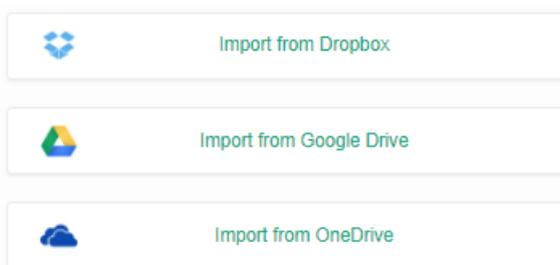
Select files by drag-and-drop

- Drag and drop the required file to the dotted frame.



Select files from a Cloud account

These options are only available if they have been properly configured by your administrator in the Configuration Index. To learn how to configure them, see the **Connective - eSignatures 5.2.7 - Configuration Documentation**.



Import from Dropbox

- Click **Import from Dropbox**.
- Click **Sign in with Google** to use your Google account as sign in method.

OR

- Enter your **Email** and **Password** and click **Sign in**. If you don't have a Dropbox account yet, click **create an account** to create one.
- Browse for the file you want to upload and click **Choose**.

Import from Google Drive

- Click **Import from Google Drive**.
- Choose the Google account you want to use.
- Select **Allow** when you're asked whether eSignatures may view and manage your Google Drive files.

- Browse for the files you want to upload and click **Select**.

Import from OneDrive

- Click **Import from OneDrive**.
- Select the Microsoft account you want to use.
- Enter your **Email** or **Phone** and **Password** and click **Sign in**.
- Click **Accept** when prompted to give Connective eSignatures permissions.
- Browse for the files you want to upload and click **Open**.

Removing files

If you accidentally selected the wrong files, you can always remove them by clicking the x icon.



Select the output format

Select the output format you want to generate. In a default configuration you can convert your input files to the following output formats: **PDF**, **PDF/A-1** and **PDF/A-2**.

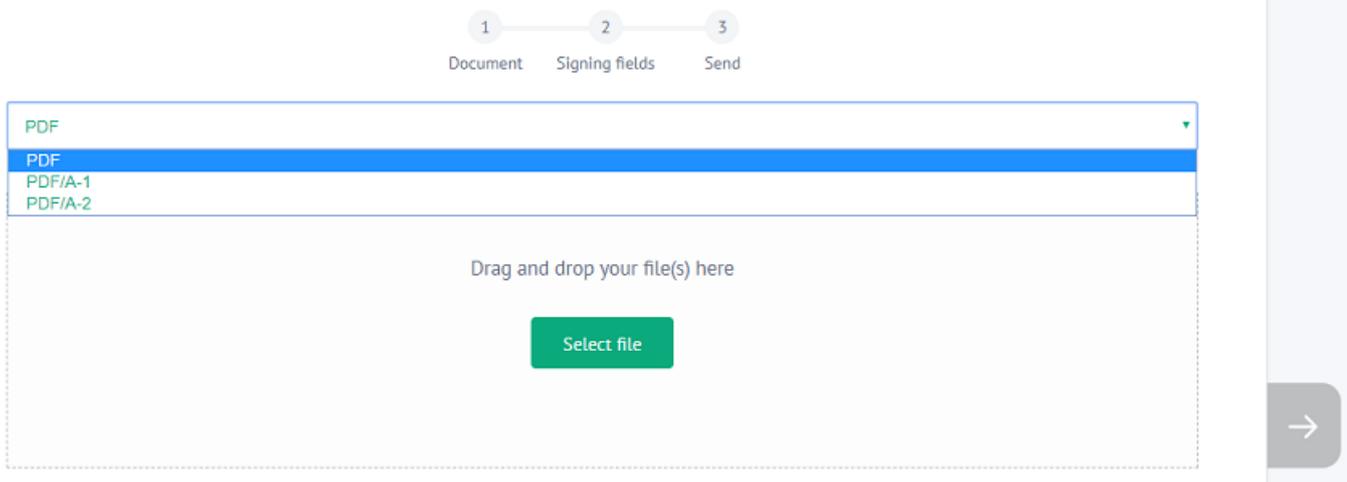
PDF is the standard PDF format.

PDF/A-1 is a standard long-term archiving format and is the constrained version of Adobe PDF version 1.4.

PDF/A-2 is also a standard long-term archiving format and is the constrained version of Adobe PDF version 1.7.

Both PDF/A formats prohibit features that are ill-fitted for long-term archiving.

Important: when using **itsme** as signing method, it is mandatory to use **PDF/A-1** or **PDF/A-2** as output format. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solutions, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.



When the file has been uploaded, click the arrow to go to the next step.

PDF

Convert to

Select file

Selected file(s)

sample contract.pdf
(198KB)

→

Choose whether to keep using the available markers (if applicable)

If the document you're uploading contains Text Markers and/or Text Fields that correspond to the respective Text Marker Format and Text Field Format that was defined in the Configuration Index, the Text Markers and Text Fields will be automatically detected.

Text Markers are pieces of 'code' you can add to a document, which indicate where a signature field must be placed and which dimensions it must have.

Text Fields are editable form fields inside a PDF. eSignatures can convert these text fields into a signature field.

Note that in both cases the format must correspond to the format that was defined in the Configuration Index.

By default, eSignatures will continue using the available markers. This means the signature fields will be created at the markers' position. This feature speeds up the document upload.

The document you have chosen contains markers

Continue using the available markers in the document

Confirm

- To continue using the available markers, click **Confirm**. The signature field(s) will be placed on the markers' position.
- To continue without the available markers, click the markers button to disable it, and then click **Confirm**. You will need to drag your signature field(s) to the required position.

Determine the Properties

- Enter a title (optional).
- Select the document group to which you want to upload the document. The default document group is My Documents.
- Select the document language. You can add documents in English, French, Dutch, German and Spanish. When you choose to add legal notices, the default legal notices will be written in the language you select here. If you're uploading a document of a different language, select one of the languages that the signer understands.

Legal notice

Decide whether to add a **legal notice** to your documents.

Attention: When you enable the legal notice setting in this screen, the corresponding legal notice setting will be enabled for each signer you add. Note however, that for each signer you can modify this default legal notice and choose to configure a personal one. Personalizing the legal notice per signer is done in [Step 2: define signing fields](#).

Important:

When you add a legal notice, the signers need to retype the exact content of the legal notice before they're able to sign the document.

When you add a legal notice to a package that contains multiple documents, each signature field will need to be signed individually.

- Click the legal notice button  to enable the legal notice settings.
- Three predefined legal notices are available in the drop-down list. The language of the legal notice corresponds to the document language you selected.

Do you want to add a legal notice?

Read and approved

Read and approved

Read and approved for [AMOUNT] EUR. Signed at [PLACE] on date [DATE]

Guarantor liable for an amount of [AMOUNT] EURO for [TOPIC]

Read and approved

Edit the legal notice.

- Modify the content of the legal notices to your liking. For instance enter variables between square brackets. When you do so, the signer needs to enter a value for each variable when signing the document.

Read and approved for [AMOUNT] EUR. Signed at [PLACE] on date [DATE]

Select the preferred legal notice.

Read and approved for [AMOUNT] EUR. Signed at [PLACE] on date [DATE]

Important: limit the legal notice to 255 characters including any predefined content. Otherwise an error will occur during

signing.

Note when using biometric signing: the Wacom signature pads display only a limited number of characters on the signing screen. The STU-430 approximately displays 24 characters over 1 line of text, while the STU-530 displays approximately 74 characters over 2 lines. If the legal notice contains more characters, they fall off the screen. You're recommended to run some tests beforehand to see if the text fits.

- To add additional documents to your package, click **Add additional documents**.

Note: this button is only visible if you already added more than one document.

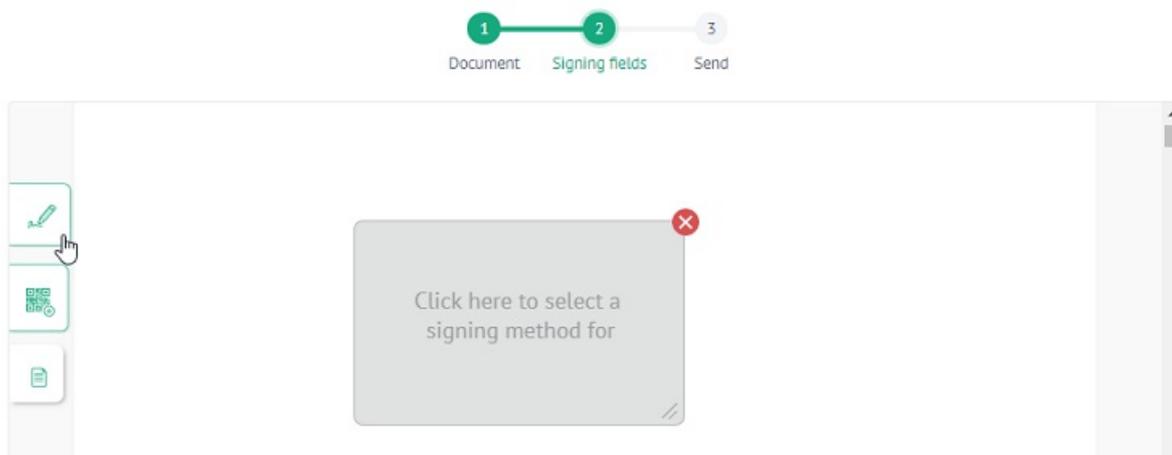
- Click the arrow to go to the next step.

Note: your documents are automatically saved as draft when you quit the upload flow after step 1.

Step 2: define signing fields

Add signing fields

- Scroll to the page where want to add the signing field.
- Click the signature icon to add a signing field.
- Drag the signing field to the required position on the page.



Technical Notes:

Signature fields that have been placed by using **Text Markers** in the upload document cannot be moved; these signature fields are automatically placed on the position defined by the Text Markers. Note however, that the Text Markers must correspond to the **Text Marker Format** the administrator defined in the Configuration Index, and that you must have confirmed to use Text Markers in [Step 1](#) of the upload process.

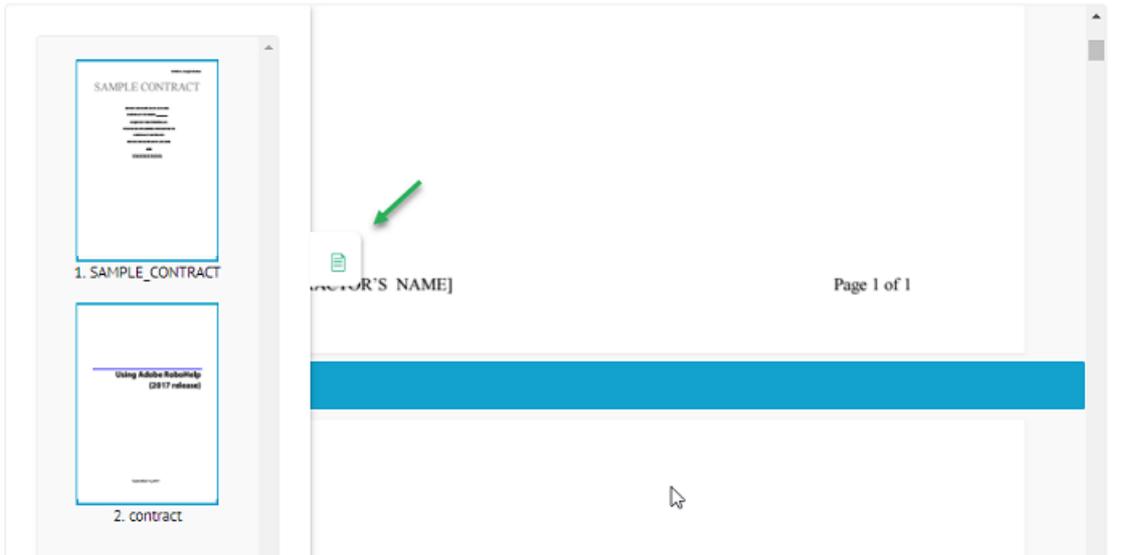
If your documents contain **Text Fields**, and their format corresponds to the **Text Field Format** the administrator defined in the Configuration Index, the signing fields will be placed inside those Text Fields. Again, provided you confirmed to use Text Markers in [Step 1](#) of the upload process.

Important: When you uploaded multiple documents, make sure to add at least one signing field per document:

- Click the document icon  to have an overview of all your documents inside the package.
- Click a document thumbnail to go to the corresponding document, click inside the document, and then add the signing field.

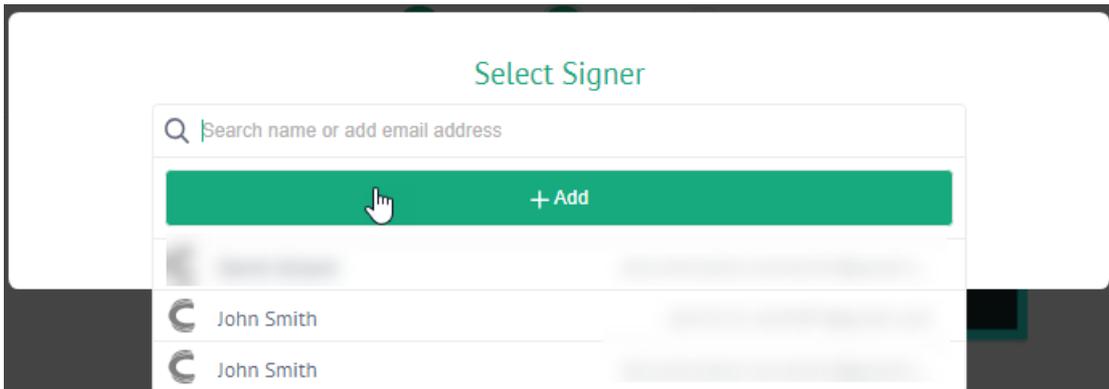
Tip: the blue header indicates where a new document begins.

- Repeat these steps to add a signing field to each document.



Add a signer to the signing field

- Click inside the signing field to add a signer.
- Click inside the **Search name or email address** field.
- If contacts already exist in your system, they're displayed in a list.



- If no contacts appear, click the **Add** button to add a new signer.

Attention: do *not* create more than one contact using the same email address, e.g. for couples where one partner doesn't have an email address. Each contact must have an individual email address.

Create new contact

<input type="text" value="Email"/> *	<input type="text" value="Personal title"/>
<small>Enter a valid email address.</small>	<small>Enter a personal title.</small>
<input type="text" value="First Name"/> *	<input type="text" value="Last Name"/> *
<small>Enter your first name here.</small>	<small>Enter your last name here.</small>
<input type="text" value="DD"/> <input type="text" value="-"/> <input type="text" value="YYYY"/>	<input type="text" value="BE (+32)"/>
<small>Day</small> <small>Month</small> <small>Year</small>	<small>Enter a valid phone number to receive SMS OTP.</small>
<input type="text" value="English"/>	
<small>Choose a preferred language.</small>	

- Enter all required fields (marked by an asterisk) and click **Confirm**.

Technical Notes:

When the contact will sign with eID, BeLawyer or itsme, make sure the info you enter in the contact corresponds to the info on the signing certificate of their Belgian eID card, lawyer card or itsme account. The authenticity of the signer will be checked against their given name(s), last name(s) and birth date when signing with eID. Make sure the name of the contact is identical to the one on the signing certificate of his Belgian eID card. Note however that there is no guarantee that the date of birth is correctly registered on the eID, lawyer card or in the itsme account.

When the **MandatedSignerType** has been set to **matchid** in the Configuration Index, the national security number field must also be filled in. Only numerical characters must be inserted in this field.

- Repeat the steps above to add multiple signers.

Select a signing method for the selected signer

When you've selected a signer, you can determine their signing methods.

Depending on the configuration of your eSignatures solution, you may be able to select multiple signing methods. This way, the signer will have 'choice of signing'. In other words, they can select one of the signing methods that were defined for them.

Important note: eID signing and BeLawyer signing can't be combined when mandated signing based on **MatchId** is activated for both signing types in the Configuration Index.

Select signing method for: Jane Jefferson

 Manually	 eID
 Manually + eID	 SMS OTP
 Email OTP	 iDIN
 Biometric	 BeLawyer
 itsme	

Do you want to add a legal notice?

Signer must sign with the selected signing method

Cancel

Select

The following signing methods are supported in eSignatures. Note that not all signing methods may be available in your eSignatures solution, depending on the configuration.

Tip: if a desired signing method is not available, contact your administrator to enable it in the Configuration Index.

Manually

Manual signing means that a manual signature is needed, as if signing a paper document with a regular pen. Signers need to draw their signature on-screen using a mouse or touchpad, or using their fingers on a touchscreen.

eID

Select eID if you want signers to use their Belgian eID to sign.

A third-party card reader is required. See [Web Portal FAQ > Signing documents > Which smart card readers are supported?](#) for more info.

Note: when using the eID signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

Manually + eID

- This signing method combines manual signing and eID signing. Users first need to draw their signature manually and then enter their eID.

A third-party card reader is required. See [Web Portal FAQ > Signing documents > Which smart card readers are supported?](#) for more info.

Note: when using the eID signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

SMS OTP

Select **SMS OTP** if you want signers to sign using an SMS code. They will need to enter the last four digits of their phone number. In return, they'll receive a one-time password via SMS.

Note: the phone number of the signers must be known in order to use their signing method.

Important: if phone number confirmation is disabled in the Configuration Index, signers won't need to complete their phone number. They will receive the sms code directly.

Email OTP

Select **Email OTP** if you want signers to sign using a one-time password they receive via email. They will need to complete their email address. In return, they'll receive the password via email.

Important: if email confirmation is disabled in the Configuration Index, signers won't need to complete their email address. They will receive the email code directly.

iDIN

iDIN signing allows signers to sign using their Dutch bank card.

The following issuers are currently supported: ABN Amro Bank, ASN Bank, ING Bank, Rabobank, RegioBank, SNS Bank, Triodos Bank.

Biometric

Select **Biometric** if you want signers to sign using one of the following devices:

- A biometric signature pad
- A Bamboo FINELINE Stylus (only on iPad)

Signature pads and the Bamboo FINELINE Stylus allow to capture the biometrical characteristics of a signature, like where the pen is located, when the pen tip is pressed down, and how hard it is pressed down. These data are added to the signature, which would allow the signature pad manufacturer to verify the authenticity of the signature when required.

Important:

eSignatures currently supports the following biometric signature pads: Wacom STU-430, Wacom STU-530 and Wacom STU-540. Make sure the necessary Wacom SDK is installed on the signer's computer: wacom-signature-sdk-x86-3.19.2.msi (32-bit Operating System) or wacom-signature-sdk-x64-3.19.2.msi (64-bit Operating System).

eSignatures currently supports the Wacom Bamboo FINELINE Stylus (CS-600), but only on iPad.

Due to the technical setup of biometric signatures, each document within a package must be signed individually.

BeLawyer

Select **BeLawyer** if you want signers to use their electronic lawyer's card.

A third-party card reader is required. See [Which smart card readers are supported?](#) for more info.

Note: when using the BeLawyer signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

Itsme

Select **itsme** if you want signers to use their itsme app.

Itsme is your digital ID to log in securely, to share your ID data or to sign using your mobile phone.

Prerequisites:

- The signer must have an itsme account and have the itsme app installed on his mobile phone in order to sign.

Important notes:

- When using itsme, the output format of your documents (which you selected at [Step 1: upload documents](#)) must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.

- When using itsme signing in packages, each document within the package must be signed individually.

Limitation:

- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Custom

As of eSignatures 5.1, a custom signing method - based on OpenID Connect - may have been configured that allows users to sign by pincode.

The name and icon of the signing type depends on the configuration done in the Configuration Index.

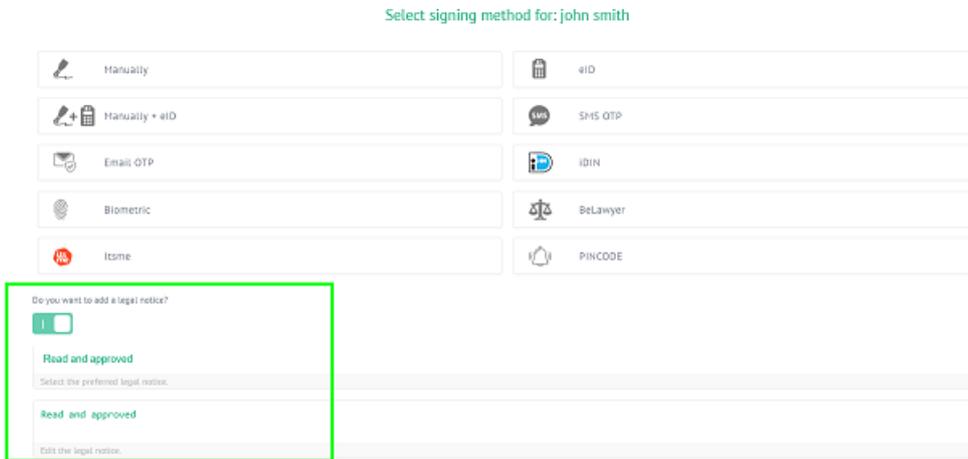
Technical note:

When uploading documents via the eSignatures API, you can also use **Server signing**. Server signing signs documents automatically on the server once they have been uploaded. Since Server signing happens automatically as soon as a document is uploaded, it cannot be combined with choice of signing. Choice of signing requires user interaction, whereas Server signing is an automatic feature.

Legal notice

Decide whether the signer must enter a legal notice.

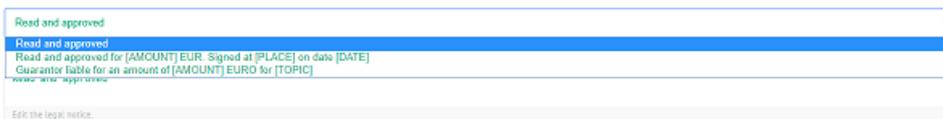
If you enabled the legal notice setting at **Step 1: upload documents**, the legal notice setting you configured here is enabled for each signer you add.



You can now choose to customize the legal notice per signer, or disable it if certain signers do not need to enter a legal notice.

- Three predefined legal notices are available in the drop-down list.

The language of the legal notice corresponds to the document language you selected during the upload.



- Modify the content of the legal notices to your liking.

For instance enter variables between square brackets. When you do so, the signer needs to enter a value for each variable when signing the document.

Important notes:

When you add a legal notice, the signers need to retype the exact content of their legal notice before they're able to sign the document.

When you add a legal notice to a package that contains multiple documents, each signature field will need to be signed individually.

Limit the legal notice to 255 characters including any predefined content. Otherwise an error will occur during signing.

When using biometric signing: the Wacom signature pads display only a limited number of characters on the signing screen. The STU-430 approximately displays 24 characters over 1 line of text, while the STU-530 displays approximately 74 characters over 2 lines. If the legal notice contains more characters, they fall off the screen. You're recommended to run some tests beforehand to see if the text fits.

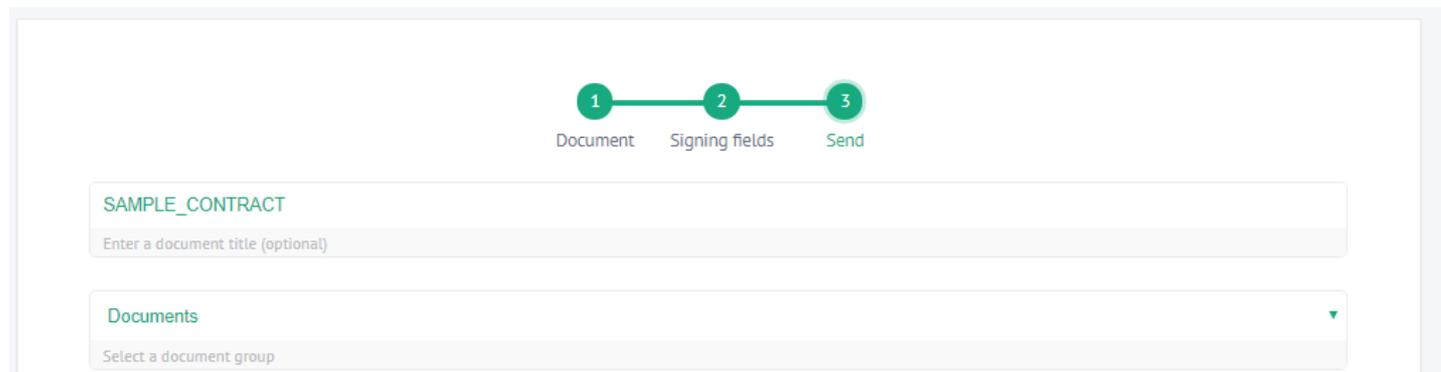
When you're done, click the green arrow to go to the next step.

Step 3: send the documents

Sending the documents is the final step in the upload process. All the settings you can configure in this screen are optional.

Enter a title

If necessary, you can still modify the title here.



Select the document group

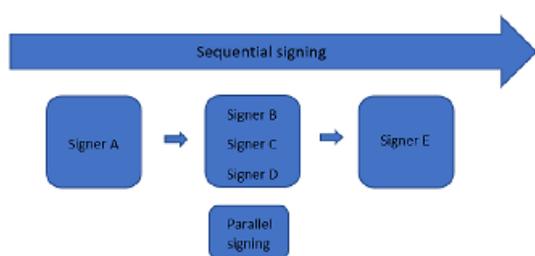
If necessary, you can still modify the document group to which the documents must be uploaded.

Arrange the signers

When you uploaded a document or package, and assigned multiple signers, you can decide whether they must **sign in parallel**, **sign sequentially** or design a **complex** signing flow.

When the order in which the signers sign the document has no importance, choose **Sign in parallel**. When you want to define a fixed signing order, choose **Sign sequentially**. In this case a signer won't be able to sign the document if his preceding signer hasn't signed the document yet.

When you want to combine parallel and sequential signing, choose **Sign complex**. An example of complex signing would look as follows : User A must sign the package first. Once he has signed, users B, C and D must sign the package. The order in which they sign doesn't matter, as long as all three users sign. When all three users have signed, User E needs to sign to give his final approval. In this example Users B, C and D sign in parallel, while the overall signing flow is sequential.



To have signers sign sequentially:

- Click **Sign sequentially**.
- Drag and drop the signers to the required position.

Signers



For each signer you have an overview of the documents they need to sign and the signing method they must use.

Signers

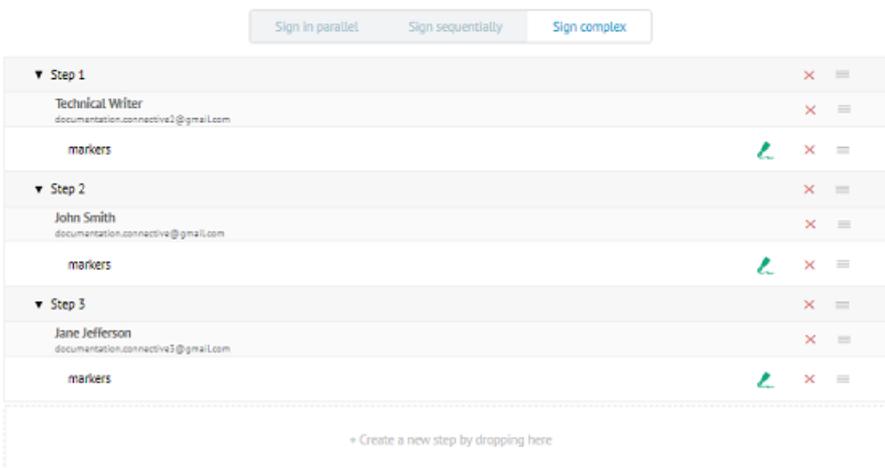
John Smith		✕
SAMPLE_CONTRACT	👤	✕
Jane Jefferson		✕
SAMPLE_CONTRACT	🔒	✕
Addendum	🔒	✕

Review the signers in your package

To configure complex signing:

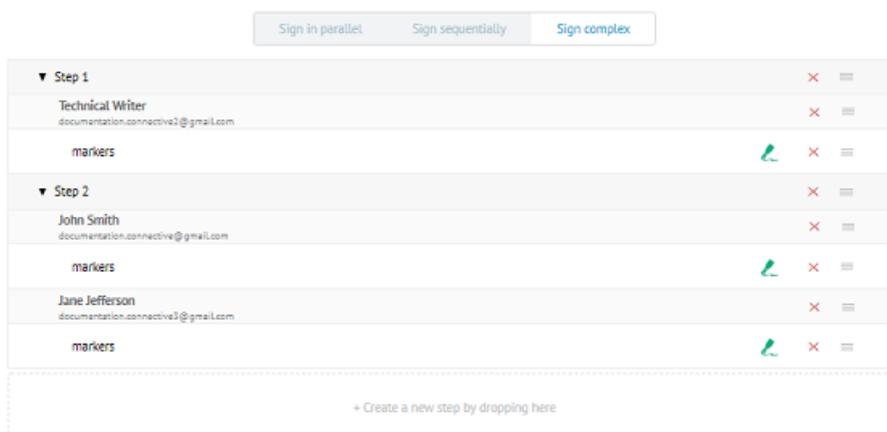
- Click **Sign complex**.
- Drag the signers to the required position. The steps indicate the order in which the signers must sign.

Signers



- The signers who may sign in parallel must be placed under the same step, as shown in the image below.

Signers



Tip: to create an additional step, drag a signer to the dotted frame.

Add receivers

If someone needs to receive a document link to the fully signed document, add that person as receiver. Note that a receiver does *not* sign documents in eSignatures.

- Click inside the search field to add a receiver.
- If contacts already exist in your system, they're displayed in a list. Select the required receiver from the list.



- If no contacts appear, click the **Add** button to create a new contact and add them as receiver.
- Enter all required fields and click **Confirm**. The required fields are marked by an asterisk.
- Repeat the steps above to add additional receivers.

Set an expiry date

To make the document become unavailable to the signers after a certain time, set an expiry date.

- Click the expiry date button and then select the date.

To make the document available to the signer again, the initiator will have to [extend the expiry date](#) in the Document Portal.

Customize the email message to the signers

To personalize the email message, click "**Click here to change the email message to the signers**".

Save draft

To save a draft of your document, click **Save draft**.

You now return to the Document Portal overview where you can still edit the document and send it later on.

Send the document

Click **Confirm** when you're ready to send the document.

The following message is displayed and the signers will receive a download link to the document.



Important: as initiator you can also have signers **sign immediately**. This is useful in case of "face to face signing" when all signers are physically present. See [How do I use face to face signing](#) for more info.

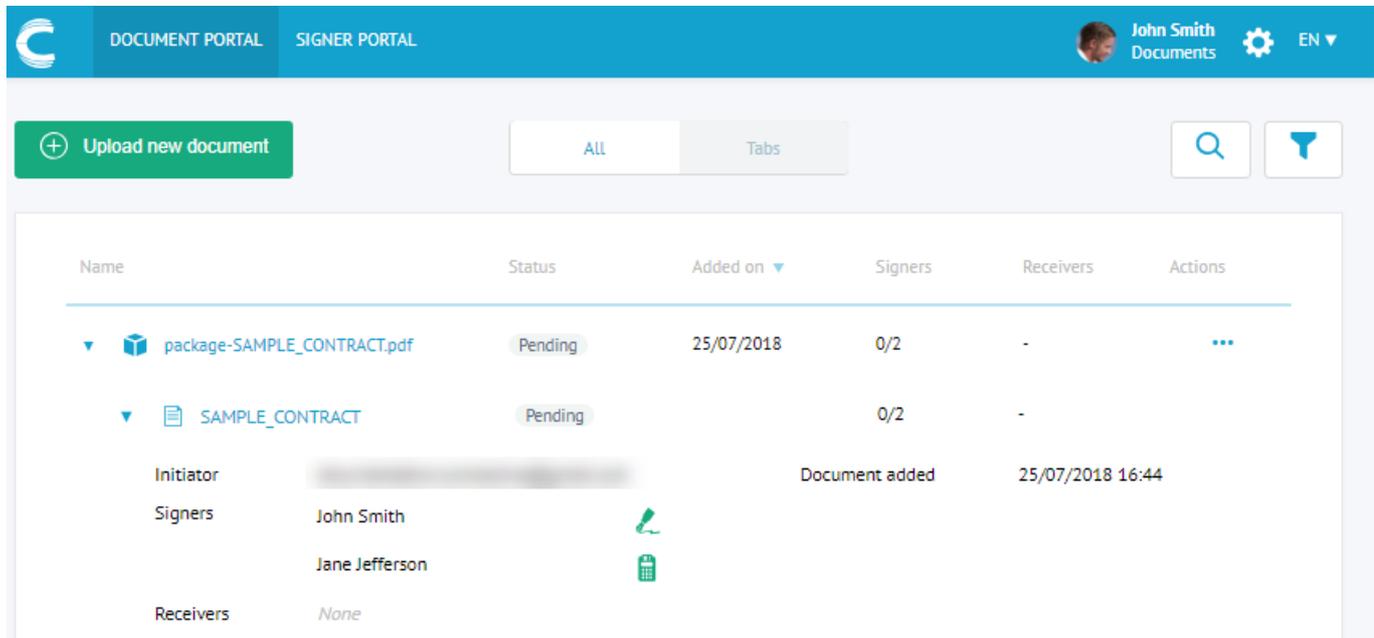
2.4.4 How do I view my uploaded documents?

All the documents you've uploaded are displayed in the Document Portal.

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.

All view

The **All** view lists all the documents that have been uploaded. You can see their status, the date on which they've been added, who has signed them, who still needs to sign them and the receivers they've been sent to.

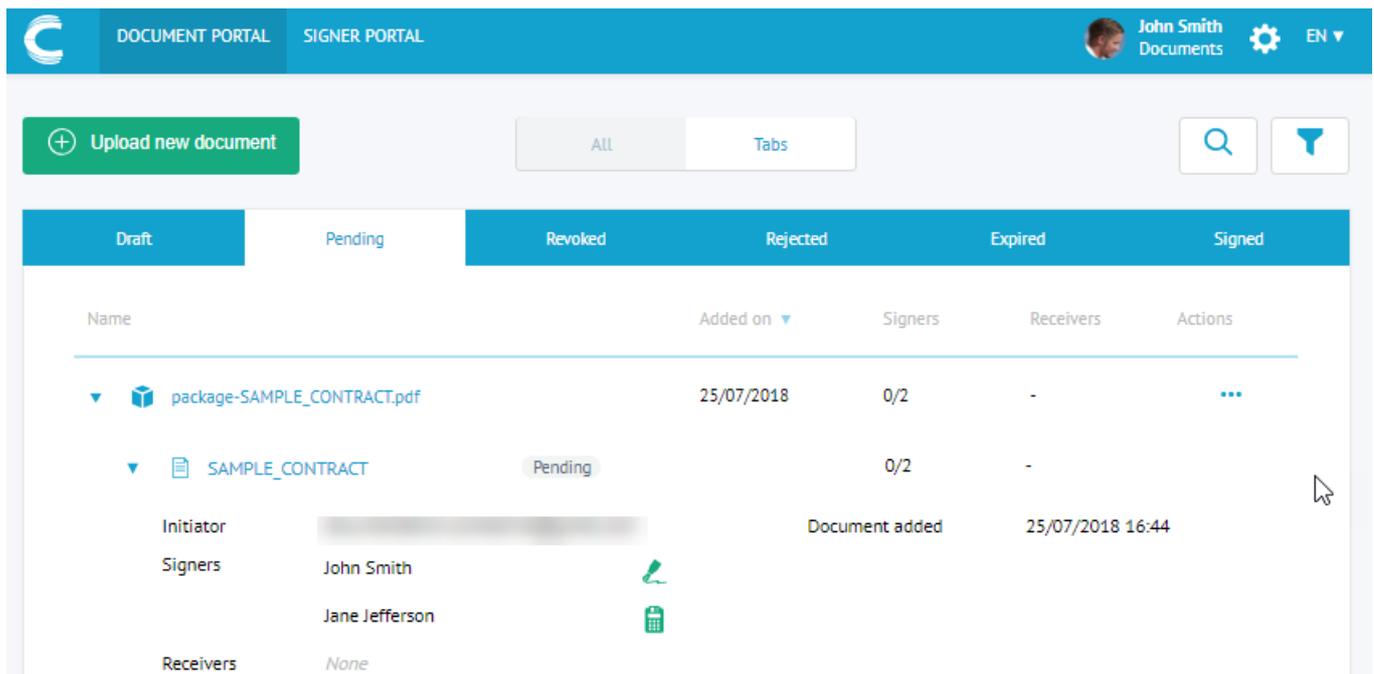


The screenshot shows the 'All' view of the Document Portal. The header includes 'DOCUMENT PORTAL' and 'SIGNER PORTAL' tabs, a user profile for 'John Smith Documents', and a language selector 'EN'. A green button 'Upload new document' is on the left. Below it are 'All' and 'Tabs' view tabs, a search icon, and a filter icon. The main content is a table with columns: Name, Status, Added on, Signers, Receivers, and Actions. The table lists a document 'package-SAMPLE_CONTRACT.pdf' with status 'Pending', added on '25/07/2018', and 0/2 signers. Below the document name, there is a sub-table for details: Initiator (blurred), Signers (John Smith, Jane Jefferson), and Receivers (None). The 'Actions' column contains a three-dot menu icon.

To see which actions can be done on a document, click the menu icon **...** in the **Actions** column.

Tabs view

In **Tabs** view, the documents are sorted per **status**.



The screenshot shows the 'Tabs' view of the Document Portal. The header is the same as in the 'All' view. The 'Upload new document' button is on the left. Below it are 'All' and 'Tabs' view tabs, a search icon, and a filter icon. The main content is a table with tabs for document status: Draft, Pending, Revoked, Rejected, Expired, and Signed. The 'Pending' tab is selected. Below the tabs is a table with columns: Name, Added on, Signers, Receivers, and Actions. The table lists the same document 'package-SAMPLE_CONTRACT.pdf' with status 'Pending', added on '25/07/2018', and 0/2 signers. Below the document name, there is a sub-table for details: Initiator (blurred), Signers (John Smith, Jane Jefferson), and Receivers (None). The 'Actions' column contains a three-dot menu icon.

A document can have one of the following statuses:

Draft: the document has been saved but may not have been finalized yet. The document can be edited and must be finalized

before you're able to send it.

Pending: the document has been sent and is awaiting signing.

Revoked: the document had been sent but was canceled by the initiator. The signer can no longer sign the document.

Rejected: the document has been sent to the signers but one or more signers rejected to sign it. In this case the document is no longer available to the other signers.

Expired: the document has been sent to the signer but he did not sign it before the expiration date.

Signed: the document has been sent to the signer and he signed it.

Failed: the signing has failed. Contact your system administrator to see what went wrong.

Display the details of a document

To display the details about each document, click the expand arrow ► in the **Name** column. Depending on the state of the document different details are displayed.

The following details are available for the different states:

Draft

- Initiator (who sent the document)
- Date when the document was added
- Signers (if already defined)
- Signing method (if already defined)
- Receivers (if applicable)

Pending

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer has to use
- Date when the document will expire (if applicable)
- Receivers (if applicable)

Revoked

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer has to use
- Date when the document was revoked
- Receivers

Rejected

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer has to use
- Receivers
- Signer who rejected the document
- Date when the document was rejected
- Reason for rejection

Expired

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer has to use
- Date on which the document expired
- Receivers

Signed

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer used
- Date when the document was signed
- Receivers

Failed

- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing method the signer used
- Receivers

Search for a document

To search for a specific document:

- Click the search icon  to search for a specific document.
- Enter the document name or signer name and click **Search** to display the results.

Filter the document list

To filter the document list:

- Click the filter icon  to filter the document by period.
- Enter the **From** and the **To** date and click **Filter** to display the results.

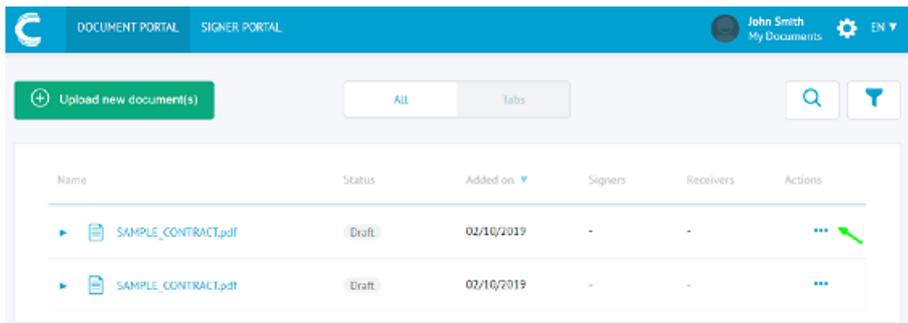
2.4.5 How do I edit a draft document?

When you saved an uploaded document as draft, or when a document was automatically saved as draft after you quit the upload flow, the document can still be edited and finished in the Document Portal.

Important: you can only edit a document of which you are the initiator. Editing a draft that was uploaded by another user is not possible, not even if you have viewing rights to his document group.

To edit a draft document:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document you want to edit, and click the menu button in the **Actions** column.



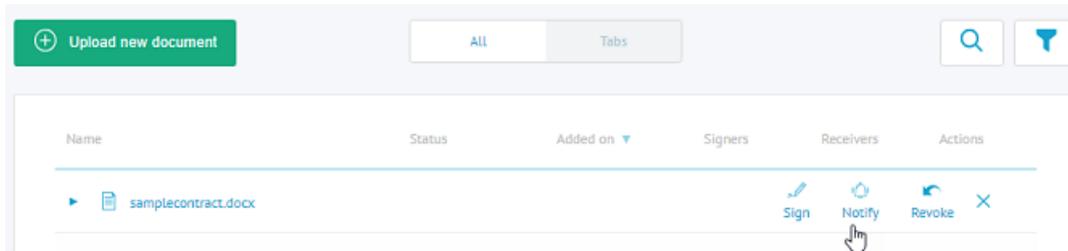
- Click the **Edit** button . Note that the **Edit** button is only available for documents that have the status 'draft'.
- The document is opened at step 1 of the upload flow. You can now go through all the [upload steps](#) again.

2.4.6 How do I send a notification to the signer?

As initiator you can send a notification to the signers, reminding them to sign the document.

To send a notification:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document that is pending signing, and click the menu button **☰** in the **Actions** column.
- Click **Notify**.



- A pop-up message appears when the notification has been sent.
- The signers will receive a reminder email in their mailbox.

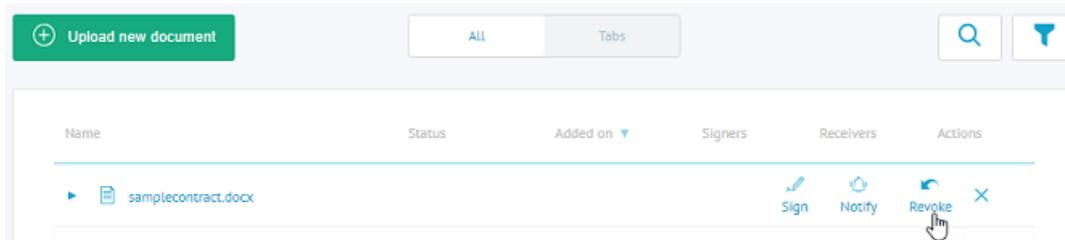
2.4.7 How do I revoke a document?

As initiator you can revoke documents that were already sent to signers, but haven't been signed yet, or documents of which the signing has failed. Once a document has been revoked, signers will receive an email and they'll no longer be able to sign their documents.

Important: revoking a document is irreversible.

To revoke a document:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document you want to revoke, and click the menu button **☰** in the **Actions** column.
- Click **Revoke**.



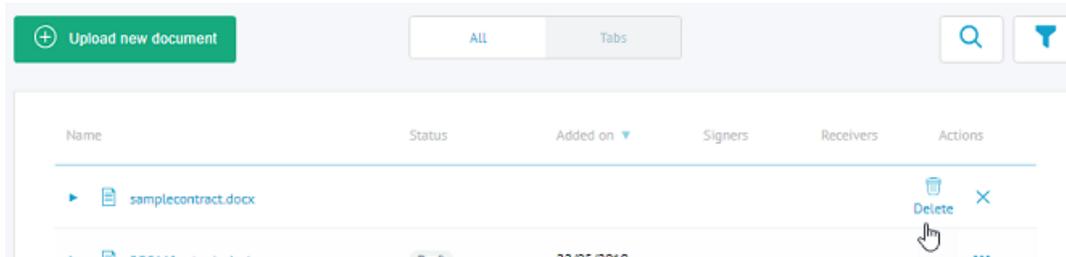
- Then click **Confirm** in the pop-up window.

2.4.8 How do I delete a document?

Initiators can delete documents from the Document Portal. A document can be deleted when it's in draft, signed, rejected or revoked status.

To delete a document:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document you want to delete, and click the menu button **⋮** in the **Actions** column.
- Click **Delete**.



- Then click **Confirm** in the pop-up window.

Are you sure you want to **delete** this package?

Document name	samplecontract.docx
Date added	01/06/2018

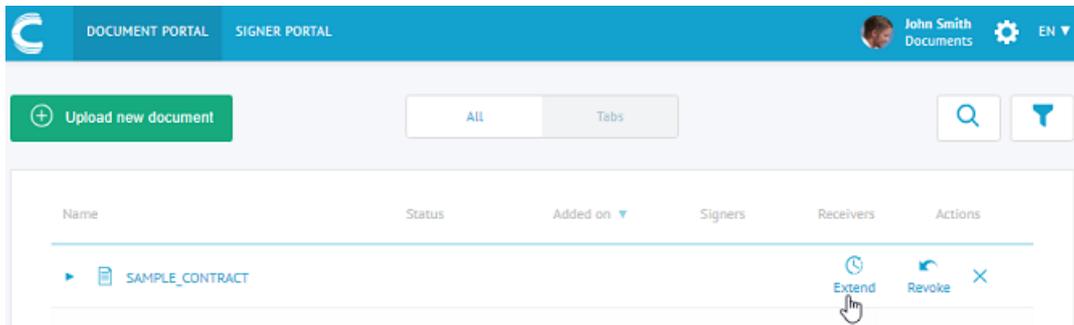
Note: documents or packages are never deleted automatically. They are stored indefinitely if they are not deleted manually.

2.4.9 How do I extend the expiration date?

When a document has expired - meaning when the signer did not sign it before the expiration date - you can extend the expiration date as initiator.

To extend the expiration date:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document of which you want to extend the expiration date, and click the menu button **☰** in the **Actions** column.
- Click **Extend**.



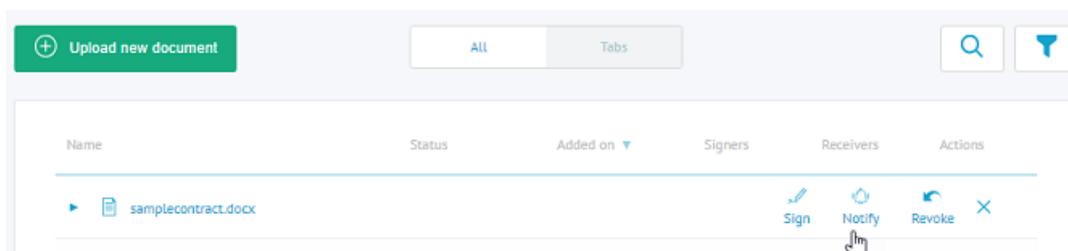
- Set the new expiration date and time, and click **Confirm**.

Extend expiry period

26/07/2018  Pick a date.

12  15 
Hour Minutes

- To send a reminder email to the signer, click the menu button and click the **Notify** button.

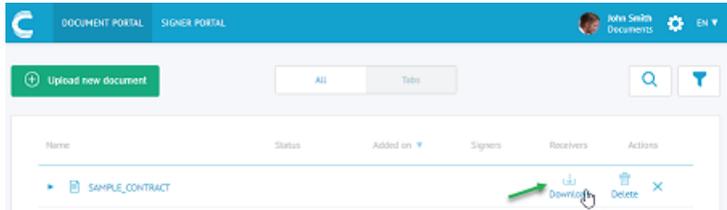


2.4.10 How do I download a document?

You can download a document from the Portal once it's been signed. Signers can also [download their signed documents from the Signer Portal](#) and from their email.

To download a document:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab. Signers should click the **Signer Portal** tab.
- Go to the document you want to download, and click the menu button in the **Actions** column.
- Click the **Download** button.



- The document is downloaded in your browser in the bottom left corner or top of the screen.
- Click the document to open it.

Note: a package is always downloaded as .zip file. You'll need to unzip it to view its contents.

2.5 Signing documents

2.5.1 How do I sign a document?

2.5.2 How do I QuickSign a package?

2.5.3 How do I bulk sign?

2.5.4 How do I sign face to face?

2.5.5 How do I reject a document?

2.5.6 How do I download a signed document?

2.5.1 How do I sign a document?

You can sign documents in various ways:

- Via **email**
- In the **Signer Portal**
- In the **Document Portal**
- In a **mobile app**

Signing via email

Even if you don't have an eSignatures account you can still sign documents generated by eSignatures. Simply open the email that was sent to you, and click the secure link inside. The documents that require signing will open in your default web browser.

See the [FAQ](#) for more information about the supported [browsers](#) and [operating systems](#).

Signing in the Signer Portal

If you have an eSignatures account, you [log in](#) to your account, go to the Signer Portal and sign your documents.

Signing in the Document Portal

As initiator, i.e. the person who uploads and sends documents for signing, you can have your signers [sign face-to-face](#) in the Document Portal.

Note that you can't access the Document Portal from a smartphone. It can be accessed from a tablet, however.

Signing in a mobile app

If a rebranded app for iOS or Android has been made available for your eSignatures solution, you can download the app from the App store or Google Play and sign documents inside the app.

Important: the standard Connective app available in the App store and Google Play only works with the **standard demo environment** of eSignatures.

You can find the different signing steps in [Signing a document step-by-step](#).

Signing via email

- Open the email you received from your eSignatures solution. In our standard demo environment you receive an email from **esigner@connective.be**.
- Click the secure link inside the email. The document will open in a new tab in your default web browser.

Important: this link will only work once. Once you've clicked it, you need to sign or reject the document. If the link expired, click **Request a new email** to receive a new link.

Please sign your document or package with name SAMPLE_CONTRACT Inbox x



esigner@connective.be

to me ▾

Dear John Smith,

Please click on the link below to sign your document or package (Name: SAMPLE_CONTRACT):

https://www.esignatures.eu/signinit?packageSignId=7a6bdd06-59b7-4c8b-8bd5-0a2c36b2de72&token=dcV7g_Gn1Bqui1Dc5YJpySdyADo3TdYIU4TLwANbv0mhwBJoBrpg286Qc55m2j4

Kind Regards,
John Smith for Connective

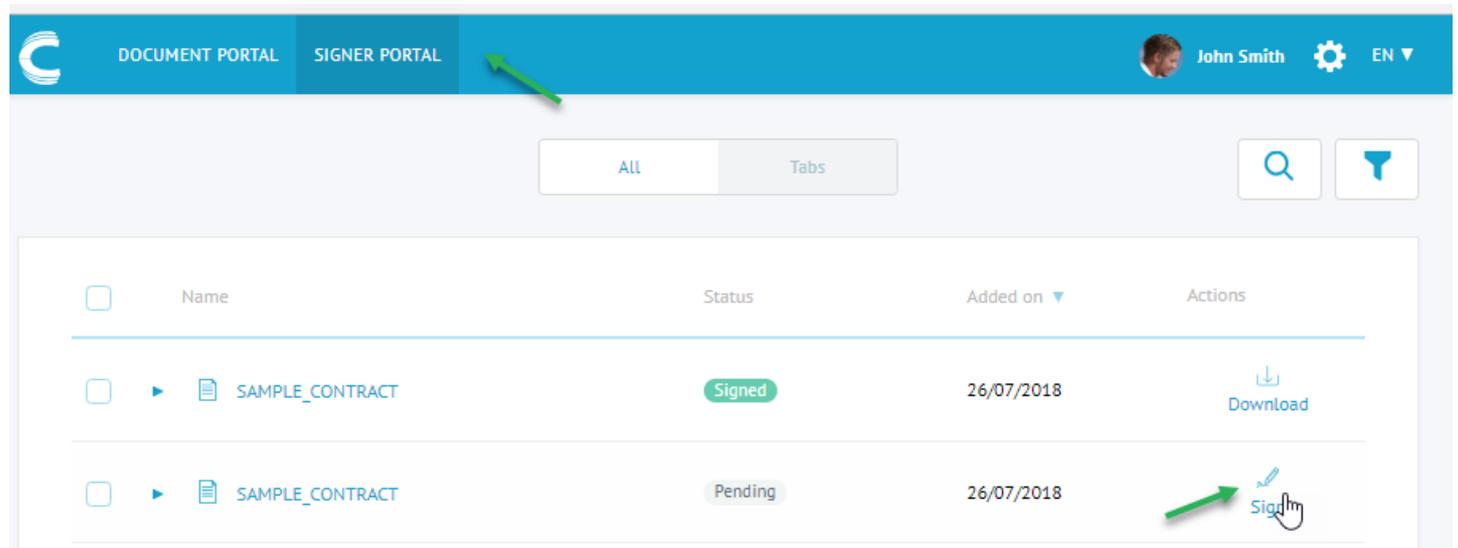


- You can find the different signing steps in [Signing a document step-by-step](#).

Signing in the Signer Portal

- [Log in](#) to your eSignatures account.
- Click the **Signer Portal** tab.
- Click the sign icon for the package you want to sign.

Tip: to sign multiple packages at once, use [Bulk sign](#).



- You can find the different signing steps in [Signing a document step-by-step](#).

Document details

To display the document details, click the expand arrow ► in front of each document to display its details.

To display the document details inside a package, first click an expand arrow ► in the **Name** column. The different documents inside the package are now displayed. Now click the expand arrow in front of each document to display its details.

Depending on the state of the package, different document details are displayed. The following details are available for the different states:

Pending

- Initiator (who sent the document)
- Date and time when the document was added

In progress

- Initiator (who sent the document)
- Date and time when the document was added

Rejected

- Initiator (who sent the document)
- Date and time when the document was added
- Date and time when the document was rejected
- Reason for rejection

Signed

- Initiator (who sent the document)
- Date and time when the document was added

Failed

- Initiator (who sent the document)
- Date and time when the document was added

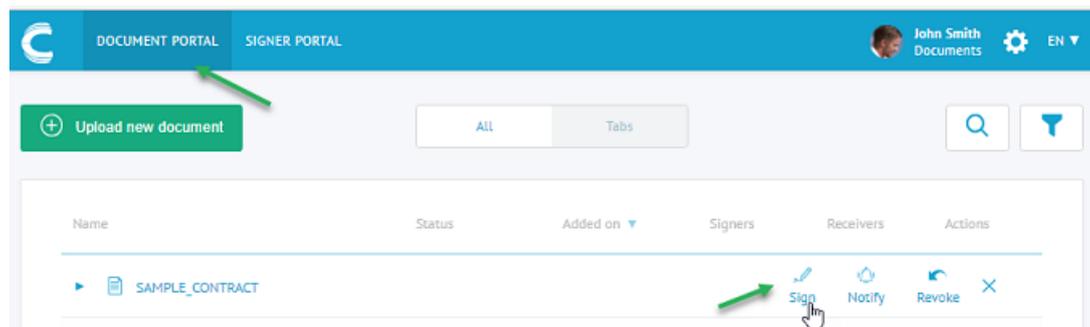
Tip: also check the [video tutorial](#):

Signing in the Document Portal (face-to-face signing)

Signing documents in the Document Portal - also known as face-to-face signing - is only meant for **initiators** when their signers are physically present.

Instead of signers having to check their email, log in to their account or open an app, they can sign their documents immediately on the initiator's screen.

- [Log in](#) to your eSignatures account as initiator.
- Click the **Document Portal** tab.
- Go to the document you want to sign, and click the menu icon **☰** in the **Actions** column. Then click the sign icon.



- In the **Start signing** window:
 - Click **Sign all** if all signing parties are present.
 - Or click **Sign** next to the required signer, if only that signer is present.

Attention: if one of the other signers is simultaneously signing the document using the email link they received, or using the Signing Portal, a message will appear informing you that someone else is signing the document. You will have to wait until that signer has finished.



- You can find the different signing steps in [Signing a document step-by-step](#).

Tip: also check out this [video tutorial](#) on face to face signing:

Signing in a mobile app

- Download your company-specific app from the App store or Google Play, or contact your system administrator.
- Log in to the app.
- Click the sign icon next to the document you want to sign.

See the separate chapter [Signing in the Connective app](#) in the [End users](#) part of this documentation for general instructions on how to use the standard demo app.

Signing step-by-step (WebPortal users)

In this section you'll learn how to sign step-by-step.

Note that it has no importance whether you're signing single documents or packages that contain multiple documents. The same signing steps apply.

eSignatures now supports **asynchronous signing**. This means you no longer need to wait until all documents are signed before you may close your signing session. This comes in handy especially when signing high volumes of (large) documents. As soon as you've started the actual signing, you may close the signing session and continue working on other things.

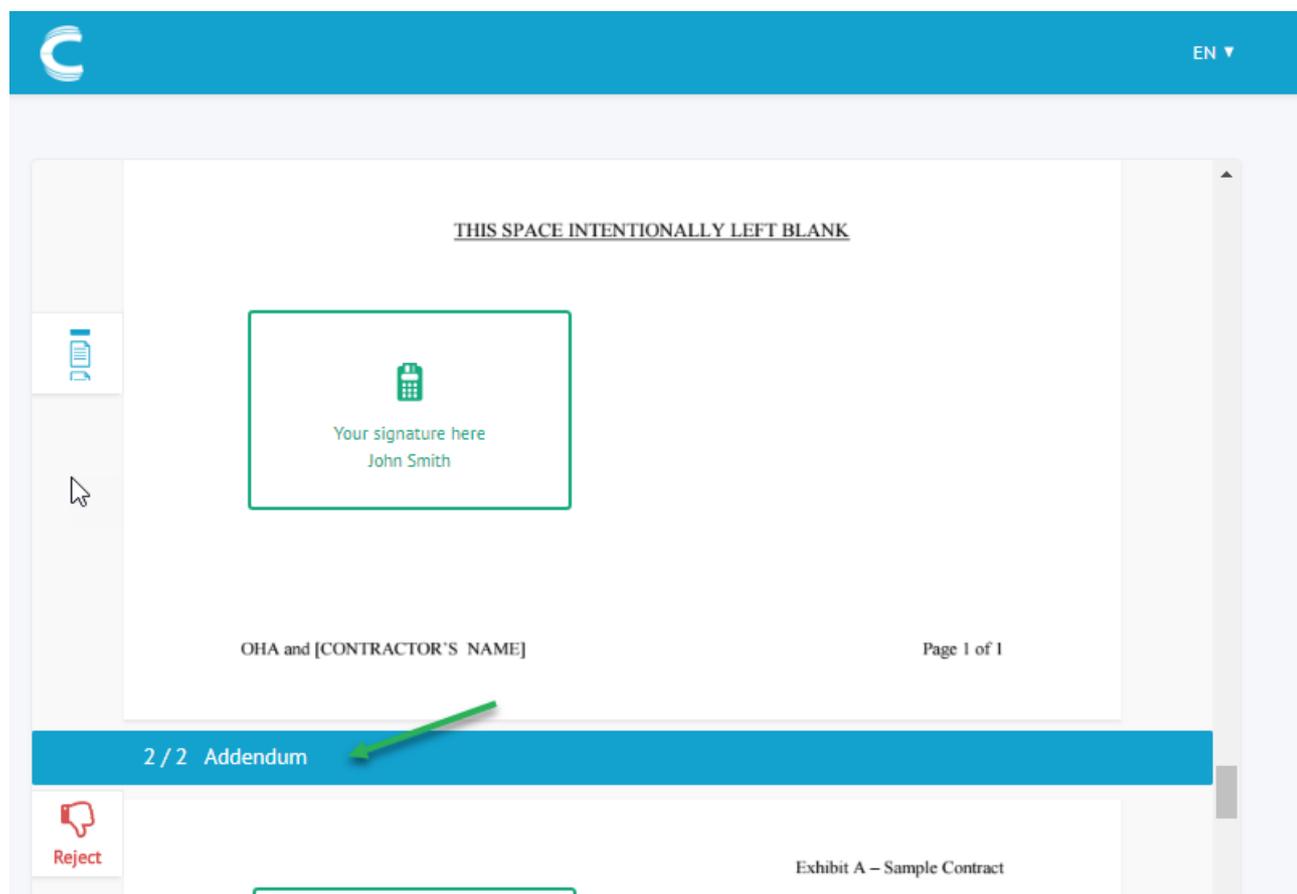
Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (through OpenID Connect) and Biometric.

Attention: when documents have been sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.

Signing step-by-step

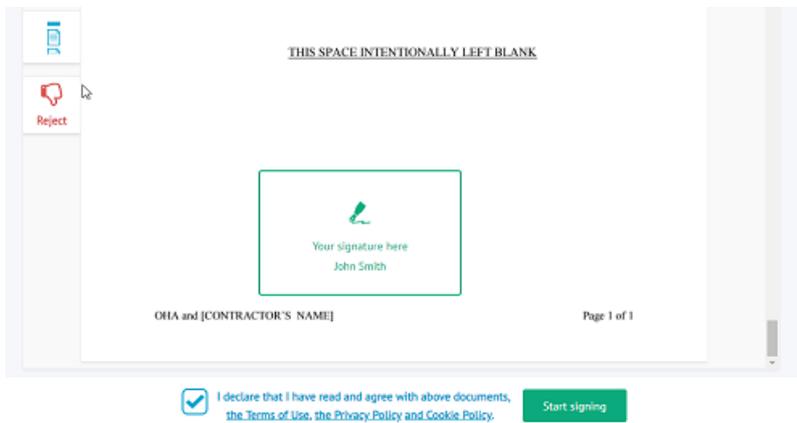
- [Access the documents](#) that require your signature.
- Read the entire document and scroll down to the last page.

If your package contains multiple documents, the start of each document is indicated by a header. The header also indicates how many documents the package contains, and which document you are viewing.



- When you've scrolled down to the last page, the checkbox **I declare that I have read and agree with above documents, the Terms of Use, the Privacy Policy and Cookie Policy** now becomes available. Check it and then click **Start signing**.

Tip: click the links to consult the Terms of Use, Privacy Policy and Cookie Policy respectively.



Install the Connective browser package

Depending on the signing method that was defined for you, you might be prompted to install the Connective browser package. The Connective browser package is required on Windows and macOS when using any signing method that requires additional hardware:

- **eID signing:** requires an eID card reader
- **BeLawyer signing:** requires a transparent card reader
- **Biometric signing:** requires a biometric signing pad

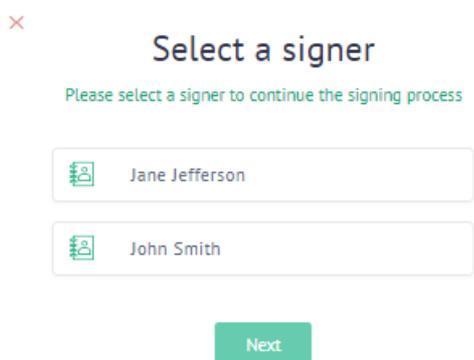
To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.



Select the signer

Note: this step only applies if multiple signers need to sign the document, and you're having them sign in the Document Portal, by means of the **Sign all** button.

- Select the signer you want to sign first.



- Click **Next** and continue the signing process. Once the signing process is finished for the first signer, click **Start signing** again to go through the signing process for the other signers.

Choice of signing

- If multiple signing methods have been defined, you are prompted to **select a signing method**. Select your preferred signing method, and then click **Next**.

For more information about the signing methods, see [What are the different signing methods?](#)

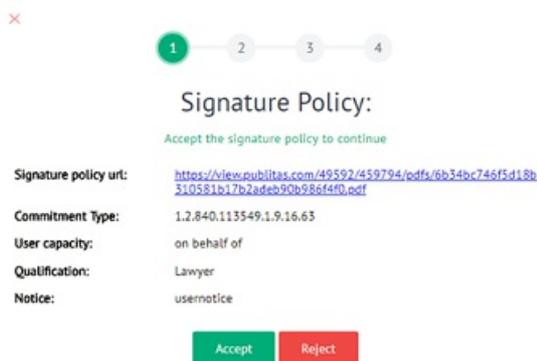


Signature policy

- If your document contains a **signature policy**, you are prompted to accept it.

Technical note: a signature policy is a set of rules that details the terms and conditions of how a valid signature should be created and validated. A signature policy is specified by means of a single ID that must be configured in the eSignatures database. Once the signature policy is configured in the eSignatures database it can be used in eSignatures API calls. See **Appendix IV** of the **Connective - eSignatures 5.2.7 - Configuration Documentation** to learn how to use signature policies. Note that signature policies can only be entered in the eSignatures API, not in the WebPortal.

- To check the signature policy before accepting it, click the link in the **Signature policy** screen.
- If you agree with the signature policy, click **Accept**. If you reject, you won't be able to sign the document.

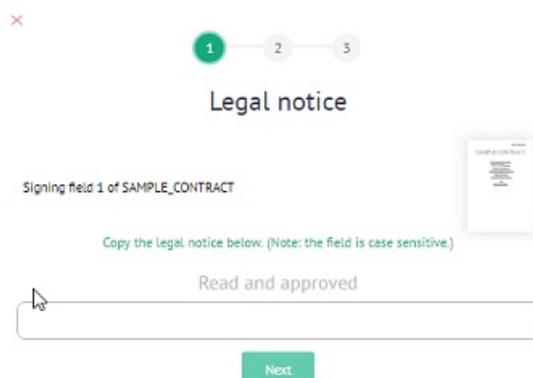


Legal notice

- If a legal notice was defined for your document, the Legal notice window opens.
- Type the content of the legal notice in the empty field. Copy-Paste is not supported.

Important: the legal notice you enter in the field must be exactly the same as the original legal notice, including spaces, cases, punctuation marks.

- Click **Next**.



Signing vs QuickSigning

- The signing window now opens, based on the signing method that was defined for you.
- If QuickSigning has been enabled, you'll be able to sign all signing fields inside the package using a single signature.



Quicksign manually

Sign in the field below.

Retry Next

- If QuickSigning is not enabled, or the conditions are not met, you'll need to sign each signing field that is assigned to you.



Sign manually

Signing field 1 of marker



Sign in the field below.

Retry Next

Conditions for QuickSigning

- The following two signing methods cannot be combined with QuickSigning: sign manually+eID and biometric. If one of these signing methods has been selected for you, each field within the package must be signed individually.
- When using eID signing, a transparent eID card reader is required, i.e. a card reader without PIN pad. If you're using a PIN pad eID reader you'll be prompted to sign each field individually.

- If your package contains a legal notice, each field within the package must be signed individually.

Signing methods

One of the following signing methods may have been assigned to you. Click the links below for more information about each signing method.

Important: not all signing methods listed below may be available in your eSignatures solution, depending on the configuration.

Sign manually

- Use the mouse cursor to draw your signature in the signature field. To sign manually, a manual signature is needed as if signing a paper document.
- If you're not satisfied with the signature, click **Retry** to start over.
- When you are done, click **Next**.

×



Quicksign manually

Sign in the field below.

Retry Next

- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.

×



Adding signatures to documents ...

We are adding signatures to your documents, you can close this window at any time

Close

- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign with eID

- Connect a [supported card reader](#) and insert your eID. The application now does all necessary checks.

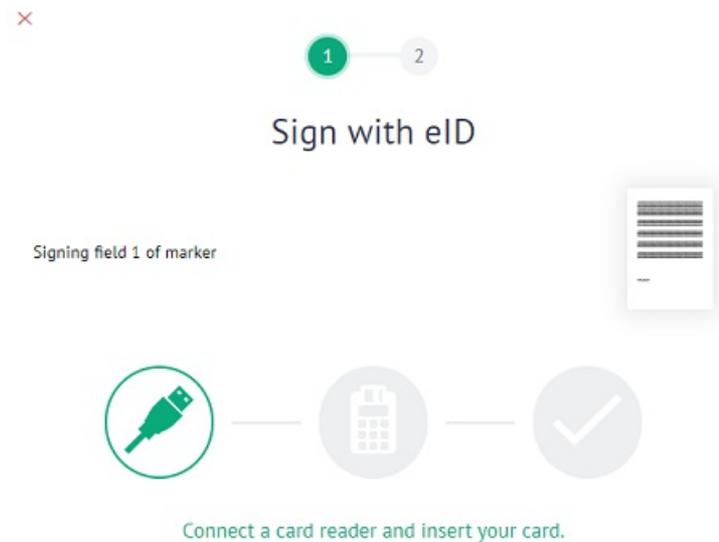
Important notes:

Make sure the Connective browser package has been properly installed. Otherwise eID signing will not work. To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

When using Internet Explorer, do *not* use Compatibility View. This is not supported by the Connective browser package.

To QuickSign using eID you need a transparent eID card reader, i.e. a card reader without PIN pad.

To use eID signing on an iOS or Android device, you need a rebranded [app](#) and use the Vasco 875 Bluetooth eID card reader. eID signing is not supported in a mobile web browser.



- When asked, enter your PIN. If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message appears on-screen.
- Click **Finish** to complete the process.

Sign manually + Sign with eID

- Use the mouse cursor to draw your signature in the signature field.
Tip: if you're not satisfied with the signature, click **Retry** to start over.
- When you're done, click **Next**.



Sign manually

Signing field 1 of marker



Sign in the field below.

Retry Next

- Connect a [supported card reader](#) and insert your eID. The application now does all necessary checks.



Sign with eID

Signing field 1 of marker



Connect a card reader and insert your card.

Important notes:

Make sure the Connective browser package has been properly installed. Otherwise eID signing will not work. To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

When using Internet Explorer, do *not* use Compatibility View. This is not supported by the Connective browser package.

To QuickSign using eID you need a transparent eID card reader, i.e. a card reader without PIN pad.

To use eID signing on an iOS or Android device, you need a rebranded [app](#) and use the Vasco 875 Bluetooth eID card reader. eID signing is not supported in a mobile web browser.

- When asked, enter your PIN. If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message

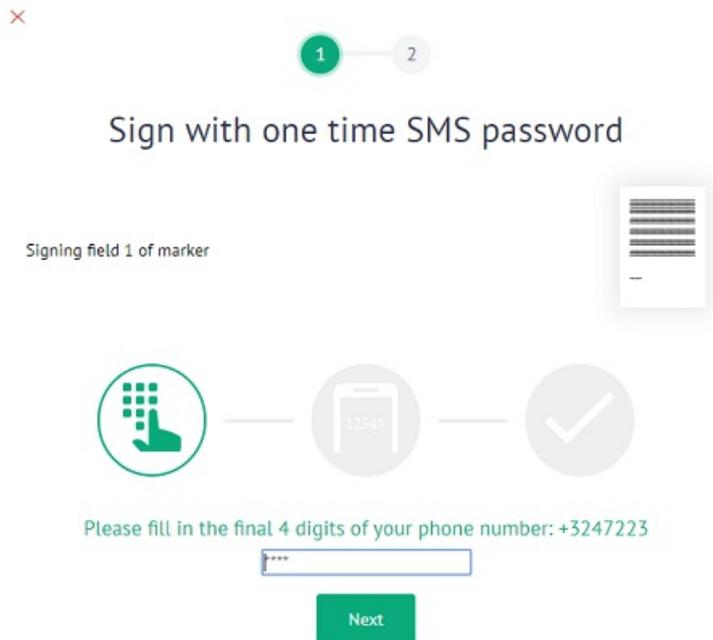
appears on-screen.

- Click **Finish** to complete the process.

Sign with SMS OTP

- Enter the 4 last digits of your phone number.

Important: if phone number confirmation is disabled in the Configuration Index, you won't need to confirm your phone number. You will receive the sms code directly.



- Click **Next**. The password is sent by SMS to your phone.
- Enter the code you received and click **Next**.

Notes:

Make sure to enter the correct code. By default the number of attempts is limited to 10, but can be modified by an administrator in the Configuration Index.

The SMS code is valid for a limited amount of time. By default, this is 5 minutes, but can also be modified.



- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



Adding signatures to documents ...



We are adding signatures to your documents, you can close this window at any time

Close

- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Tip: also check the [video tutorial](#) on sms signing:

Sign with email OTP

- Enter your full email address. A hint about the expected email address is shown between parentheses.

Important: if email address confirmation is disabled in the Configuration Index, you won't need to confirm your email address. You will receive the email code directly.



Sign with one time password via email

Signing field 1 of marker



Provide your **full email address** (documentatio*****@gmail.com).

Next

- Click **Next**. If you entered the correct email address, an email is sent to the email address you entered.
- Enter the code you received in the confirmation field.



Quicksign via email OTP



Enter the confirmation code to sign the document.

Generate new OTP code. Next

Notes:

Make sure to enter the correct code. By default the number of attempts is limited to 10, but can be modified by an administrator in the Configuration Index.

The SMS code is valid for a limited amount of time. By default, this is 5 minutes, but can also be modified.

- Click **Next**.
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



Adding signatures to documents ...



We are adding signatures to your documents, you can close this window at any time

Close

- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign via signing pad (biometric)

- Connect your signature pad to your computer using the provided USB cable.

Signing via signing pad requires a biometric signature pad. Such a signature pad allows to capture biometrical characteristics of your signature, like where the pen is located, when the pen tip is pressed down, and how hard it is pressed down. These data are added to the signature, which would allow the signature pad manufacturer to verify the authenticity of the signature when required.

Important: eSignatures currently supports the following biometric signature pads: Wacom STU-430, Wacom STU-530 and Wacom STU-540. Make sure the necessary Wacom SDK is installed on your computer: wacom-signature-sdk-x86-3.19.2.msi (32-bit Operating System) or wacom-signature-sdk-x64-3.19.2.msi (64-bit Operating System).

Important: due to the technical setup of biometric signatures, each document within a package must be signed individually.



Sign via signing pad

Signing field 1 of SAMPLE_CONTRACT



Connect a signing pad.



- When the device is successfully connected, "Please sign using your signature pad" appears on-screen.
- Draw your signature on the signature pad using the provided stylus.

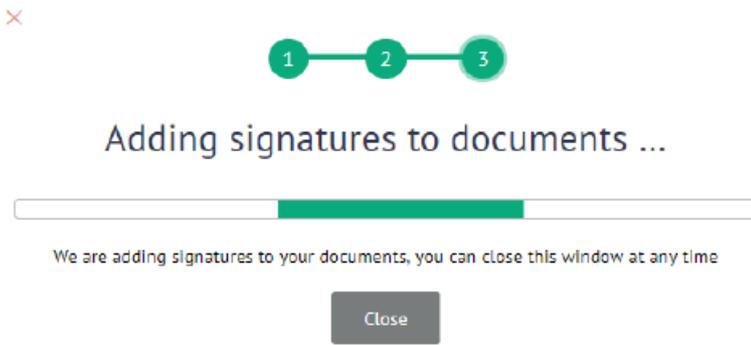
To start over, tap **Clear**.

To cancel the operation, tap **Cancel**.



- When you're done, tap **OK** on the signature pad screen using the stylus.

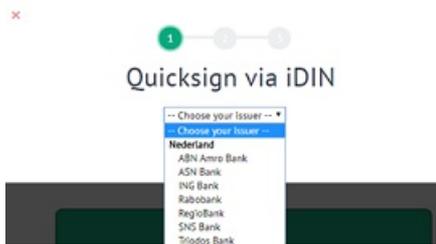
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign with iDIN

iDIN signing allows users to sign using their Dutch bank card. The following issuers are currently supported: ABN Amro Bank, ASN Bank, ING Bank, Rabobank, RegioBank, SNS Bank, Triodos Bank.



Sign with BeLawyer card

- Connect a [supported card reader](#) and insert your Belgian lawyer card. The application now does all necessary checks.

Important: make sure the Connective browser package (Windows and macOS) has been properly installed. Otherwise signing will not work.

When using Internet Explorer, do *not* use Compatibility View. This is not supported by the Connective browser package.

- When asked, enter your PIN.

If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message appears on-screen.



Sign with BeLawyer card

Signing field 1 of marker



Connect a card reader and insert your card.

- Click **Finish** to complete the process.

Sign with itsme

Attention:

- Before you can sign with itsme, you must have [signed up for an ItsMe account](#) and the itsme app must be installed on your mobile phone.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

To sign with itsme:

- Enter your **Mobile Phone number**, and click **Send**.



Sign with Itsme



en ▾

Identify yourself

Mobile Phone number

BE (+32) 4

Remember my phone number ?

Send

- When you're using itsme signing for the first time, you're prompted to create a certificate.
- Click **Create my certificate**. A message is now sent to your itsme app.



en ▾

Create your certificate

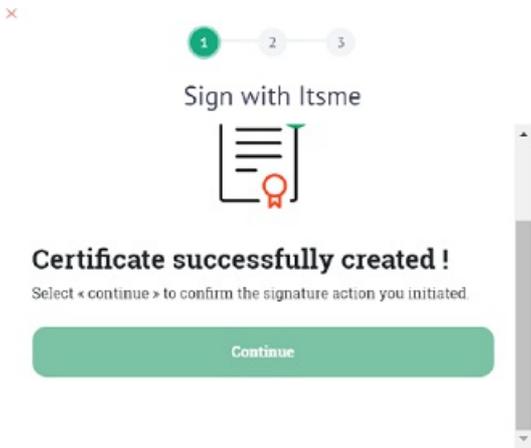
To enable the electronic signature with itsme, a signature certificate linked to your identity needs to be created.

By clicking on « Create my certificate », you agree to Quo Vadis [terms and conditions](#).

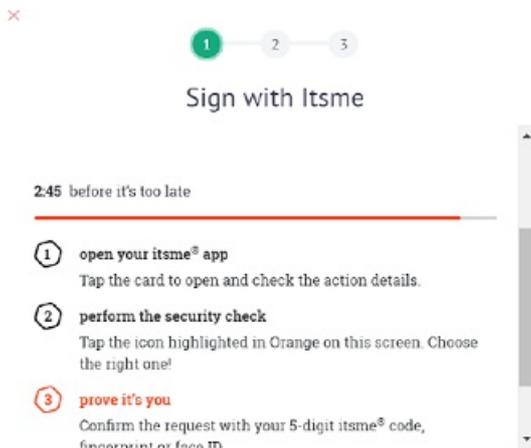
Create my certificate

- Open your itsme app on your mobile phone, and click **Confirm**.

- Then enter your pincode, and click **OK**. The certificate will now be created.
- When the certificate has been created the following message appears in eSignatures.



- Click **Continue**.
- You're now prompted to return to your itsme app.
- Confirm your identity in the itsme app.



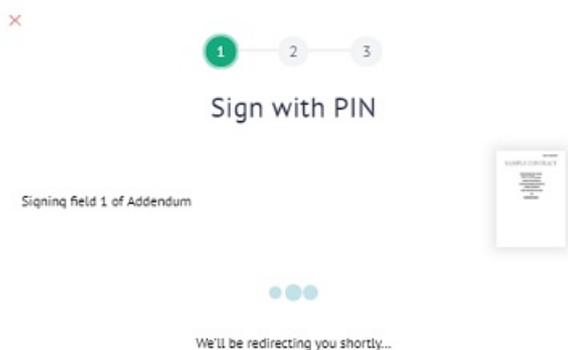
- Your document will now be signed with itsme.

Sign with custom signing type

As of eSignatures 5.1, a custom signing method - based on OpenID Connect - may have been configured that allows users to sign by pincode.

The name and icon of the signing type depends on the configuration done in the Configuration Index.

By way of example, we've named the custom signing method 'PINCODE'.



- Enter the pincode in the window that appears, and click **Validate**. If you don't remember your pincode or haven't received one, contact your system administrator.
- Click **Yes** to confirm.
- Your document will now be signed.

Sign with PIN

Signing field 1 of Addendum



Please wait while we add your signature to each of the selected documents.

Are you using your card reader to sign? Please leave your eID card inserted until the signing process has finished completely.



Cancel

- When you are done, close the browser tab.

2.5.2 How do I QuickSign a package?

Signing a package functions in a similar way as signing a document. See [Signing a document step-by-step](#) for more information.

2.5.3 How do I bulk sign?

By means of bulk signing you can sign multiple documents and packages simultaneously, using a single signature.

Attention: whether you are able to bulk sign depends on the configuration of your eSignatures solution. If this feature has not been enabled you won't be able to bulk sign.

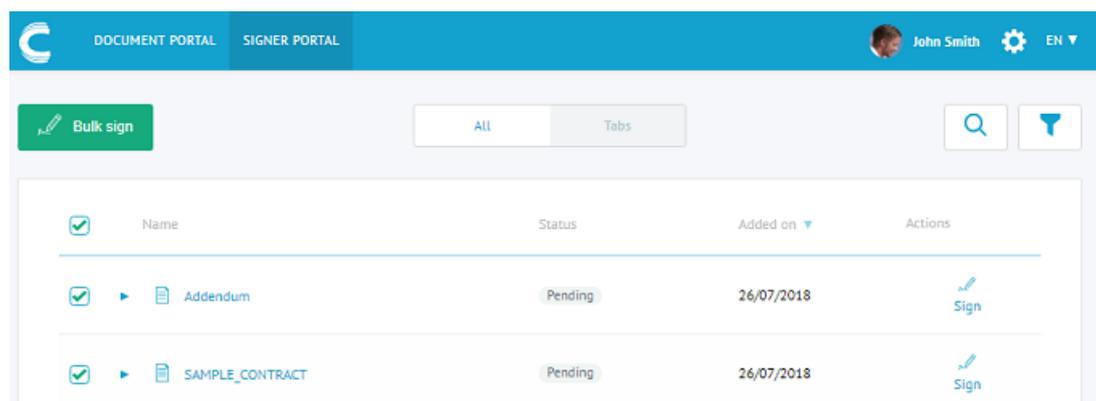
Limitations

- Bulk signing can only be combined with eID signing.
- Bulk signing requires a transparent card reader, i.e. a card reader without PIN pad.
- Bulk signing should not be used on documents or packages that contain legal notices, since legal notices will be ignored during bulk signing.
- Bulk signing is only supported when you log in to the Signer Portal on a computer. It's not possible to bulk sign in an app, or in a mobile web browser.
- It is not supported to select documents/packages across different pages in the Signer Portal and bulk sign them. The documents/packages you bulk sign must be displayed on the same page in the Document Portal.
- **Beware of the size limitations:** the same limitations apply as to a single package, meaning:
 - The total size of all packages combined must not exceed 150 MB.
 - A package may contain a maximum of 15 documents.
 - A single document must not exceed 30 MB.
 - The physical dimensions of a document must not exceed 3.99 m by 3.99 m.
 - A total number of 20 documents spread over different packages can be bulk signed.

As long as these size limitations are respected bulk signing of packages is supported. We recommend you do not upload files that exceed these limits. Depending on the internet connection, large documents may affect user experience and signing performance. Documents that exceed the specified limitations are officially not supported.

To bulk sign:

- [Log in](#) to your eSignatures account.
- Click the **Signer Portal** tab.
- Select the different documents/packages that require signing. The check boxes of documents/packages that can't be selected - because they require a signing method other than eID - are not clickable.
- To select all the documents/packages that can be bulk signed all at once, click the check box next to **Name**.



- Click **Bulk sign**.
- Then click **Start bulk signing**.

×

Confirm bulk sign?

Cancel

Start bulk signing 2 packages

- Connect your card reader and insert your eID card. Follow the on-screen instructions to complete the signing.

2.5.4 How do I sign face to face?

Face-to-face signing comes down to signing in the Document Portal. Face-to-face signing is used when you're the initiator - i.e. the person who uploads and sends the documents for signing - and your signers are physically present. This method allows signers to sign their documents immediately instead of having to check their email, log in to their account or open the app, and then do the signing.

See [Signing in the Document Portal](#) for more info.

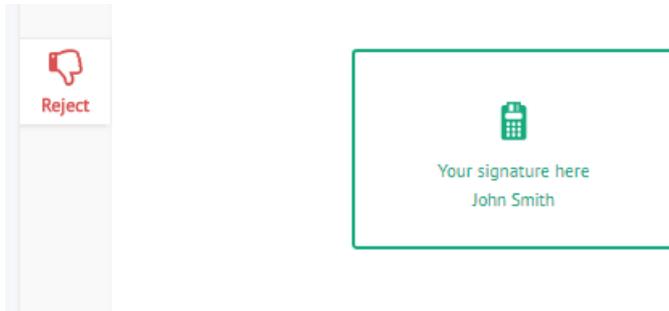
Tip: also check out this [video tutorial](#) on face to face signing:

2.5.5 How do I reject a document?

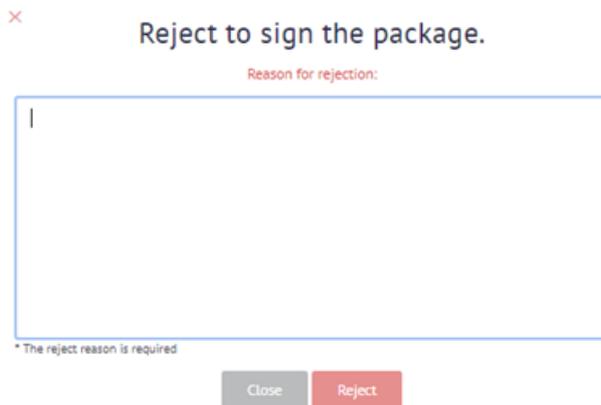
As a signer you're not obliged to sign a document. When you disagree with the conditions you can choose to reject it.

To reject a document:

- Open your document.
- Click the **Reject** button on the left-hand side of the document.

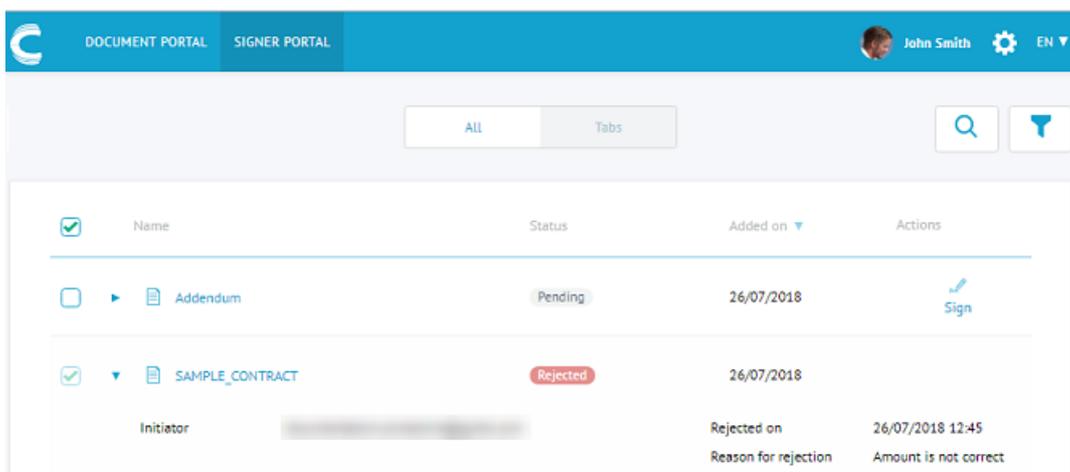


- Enter a reason for rejection, and then click **Reject**. Note that entering a reason for rejection is mandatory.



- When you're done, close the browser tab.

The initiator who sent the document can see in the Document Portal that the document has been rejected and why.

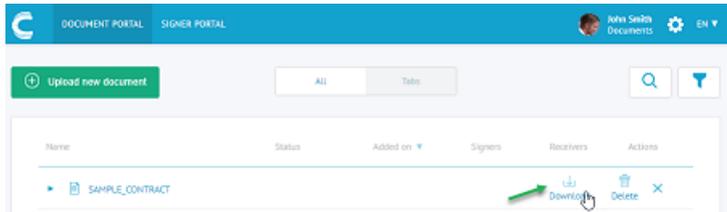


2.5.6 How do I download a signed document?

A document can only be downloaded once it's been signed. Initiators can download documents from the Document Portal. Signers can download documents from the Signer Portal.

Downloading from the Document Portal

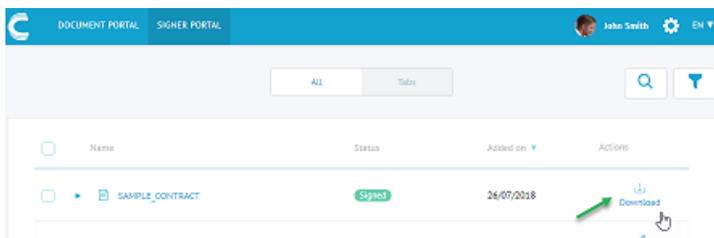
- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Click the menu icon **☰** in the **Actions** column, next to the document you want to download.
- Click the **Download** button. Documents are downloaded as .pdf files and packages are downloaded as .zip files. The downloaded files can be seen in the bottom left corner of your browser, or at the top of the screen.



- Click the document to open it.

Downloading from the Signer Portal

- [Log in](#) to your eSignatures account.
- Click the **Signer Portal** tab.
- Scroll to the document you want to download, and then click the corresponding **Download** button. Documents are downloaded as .pdf files and packages are downloaded as .zip files. The downloaded files can be seen in the bottom left corner of your browser, or at the top of the screen.



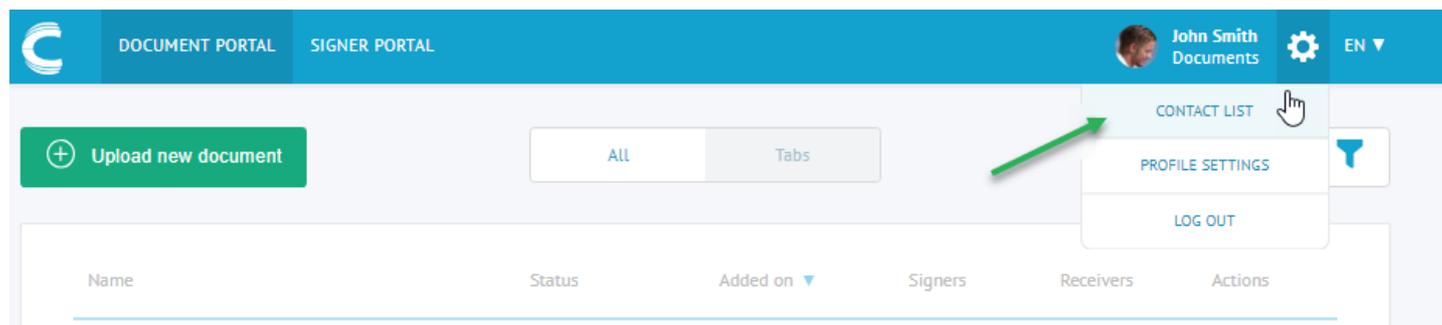
2.6 Managing contacts and accounts

Instead of creating contacts one by one during the upload of a document, you can also create contacts in the **Contact List** section of eSignatures, and import existing contact lists from Google and Office 365.

Important: don't confuse *contacts* with *users*. Contacts simply receive emails with documents to sign. Users have access to the eSignatures WebPortal.

To access the Contact List section:

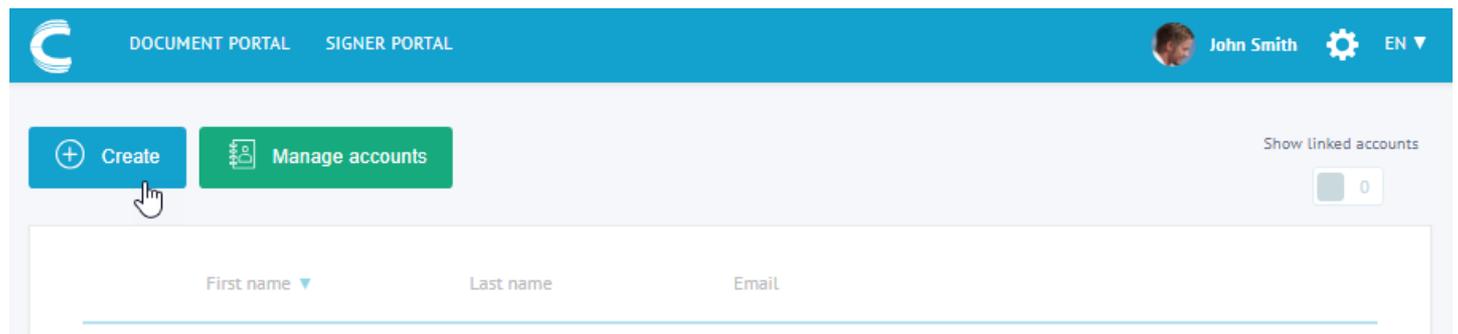
- Click the settings icon in the top toolbar.
- Click **Contact list**.



- In the **Contact list** section you can do the following actions:
 - [Create new contacts](#)
 - [Edit existing contacts](#)
 - [Delete existing contacts](#)
 - [Manage accounts](#)

2.6.1 Create new contacts

- Click the settings icon  in the top toolbar.
- Click **Contact list**.
- Click **Create** to add a new contact.



- Enter the following information. The fields with an asterisk are mandatory.

Email address

Personal title

First Name

Last Name

Birthdate

Note: If the **MandatedSignerType** is set to **nameandbirthdate** in the Configuration Index, the birth date is required. The authenticity of the signer will be checked against his first name, last name and birth date when signing with eID, BeLawyer card or itsme. Make sure the name of a contact is identical to the one on the signing certificate of his Belgian eID card, BeLawyer card or itsme account.. The contact's first name must be identical to the given names on the signing certificate, and the contact's last name must be identical to the signing certificate surname. Note however that there is no guarantee that a person's details are correctly registered on the eID, BeLawyer card or itsme account.

Phone number

Note: if you want the contact to be able to use SMS signing, make sure to enter the mobile phone number. The following countries are supported: Afghanistan, Albania, Algeria, American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia and Herzegovina, Botswana, Brazil, British Indian Ocean Territory, British Virgin Islands, Brunei, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Christmas Island, Cocos Islands, Colombia, Comoros, Cook Islands, Costa Rica, Croatia, Cuba, Curacao, Cyprus, Czech Republic, Democratic Republic of the Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equatorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Fiji, Finland, France, French Polynesia, Gabon, Gambia, Georgia, Germany, Ghana, Gibraltar, Greece Greenland, Grenada, Guam, Guatemala, Guernsey, Guinea, Guinea-Bissau, Guyana, Haiti, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Isle of Man, Israel, Italy, Ivory Coast, Jamaica, Japan, Jersey, Jordan, Kazakhstan, Kenya, Kiribati, Kosovo, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Mauritania, Mauritius, Mayotte, Mexico, Micronesia, Moldova, Monaco, Mongolia, Montenegro, Montserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, North Korea, Northern Mariana Islands, Norway, Oman, Pakistan, Palau, Palestine, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn, Poland, Portugal, Puerto Rico, Qatar, Republic of the Congo, Reunion, Romania, Russia, Rwanda, Saint Barthelemy, Saint Helena, Saint Kitts and Nevis, Saint Lucia, Saint Martin, Saint Pierre and Miquelon, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Serbia, Seychelles, Sierra Leone, Singapore, Sint Maarten, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, South Korea, South Sudan, Spain, Sri Lanka, Sudan,

Suriname, Svalbard and Jan Mayen, Swaziland, Sweden, Switzerland, Syria, Taiwan, Tajikistan, Tanzania, Thailand, Togo, Tokelau, Tonga, Trinidad and Tobago, Tunisia, Turkey, Turkmenistan, Turks and Caicos Islands, Tuvalu, U.S. Virgin Islands, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Vatican, Venezuela, Vietnam, Wallis and Futuna, Western Sahara, Yemen, Zambia, Zimbabwe.

Important: whether all these countries are available in eSignatures, depends on the configuration of the **Configuration Index**.

Language

National security number.

This field is only available - and mandatory - if the **MandatedSignerType** is set to **matchid** in the Configuration Index. The authenticity of the signer will be checked against his national security numbers when signing with eID. Note that only numerical characters must be inserted.

Create new contact

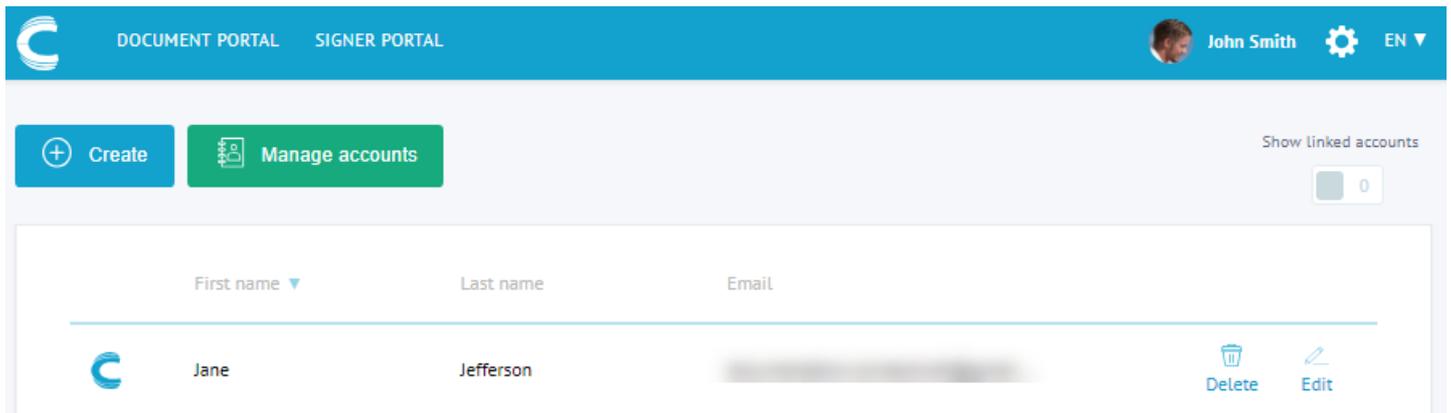
<input type="text" value="Email"/> Enter a valid email address.	<input type="text" value="Personal title"/> Enter a personal title.		
<input type="text" value="First Name"/> Enter your first name here.	<input type="text" value="Last Name"/> Enter your last name here.		
<input type="text" value="DD"/> Day	<input type="text" value="-"/> Month	<input type="text" value="YYYY"/> Year	<input type="text" value="BE (+32)"/> Enter a valid phone number to receive SMS OTP.
<input type="text" value="English"/> Choose a preferred language.			

- Click **Confirm** to create the contact. The new contact is now added to the contacts list.

The screenshot shows the top navigation bar with 'DOCUMENT PORTAL' and 'SIGNER PORTAL' tabs. The user 'John Smith' is logged in. Below the navigation bar, there are 'Create' and 'Manage accounts' buttons. A 'Show linked accounts' toggle is set to '0'. The main content area displays a table with columns for 'First name', 'Last name', and 'Email'. One contact is listed: Jane Jefferson. Action buttons for 'Delete' and 'Edit' are visible for this contact.

2.6.2 Edit existing contacts

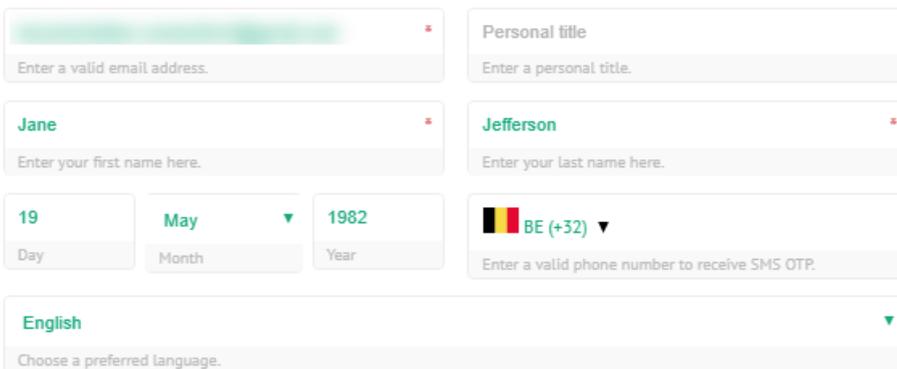
- Click the settings icon  in the top toolbar.
- Click **Contact List**.
- Click the **Edit** button next to the contact you want to edit.



The screenshot shows the top navigation bar with 'DOCUMENT PORTAL' and 'SIGNER PORTAL' links, a user profile for 'John Smith', and a language dropdown set to 'EN'. Below the navigation bar are two buttons: 'Create' (blue) and 'Manage accounts' (green). To the right, there is a 'Show linked accounts' toggle switch set to '0'. The main content area displays a table with columns for 'First name', 'Last name', and 'Email'. A single contact is listed with the first name 'Jane' and last name 'Jefferson'. To the right of the contact name are 'Delete' and 'Edit' icons.

- Edit the contact's information.
- Click **Confirm**.

Edit this contact

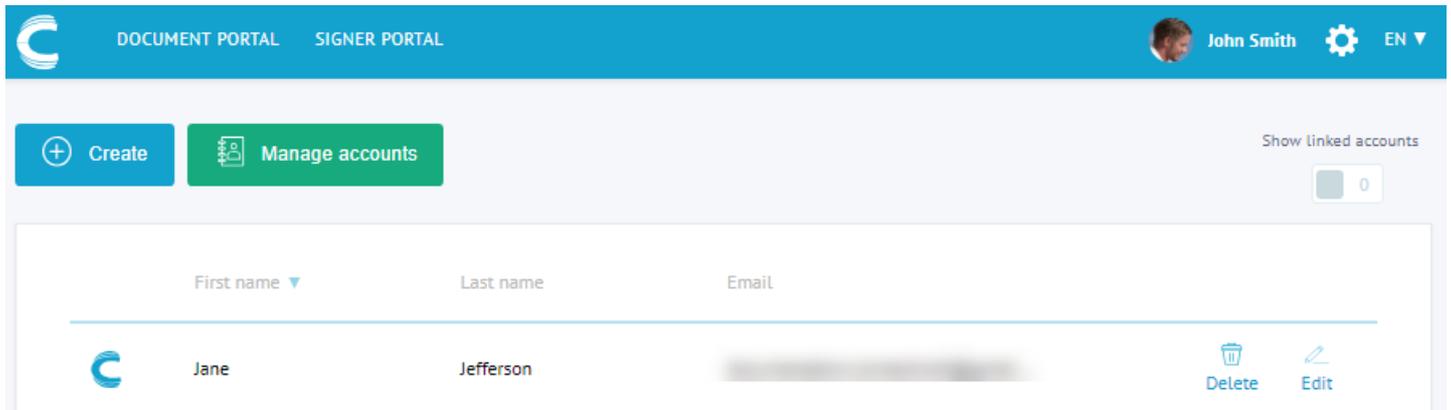


The 'Edit this contact' form contains several input fields:

- Email:** A text input field with a placeholder 'Enter a valid email address.' and a red error icon.
- Personal title:** A text input field with a placeholder 'Enter a personal title.' and a red error icon.
- First name:** A text input field with the value 'Jane' and a placeholder 'Enter your first name here.' and a red error icon.
- Last name:** A text input field with the value 'Jefferson' and a placeholder 'Enter your last name here.' and a red error icon.
- Birth date:** Three separate input fields for 'Day' (19), 'Month' (May), and 'Year' (1982).
- Phone number:** A dropdown menu for the country code (BE (+32)) and a text input field with a placeholder 'Enter a valid phone number to receive SMS OTP.'
- Language:** A dropdown menu with the value 'English' and a placeholder 'Choose a preferred language.'

2.6.3 Delete existing contacts

- Click the settings icon  in the top toolbar.
- Click **Contact List**.
- Click the **Delete** button next to the contact you want to delete.
- Click **Confirm** to delete.



The screenshot shows the top navigation bar with the Document Portal logo, 'DOCUMENT PORTAL', 'SIGNER PORTAL', user profile 'John Smith', a settings gear icon, and 'EN'. Below the navigation bar are two buttons: 'Create' (blue) and 'Manage accounts' (green). To the right is a 'Show linked accounts' toggle set to '0'. The main content area displays a table with columns for 'First name', 'Last name', and 'Email'. A single contact is listed: Jane Jefferson. To the right of the contact name is a 'Delete' button (trash icon) and an 'Edit' button (pencil icon).

First name ▼	Last name	Email	
Jane	Jefferson		 

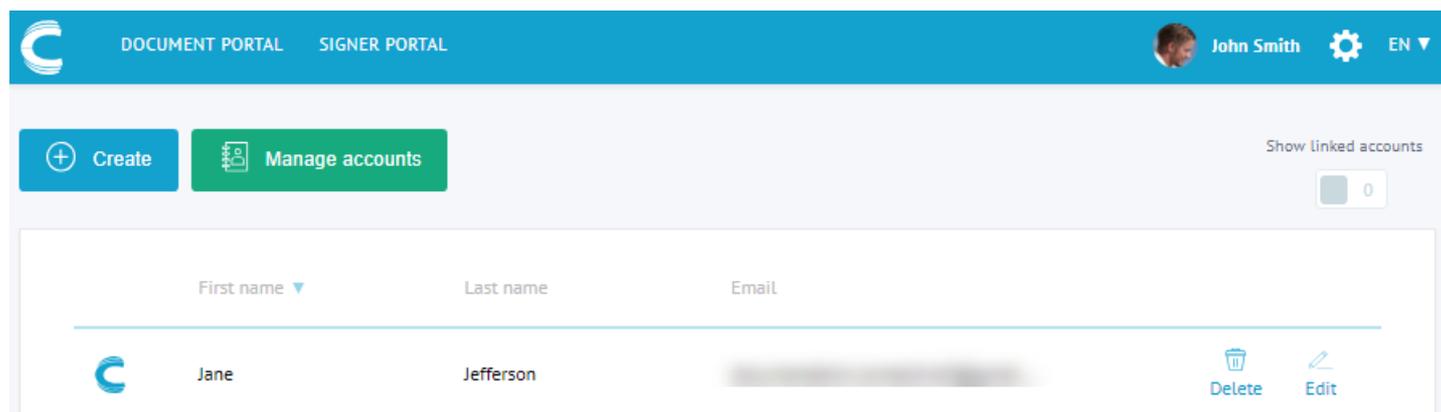
2.6.4 Manage accounts

Besides adding contacts manually to your account you can also link the contacts you have in Google Contacts and Office 365 to eSignatures. When these accounts have been linked, your contacts are available in the **Signers** list and in the **Receivers** list.

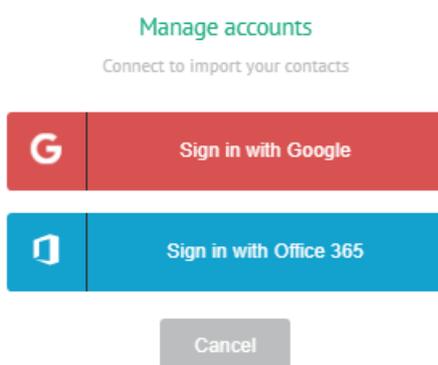
Important: in order for this feature to work, the **Cloud settings** must be properly configured in the Configuration Index. To learn how to do so, see **Connective - eSignatures 5.2.7 - Configuration Documentation**.

- Click the settings icon in the top toolbar.
- Click **Contact List**.
- Click **Manage accounts**.

Tip: if the **Manage accounts** button is not displayed, this means the Cloud settings have not been properly configured.



- Select the account you want to link to eSignatures:
 - **Sign in with Google.**
 - **Sign in with Office 365.**



Signing in with Google

- Click **Sign in with Google**.
- Select the Google account you want to link to eSignatures.
- Enter your password, and click **Next**.
- Click **Allow** to allow eSignatures to view your contacts.
- Your Google contacts are now linked to eSignatures.
- Select **Show linked accounts** to display them.

Note that linked accounts can't be edited or deleted inside eSignatures.

Sign in with Office 365

- Click **Sign in with Office 365**.
- Select the Microsoft account you want to link to eSignatures.
- Enter your credentials and click **Next**.
- Click **Allow** to allow eSignatures to view your contacts.

- Your Office 365 contacts are now linked to eSignatures.
- Note that these are only the local contacts, not the company's entire directory.
- Select **Show linked accounts** to display them.

Unlinking an account

- Click **Manage Accounts**.
- Click **Unlink from Google**.

or

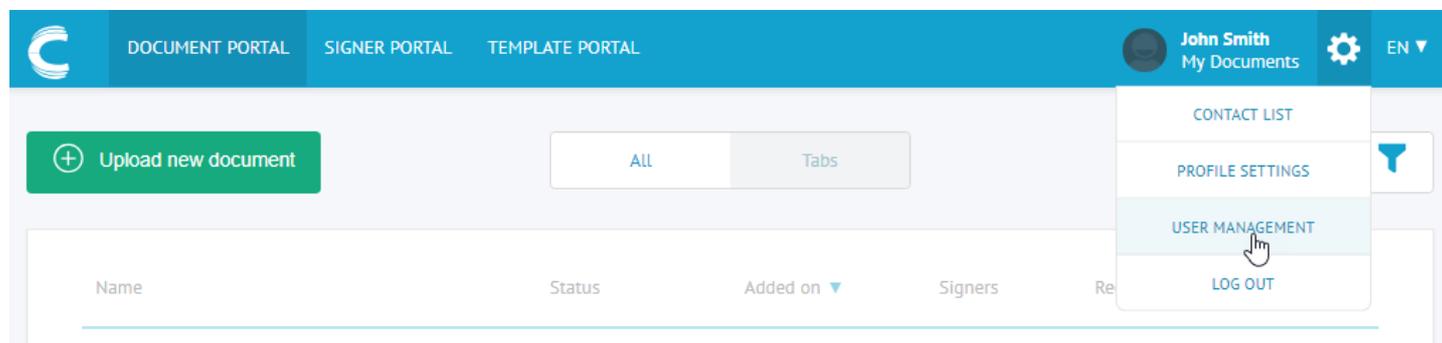
- Click **Unlink from Office 365**.

2.7 User management

To gain access to the **User management** section, contact your system administrator.

To open User management:

- Click the settings icon  the top toolbar.
- Click **User management**.



In User management you can do the following actions:

- Create, edit and delete [users](#).
- Create, edit and delete [user groups](#).
- Create, edit and delete [document groups](#).
- Create, edit and delete [template groups](#).

Important: don't confuse *contacts* with *users*. Contacts simply receive emails with documents to sign. Users have access to the eSignatures WebPortal.

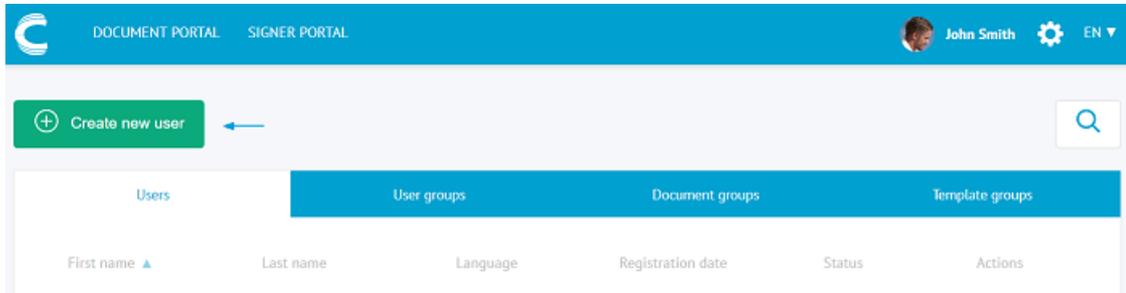
2.7.1 Users

The **Users** tab gives an overview of all the users in your system. As administrator you can create new users, edit existing users and delete users.

Note: the users who registered themselves via the eSignatures solution's homepage are also displayed here.

Create a new user

- In the **User Management** section click the **Users** tab.
- Click **Create new user**.



- Enter the new user's personal information: First Name, Last Name, Email and Company (optional).
- Select the user's preferred language. The emails the user will receive are written in this language.
- Click **Confirm**.
- The new user is added to the users list, but his status is set to inactive.

Important: don't change the status to active yet. This interferes with the registration procedure. To activate his account, the user needs to open the link that was automatically sent by email and complete the registration process. Once the registration is complete, the user can enter a password of his choosing and his status is set to active.

- A new user is by default added to the **Default User Group**. To add a user to a different user group, see [User Groups](#).

Edit a user

- Click the **Edit** button next to the user you want to edit.
- Edit the personal information.
- To temporarily keep the user from having access to eSignatures, set **Allow or deny access to the application** to disabled.

Update user

<input type="text" value="Enter your first name here."/>	<input type="text" value="Enter your last name here."/>
<input type="text" value="Enter a valid email address."/>	<input type="text" value="Dutch"/>
<input type="text" value="Connective NV"/>	<input type="checkbox"/>

Allow or deny access to the application.

- Click **Confirm** when you're done.

Delete a user

- Click **Delete** next to the user you want to delete.
- Click **Confirm**.

2.7.2 User groups

In the **User groups** tab you define the permissions of all users belonging to the user group. Users don't have individual permissions in eSignatures.

The default user groups are **Administrators** and **Default User Group**.

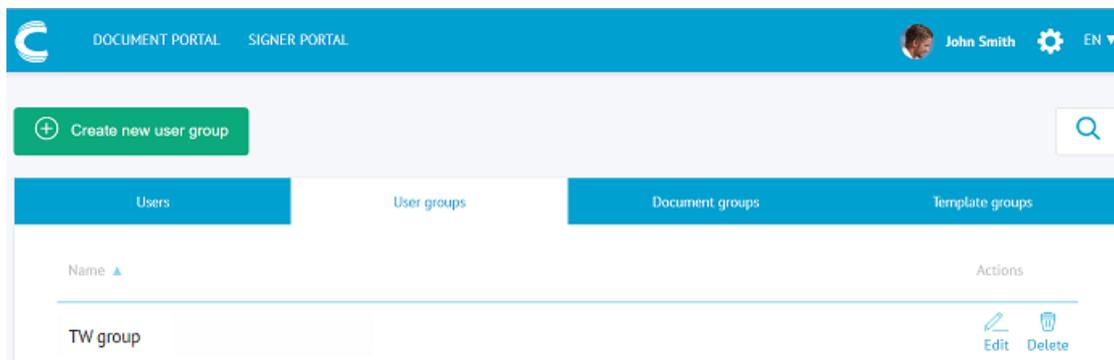
Important:

- Never change the names of the default user groups.
- Be careful when changing the settings of the default user groups, for new users are automatically added to the Default User Group and inherit its permissions. Granting administrator rights to the Default User Group would be ill-advised for instance.

If you want to apply special settings to a User group, you're recommended to create a new user group.

Create a new user group

- In the **User Management** section, click the **User groups** tab.
- Click **Create new user group**.



- Enter a name for the user group and click **Confirm**. The new user group is added to the **User Groups** list.
- Once a user group has been created you need to configure its settings. A newly created user group doesn't contain any users.

Edit a user group

- Click the **Edit** button next to the user group you want to edit.
- The following settings can be configured:
 - **General**: which actions are the users of this group allowed to do?
 - **Users**: which users belong to this user group?
 - **Document groups**: to which document groups do the users of this user group have access?
 - **Template groups**: to which template groups do the users of this user group have access?

General application settings

- Select the permissions the users of this user group must have.
- When a setting is enabled, its button is set to 1 instead of 0.
- When you are done, click **Save**.

Update Default User Group

General **Users** Document groups Template groups

GENERAL APPLICATION SETTINGS

- Access to the Document portal
- Access to the Signer portal
 - Signer portal - Bulk sign documents
- Access to the Template portal
- Access to the User management portal
 - Create new document group(s)
 - Create new template group(s)
 - Create new user(s)
 - Create new user group(s)
 - Delete existing document group(s)
 - Delete existing template group(s)

Cancel Save

Users

Determine which users must be part of the user group.

- Click the **Users** tab.
- The users outside the group are listed in the left-hand column. The users inside the group are listed in the right-hand column.
- Click the plus sign next to a user to add them to the user group.
- To remove a user, click the X in the right-hand column.

Update Default User Group

General **Users** Document groups Template groups

Search for a name

USERS NOT IN THE USER GROUP

admin admin

USERS IN THE USER GROUP

Cancel Save

- Changes are saved automatically.

Document groups

Determine the permissions the users of this user group must have on the available Document groups. By default, users of a new user group don't have any permissions at all.

- Select the Document group to which the users must have permissions.
- Click the permissions that must be enabled.
- Click **Save**.

Update Default User Group

General Users **Document groups** Template groups

DOCUMENT GROUPS	PERMISSIONS
My Documents	Upload document(s) <input type="checkbox"/> Sign document(s) <input type="checkbox"/> Download document(s) <input type="checkbox"/> Delete existing document(s) <input type="checkbox"/> View <input type="checkbox"/>

Note that the settings you configure here are automatically stored in the [Document groups](#) settings as well.

Template groups

Determine the permissions the users of this user group must have on the available Template groups.

By default, users of a new user group don't have any permissions at all.

- Select the Template group to which the users must have permissions.
- Click the permissions that must be enabled.
- Click **Save**.

Update Default User Group

General Users Document groups **Template groups**

TEMPLATE GROUPS	PERMISSIONS
Templates	Create new template(s) <input type="checkbox"/> View existing template(s) <input type="checkbox"/> Update existing template(s) <input type="checkbox"/> Delete existing template(s) <input type="checkbox"/>

Note the settings you configure here are automatically stored in the [Template groups](#) settings as well.

Delete a user group

- To delete a user group, click the **Delete** button next to it.

2.7.3 Document groups

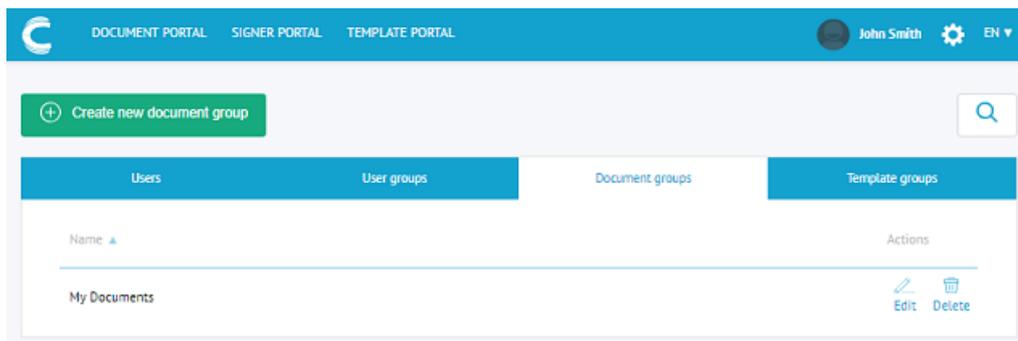
A Document group is a container of documents. By means of Document groups, documents can be shared amongst different users, controlled by the permissions given to user groups. The default Document group is **My Documents**.

Important:

- Never change the names of the default document group.
- Be careful when changing the settings of the default document groups, for all new users automatically have access to this document group. If you want to apply special settings to a document group, you are recommended to create a new document group.

Create a new document group

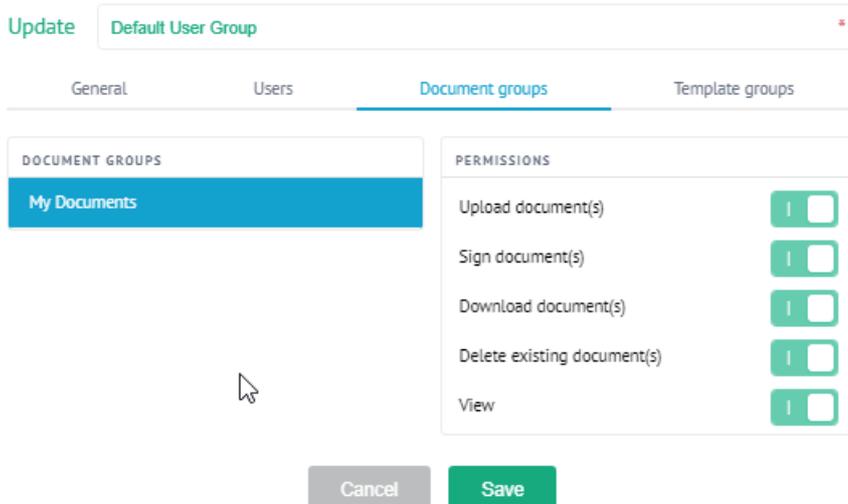
- In the **User Management** section click the **Document groups** tab.
- Click **Create new document group**.



- Enter a name for the document group and click **Confirm**. The new document group is added to the **Document groups** list.
- Once a document group has been created you need to configure its settings. By default, users don't have any permissions to a newly created document group.

Edit a document group

- Click the **Edit** button next to the document group you want to edit.
- The different user groups that are created in the system are listed in the left-hand column.
- Select the permissions in the right-hand column for each user group.
- When you are done, click **Save**.



The settings you configure here are automatically saved in the **Document groups** section of the selected **User group** as well.

Delete a document group

- To delete a document group, click the **Delete** button next to it.

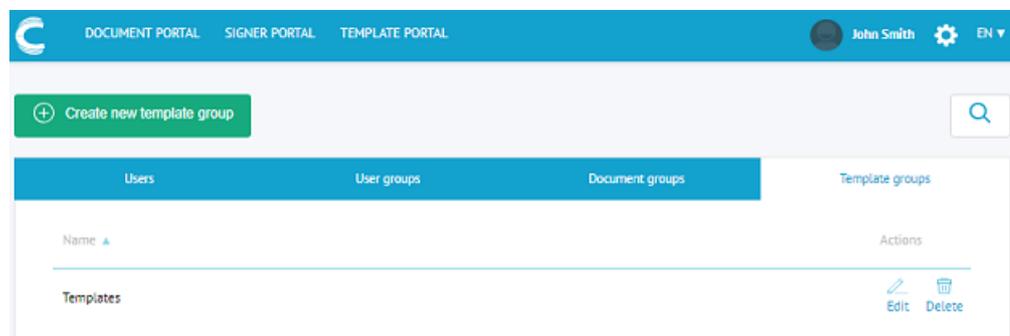
Note: a document group can only be deleted if it doesn't contain any documents.

2.7.4 Template groups

Note that the Templates feature is deprecated and will no longer be developed in future versions.

Create a new template group

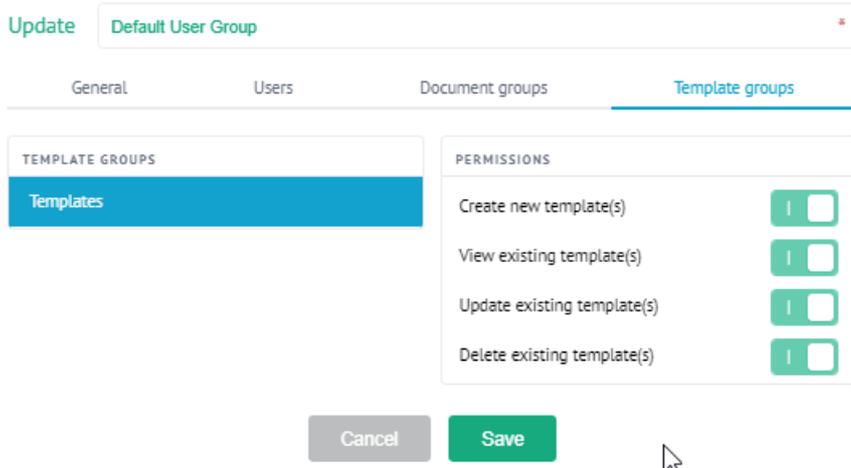
- In the **User Management** section click the **Template groups** tab.
- Click **Create new template group**.



- Enter a name for the template group and click **Confirm**. The new template group is added to the **Template groups** list.
Important: never use the default user groups (**Administrators** or **Default User Group**) as Template group.
- Once a template group has been created you need to configure its settings. By default, users don't have any permissions to a newly created template group.

Edit a template group

- Click the **Edit** button next to the template group you want to edit.
- The different template groups that are created in the system are listed in the left-hand column.
- Select the permissions in the right-hand column for each user group.
- When you are done, click **Save**.



The settings you configure here are automatically saved in the **Template groups section** of the selected **User Group** as well.

Delete a template group

Note: only empty template groups can be deleted.

To delete a template group, click the delete button next to it.

2.8 Creating templates

Note that the Templates feature is deprecated and will no longer be developed in future versions.

It is important to note that the term "template" is not used in the common sense of the word. In Connective terminology, a template refers to a **signing flow** in which you configure all the settings you would configure for each document individually in the Document Portal. When a signing flow is put in place by means of a template, all the documents that are sent via the API to this template use the same configuration. So creating templates is especially convenient for documents that need to be signed often and by the same signers. The other advantage of using templates is that you avoid having to put all these configuration settings in an API call when sending multiple documents.

Tip: to learn how to upload documents to a template via the API see **Connective - eSignatures 5.2.7 - API v3 Technical Documentation**.

In case you don't see the **Template Portal** tab in your eSignatures solution, it means you don't have access. Contact your system administrator to obtain access.

Note that the Template Portal can only be accessed in a web browser on a computer. It is not accessible from the Connective app or in a web browser on mobile device.

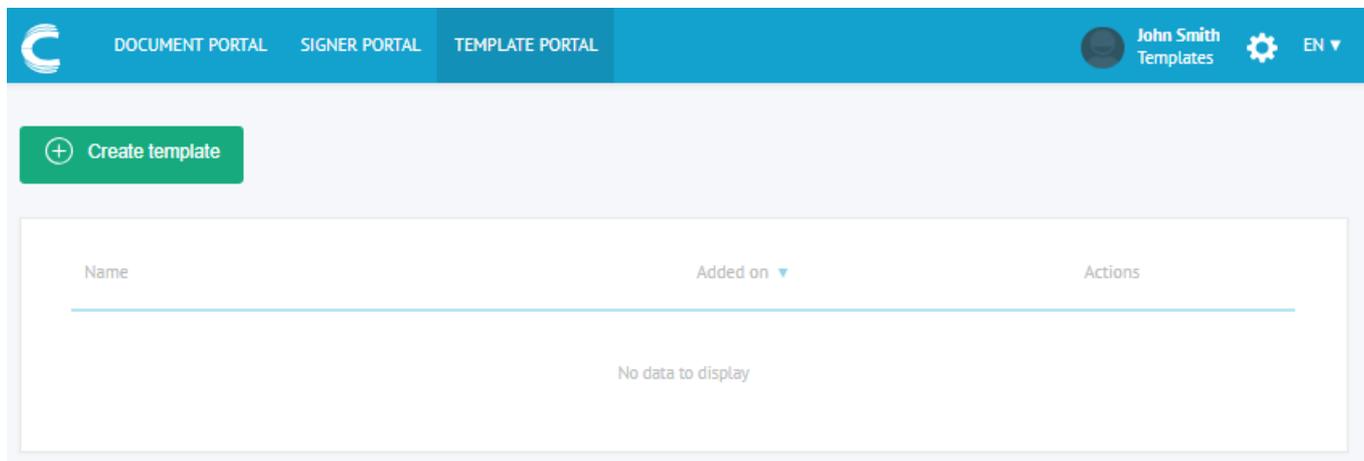
2.8.1 How do I create a template?

Tip: if you don't see the **Template Portal** tab this means you don't have access. Contact your system administrator.

- [Log in](#) to your eSignatures account.
- Click the **Template Portal** tab.

Note that the **Template Portal** can only be accessed on a computer and not in the Connective app or on a mobile device.

- Click **Create template**.



- Enter a meaningful **Template name**.
- Enter a **Callback URL** (optional).

This is the URL that will call another application each time there is a status change in the document (e.g. one person signed a field, one person rejected the document). Callback URLs are typically used by clients who don't want to use the standard emails sent by Connective and use their own driving application instead to send customized emails. If no Callback URL is entered in this field, no callback will be done.

Important: as of eSignatures 5, Callback URLs are **one-time URLs**, which means they expire once they are used. Therefore, clients' driving applications must be informed when a one-time callback URL has expired.

Important: For performance and scalability purposes, a Callback timeout has been introduced and is set to **100 seconds**. This way, if the driving application doesn't respond to eSignatures' callback, a timeout will be forced, and the rest of the flow will be finished as if the expected **200 OK** message were received. If eSignatures were to wait indefinitely for a response to finish the package, its performance would drop drastically.

For this reason, it's highly recommended that the client's callback service is developed in such a way that it sends its response as soon as possible. Any other actions done by the callback service must not depend on the response being sent but should function asynchronously.

- Enter a **Redirect URL** (optional).

This is the URL to which the application must be redirected once the document has been signed or rejected. If no URL is supplied, the signer will need to manually close the current tab in his web browser.

The **Redirect URL** you define here is the general Redirect URL on *application* level. You can also define a Redirect URL on *user* level. In the latter case, the user Redirect URL always overrules the general Redirect URL.

Important: as of eSignatures 5, Redirect URLs are **one-time URLs**, which means they expire once they are used. In case end users encounter an expired link they will have the possibility to request a new one.

- To add a tag to the template (optional), click inside the **Add tag** field and enter the content of the tag.

Tags allow you to search for templates inside the API.

Template name *
Enter a name.

Callback URL
Enter a callback URL to be called after the document is signed (optional).

Redirect URL
Enter a redirect URL to be called as soon as the document is signed (optional).

Add tag
Add tag

Once the general settings have been configured you need to add signers.

Add Signer(s)

- Click inside the **Search name or add email address** field.
- If there are known contacts in your system, the contacts are displayed in the list. You can either select an individual user or a user group.

Important: never use the default user groups Administrators and Default User Group.

Note: when you select an entire user group, its signers can only sign with eID. Otherwise it is not possible to identify who exactly signed the document.

Sign in parallel Sign sequentially

Signers

Q Search name or add email address

+ Add

Administrators	User group
Default User Group	User group

- If there are no known contacts, or if you want to create a new contact and add them as signer, click **Add** and create a new contact.

Note that it's not possible to add new user groups here. User groups can only be created in the **User Management** section.

- The selected user or user group is added to the Template. You can now configure the signer settings. See the instructions below.
- To add additional signers, repeat the steps above.

Sign in parallel Sign sequentially

Signers

Q Search name or add email address

John Smith
documentation.connective@gmail.com

+ Add a signing type

Marker name * Redirect URL

Marker name * Redirect URL

Send notifications

Default User Group
User group

+ Add a signing type

Marker name * Redirect URL

Marker name * Redirect URL

Send notifications

Configure the signer settings

- Click **Add a signing type** to select which signing methods the signer will be allowed to use. The available signing methods depend on the configuration of your system. For more information see **What are the different signing methods?**

Important: when you selected an entire User Group, the only available signing method is eID. Otherwise it is not possible to identify who exactly signed the document.

- Enter the **Text Marker name** of the signing field (mandatory). This is the Text Marker code that has been created in the API on the document that needs to be signed. A Text Marker contains a unique name and the length and height of the field: #SIG01_H_W# (where H is the height, and W is the width). E.g. #SIG01_100_200#. In this case the marker will be replaced by a field of height 100px and width 200px. For more information about markers see **Connective - eSignatures 5.2.7 - API v3 Technical Documentation**.
- For each user group you added, you can enter an individual **Redirect URL**. This individual Redirect URL overrides the general Redirect URL. When the signer has signed or rejected a document they will be redirected to the URL you provide here.

Important: as of eSignatures 5, Redirect URLs are **one-time URLs**, which means they expire once they are used. In case end users encounter an expired link they will have the possibility to request a new one.

- If you want the signer to receive an email notification when a document is ready for signing, enable the **Send notifications** option.

Arrange the signers

- Choose whether signers must **sign in parallel** or **sign sequentially**.
- When you select **Sign in parallel**, signers can sign in any given order.
- When you select **Sign sequential**, you will need to determine the order in which signers must sign. Drag the signers to the required position. The position of the signer is indicated by a number.

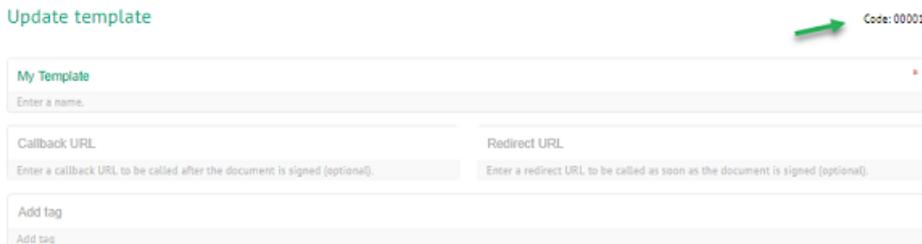
Add Receivers

Adding a receiver is an optional step. A receiver is the person who needs to receive a download link to the fully signed documents. A receiver does not sign documents.

- Click inside the **Search name or add email address** field.
- If there are known contacts in your system, the contacts are displayed in the list.
- To add a new contact, click **Add**.
 - Enter the required fields (marked by an asterisk) for the contact.
 - When you are done, click **Confirm**.

When you're done configuring the template, click **Confirm**. A unique code is now assigned to the template. You must use this code in the API call to send documents to this template. To learn how to go from here, consult **Connective - eSignatures 5.2.7 - API v3 Technical Documentation**.

Update template



Code: 00001

My Template
Enter a name.

Callback URL
Enter a callback URL to be called after the document is signed (optional).

Redirect URL
Enter a redirect URL to be called as soon as the document is signed (optional).

Add tag
Add tag

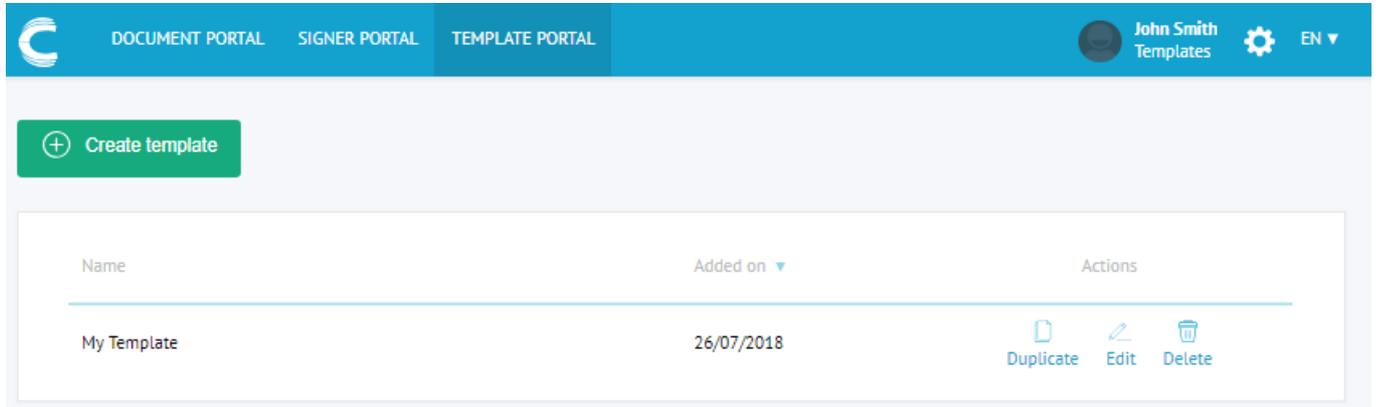


2.8.2 How do I edit a template?

- [Log in](#) to your account.
- Click the **Template Portal** tab.

Note that the **Template Portal** can only be accessed on a computer and not in the app or on a mobile device.

- Go to the template you want to edit, and click the **Edit** button.



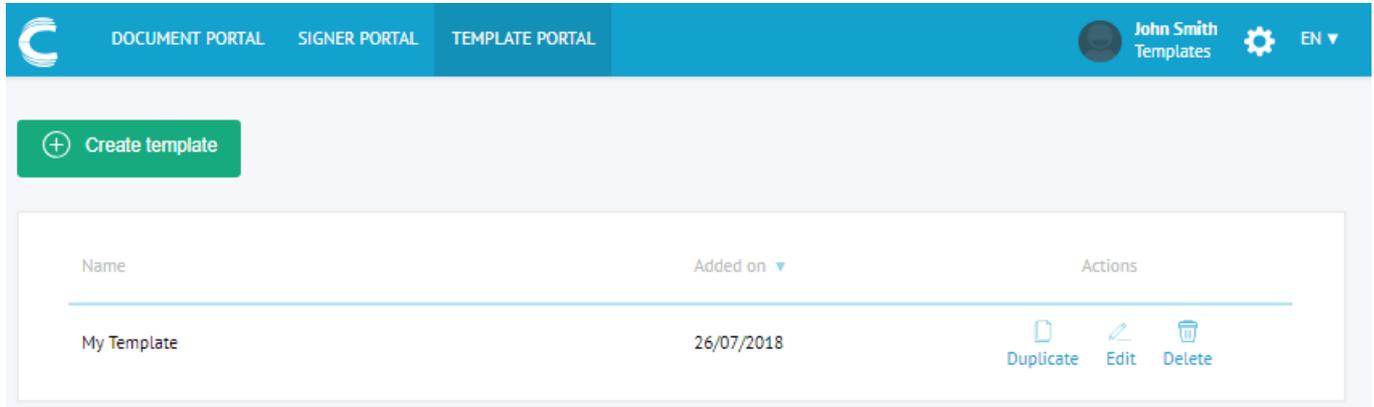
- Click **Save changes** when you're done editing.

2.8.3 How do I duplicate a template?

- [Log in](#) to your account.
- Click the **Template Portal** tab.

Note that the **Template Portal** can only be accessed on a computer and not in the Connective app or on a mobile device.

- Go to the template you want to duplicate, and click the **Duplicate** button.



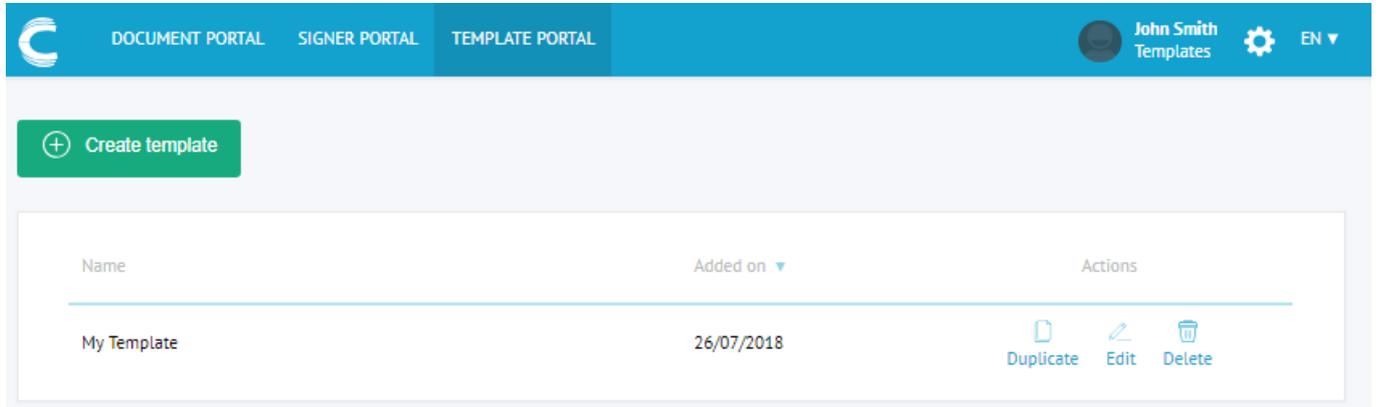
- Click **Confirm**. The duplicate will be added to the template list, using the same name and suffix `_(n)`.
- Now edit the template to modify its settings.

2.8.4 How do I delete a template?

- [Log in](#) to your account.
- Click the **Template Portal** tab.

Note that the **Template Portal** can only be accessed on a computer and not in the app or on a mobile device.

- Go to the template you want to delete, and click the **Delete** button.



The screenshot shows the 'Template Portal' interface. At the top, there is a navigation bar with three tabs: 'DOCUMENT PORTAL', 'SIGNER PORTAL', and 'TEMPLATE PORTAL'. The 'TEMPLATE PORTAL' tab is active. To the right of the navigation bar, the user's name 'John Smith' and 'Templates' are displayed, along with a gear icon and a language dropdown set to 'EN'. Below the navigation bar, there is a green button with a plus icon and the text 'Create template'. Below this button is a table with the following structure:

Name	Added on ▾	Actions
My Template	26/07/2018	Duplicate Edit Delete

- Click **Confirm**.

2.9 WebPortal FAQ

In this section we address the questions that WebPortal users might have concerning eSignatures. The questions are bundled per topic where possible.

I can no longer log in to my account

POSSIBLE CAUSE	SOLUTION
<p>You tried to do an action you don't have the user rights for, or tried to access a part of the WebPortal you don't have access to.</p>	<p>When you try to do an action you don't have user rights for, or access a part you don't have access to, the system will log you out and you won't be able to log in again.</p> <p>To solve this issue, delete your eSignatures environment cookie from your web browser. You will now be able to log back in.</p> <p>Tip: in case you don't know which cookie to delete, delete all cookies from your web browser.</p>

2.9.1 Uploading documents

Which file types can I upload?

What is the size limit for documents I upload?

Which output files can be generated?

Which file types can I upload?

eSignatures supports the following file formats:

- Microsoft Word files (.doc or .docx).
- Plain text files (.txt)
- Portable Document Format documents (.pdf). A PDF document may either be a regular PDF document, a PDF/A-1 or a PDF/A-2 document.

An administrator may disable any of the file formats listed above in the Configuration Index.

The files you upload are always converted to PDF documents as [output](#).

What is the size limit for documents I upload?

The following size limitations apply:

- A package containing multiple documents must not exceed 150 MB.
- A package may contain a maximum of 15 documents.
- A single document must not exceed 30 MB.
- The physical dimensions of a document must not exceed 3.99 m by 3.99 m.

We recommend you do *not* upload files that exceed these limits. Depending on the internet connection, large documents may affect user experience and signing performance. Documents that exceed the specified limitations are officially not supported.

Which output files can be generated?

eSignatures can generate the following output files:

- PDF
- PDF/A-1
- PDF/A-2

PDF is the standard PDF format.

PDF/A-1 is a standard long-term archiving format and is the constrained version of Adobe PDF version 1.4.

PDF/A-2 is also a standard long-term archiving format and is the constrained version of Adobe PDF version 1.7.

Both PDF/A formats prohibit features that are ill-fitted for long-term archiving.

Important: when using ItsMe as signing type, you must use **PDF/A-1** or **PDF/A-2** as output type.

An administrator may disable any of the file formats listed above in the Configuration Index.

2.9.3 Signing documents

[What are the different signing methods?](#)

[How do I QuickSign a package?](#)

[Why do I need to install card reader software?](#)

[How do I install the card reader software?](#)

[Which smart card readers are supported?](#)

[Which biometric signature pads are supported?](#)

[Which operating systems are supported?](#)

[Which web browsers are supported?](#)

[How does asynchronous work?](#)

[What is the legal value of a digital signature?](#)

What are the different signing methods?

The following signing methods are supported in eSignatures. Note that not all signing methods may be available in your eSignatures solution, depending on the configuration.

Tip: if a desired signing method is not available, contact your administrator to enable it in the Configuration Index.

Manually

Manual signing means that a manual signature is needed, as if signing a paper document with a regular pen. Signers need to draw their signature on-screen using a mouse or touchpad, or using their fingers on a touchscreen.

eID

Select **eID** if you want signers to use their Belgian eID to sign.

A third-party card reader is required. See [Which smart card readers are supported?](#) for more info.

Note: when using the eID signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

Manually + eID

This signing method combines manual signing and eID signing. Users first need to draw their signature manually and then enter their eID.

A third-party card reader is required. See [Which smart card readers are supported?](#) for more info.

Note: when using the eID signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

SMS OTP

Select **SMS OTP** if you want signers to sign using an SMS code. They will need to enter the last four digits of their phone number. In return, they'll receive a one-time password via SMS.

Note: the phone number of the signers must be known in order to use their signing method.

Email OTP

Select **Email OTP** if you want signers to sign using a one-time password they receive via email. They will need to complete their email address. In return, they'll receive the password via email.

iDIN

iDIN signing allows signers to sign using their Dutch bank card.

The following issuers are currently supported: ABN Amro Bank, ASN Bank, ING Bank, Rabobank, RegioBank, SNS Bank, Triodos Bank.

Biometric

Select **Biometric** if you want signers to sign using one of the following devices:

- A biometric signature pad.
- A Bamboo Finline Stylus (only on iPad)

Signature pads and the Bamboo Finline Stylus allow to capture the biometrical characteristics of a signature, like where the pen is located, when the pen tip is pressed down, and how hard it is pressed down. These data are added to the signature, which would allow the signature pad manufacturer to verify the authenticity of the signature when required.

Important:

eSignatures currently supports the following biometric signature pads: Wacom STU-430, STU-530 and STU-540. Make sure the necessary Wacom SDK is installed on the signer's computer: wacom-signature-sdk-x86-3.19.2.msi (32-bit Operating System) or wacom-signature-sdk-x64-3.19.2.msi (64-bit Operating System).

eSignatures currently supports the Wacom Bamboo Fintline Stylus (CS-600), but only on iPad.

Due to the technical setup of biometric signatures, each document within a package must be signed individually.

BeLawyer

Select **BeLawyer** if you want signers to use their electronic lawyer's card.

A third-party card reader is required. See [Which smart card readers are supported?](#) for more info.

Note: when using the BeLawyer signing method for the first time you need to install the **Connective browser package**. To learn how to do so, see the [Browser package section](#) on the documentation website.

Itsme

Select **itsme** if you want signers to use their itsme app.

Itsme is your digital ID to log in securely, to share your ID data or to sign using your mobile phone.

Prerequisites:

- The signer must have an itsme account and have the ItsMe app installed on his mobile phone in order to sign.

Important notes:

- When using itsme, the output format of your documents (which you selected at [Step 1: upload documents](#)) must be **PDF/A-1** or **PDF/A-2**. It's the administrator's responsibility to make sure these output formats are available in the user's eSignatures solution, and it's the user's responsibility to select the correct output format. Connective does not perform any checks whether the right output format has been selected combined with itsme.
- When using itsme signing in packages, each document within the package must be signed individually.

Custom

A custom signing method - based on OpenID Connect - may have been configured that allows users to sign by pincode.

The name and icon of the signing type depends on the configuration done in the Configuration Index.

2.5.2 How do I QuickSign a package?

Signing a package functions in a similar way as signing a document. See [Signing a document step-by-step](#) for more information.

Why do I need to install card reader software?

Card reader software is required on Windows and macOS to use any signing method that requires additional hardware:

- **eID signing:** requires an eID card reader
- **BeLawyer signing:** requires a transparent card reader
- **Biometric signing:** requires a biometric signing pad

To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

If you're using an iOS or Android device and want to use a signing method that requires additional hardware, you need a mobile app. Contact your administrator to check if an app is available for your company. The Connective demo app in the App store and Google Play can only be used with the standard demo environment of eSignatures.

How do I install the card reader software?

To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

Which smart card readers are supported?

Windows and macOS

eSignatures supports the following brands of smart card readers on Windows and macOS:

CARD READER TYPE	EID SIGNING	BELAWYER SIGNING
Vasco Digipass 875	V	V
Vasco Digipass 870	V	V
Vasco Digipass 920	V	X
ACS ACR38	V	V
ACS AGP8202	V	V
Gemalto CT1100	V	V

Transparent readers (without PIN pad) from these brands should be supported too, provided they contain the correct drivers and the drivers have been installed on the user's computer.

Android and iOS

To sign with eID in a rebranded Connective app, the following smart card reader is required:

CARD READER TYPE	EID SIGNING
Vasco Digipass 875	V

Which biometric signature pads are supported?

The following signature pads are currently supported:

- Wacom STU-540
- Wacom STU-530
- Wacom STU-430

Make sure the necessary Wacom SDK is installed on your computer: [wacom-signature-sdk-x86-3.19.2.msi](#) (32-bit Operating System) or [wacom-signature-sdk-x64-3.19.2.msi](#) (64-bit Operating System).

Note: the Wacom Bamboo Fineline stylus is supported too as biometric signing method, but only on iPad.

Which operating systems are supported?

The following operating systems are supported:

Microsoft Windows

- Microsoft Windows 10 (64-bit)
- Microsoft Windows 8.1 (64-bit)
- Microsoft Windows 7 (64-bit)

Mac OS

- macOS Mojave
- macOS High Sierra
- macOS Sierra

Android

- Oreo 8.0-8.1
- Nougat 7.0-7.1.2
- Marshmallow 6.0-6.0.1

iOS

- iOS 11
- iOS 10
- iOS 9

Other operating systems are officially *not* supported.

Note: itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Which web browsers are supported?

The following web browsers are supported:

Windows

- Microsoft Edge 44.17763 and 42.17134
- Internet Explorer 11

Earlier versions of Internet Explorer are not supported.

Note: do *not* use Compatibility View when using a signing method that requires the Connective browser package. This is not supported by the Connective browser package.

- Google Chrome n-2
- Mozilla Firefox n-2

Mac OS

- Safari n-2
- Google Chrome n-2
- Mozilla Firefox n-2

Android

- Google Chrome n-2

iOS

- Safari n-2

Other web browsers or browser versions are officially not supported.

Note that not all signing methods are supported in a mobile web browser. All signing methods that require additional hardware are only supported in the app.

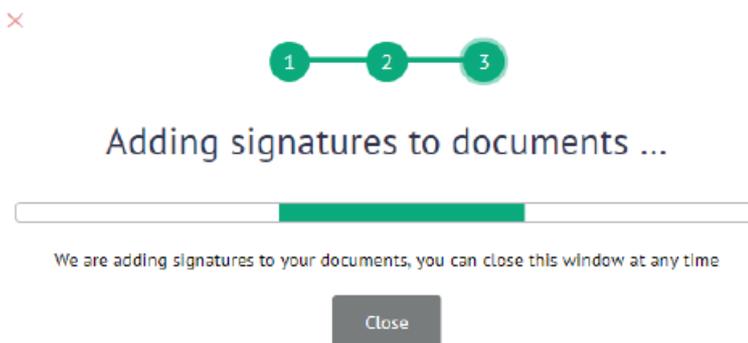
Note: itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

How does asynchronous signing work?

Asynchronous signing allows you to close the signing session and let the signing run in the background. This way, you can work on other things while your documents are being signed. This comes in handy especially when signing large documents that take quite some time to be signed.

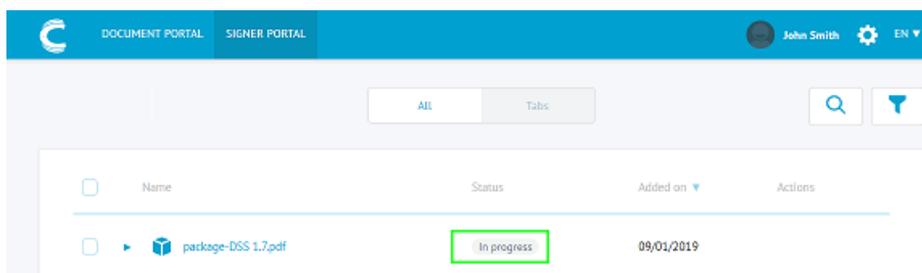
Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (though OpenID Connect) and Biometric.

- [Access the documents](#) that require your signature.
- Read the entire document and scroll down to the last page.
- When you've scrolled down to the last page, the checkbox **I declare that I have read and agree with above documents, the Terms of Use, the Privacy Policy and Cookie Policy** now becomes available. Check it and then click **Start signing**.
- Go through all the steps (if any) that require user action, such as entering a legal notice, accepting a signature policy, etc.
- The signing modal now opens. Place your signature, and then click **Next**. Note that the way of placing your signature varies depending on the signing type that was selected for you.



- You may click **Close** to the signing session, while the signing continues in the background. The status of the document is now changed to **In progress**. **Note:** the **In progress** state is only visible in the **Signer Portal**, not in the Document Portal.

Attention: when documents have been sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.



- When the signing is completed, the status changes to **Signed**.

Note: if for some reason the asynchronous signing should fail, the document will get the status **Failed**. If this happens, end users should notify the initiator (the person who sent the document), who in turn should contact the system administrator to see what went wrong.

API users can also use the **Resubmit poison queue** call to resubmit the failed documents and retry the signing. In this case the end user won't have to resign the documents. Since they already placed their signature, the system will retry the signing automatically.

What is the legal value of a digital signature?

To answer that question, we first need to make sure we're talking about the same thing. After all, the terms '**electronic signature**' and '**digital signature**' are often used interchangeably. But there's a difference.

An **electronic signature** simply captures a person's intent to agree to the content of an electronic document or a set of data. It can be a signature manually drawn on a desktop screen, but also merely the image of your signature pasted in a Word document, or your mail signature. A **digital signature**, on the other hand, relies on a cryptography-based technology, which provides an extra level of security and integrity of the document.

So a digital signature is always an electronic signature while an electronic signature is not always a digital signature.

Under eIDAS - the European legislation that oversees electronic identification and trust services for electronic transactions - there are three categories of electronic signatures. All three categories can be legally effective. The difference between them is the evidence needed to reassure a court that the signature is genuine and intentionally applied to a particular document. When using an eIDAS compliant solution like eSignatures, the signatures are legally valid across European borders.

Below you'll find an overview of the three categories of electronic signatures, and their differences.

Categories of electronic signatures

1. Basic electronic signature (BES)
2. Advanced electronic signature or **digital signature** (AES)
3. Qualified advanced electronic signature or **Qualified digital signature** (QES)

As you can see, only categories 2 and 3 qualify as **digital signatures**.

The difference between these categories are mainly based on 4 key elements:

1. **Authenticity:** Is the signature uniquely linked to the signer?
2. **Identity:** Are you capable to identify the signer?
3. **Integrity:** Is the signature linked to the data signed in such a way that any subsequent change in the data is detectable?
4. **Authentication:** How confident are you that the signature is created under the sole control of the signer?

SIGNATURE TYPE	BASIC (BES)	ADVANCED (AES)	QUALIFIED (QES)
Definition	All electronic types of signatures that prove acceptance or approval by the signer by using some sort of certificate. This can be a signature manually drawn on a desktop screen (& digitally saved), a click on an "I accept" button, etc. This type is not a digital signature.	Advanced electronic signatures must meet specific requirements providing a higher level of signer ID verification, security, and tampersealing (meaning the document cannot be changed once it is signed).	Qualified electronic signatures or non-repudiation Digital Signatures are the only electronic signature type to have special legal status in EU. Unlike the other signatures, the burden of proof lies with the party that disputes the signature(s), not with the initiator. This makes it legally equivalent to a written signature. It is backed by a certificate issued by a Qualified Trust Service Provider (QTSP) that is on the EU Trust List (EUTL) and thus certified by an EU member state.
Authenticity	Not mandatory that the signature is linked to the signer.	Certain that the signature is uniquely linked to the signer.	Certain that the signature is uniquely linked to the signer.
Identity	Checking the identity of the signer is not mandatory.	Certain that the signature is uniquely linked to the signer.	100% Capable of identifying the signer. Initial face-to-face verification or another equivalent process is required.

SIGNATURE TYPE	BASIC (BES)	ADVANCED (AES)	QUALIFIED (QES)
Integrity Certain that content cannot be changed after signature?	Yes	Yes	Yes
Legal validity	The burden of proof lies with the party that initiated the signature.	The burden of proof lies with the party that initiated the signature.	This type is non-repudiative. The burden of proof lies with the party that disputes the signature.

Conclusion

In conclusion, all three categories are legally valid, but only **qualified digital signatures** are the legal equivalent to a written signature and lay the burden of proof with the party that disputes the signature. Qualified digital signatures are the most advanced and secure type of electronic signatures. They comply with the most demanding regulatory requirements as they provide the highest levels of assurance about each signer's identity and the authenticity/integrity of the documents they sign.

2.9.4 Editing documents

How do I edit a document I'm supposed to sign?

You don't. Editing would undermine the entire concept of secured digital signing. The contents of a document can't be modified or tampered with without breaking the validity of the digital signature.

Contact the initiator and ask them to send the correct document if the document you received doesn't contain the correct information.

To contact the initiator:

- Open the document you received for signing.
- Click **Reject**.
- Enter a reason for rejection. The initiator will be informed of this reason.

2.9.5 Sessions

Which sessions does eSignatures use?

In eSignatures you have the following sessions:

Login session

Description

The login session starts as soon as a user logs in to the WebPortal.

Its ends when the user logs out, or when the **SessionTimeout** limit has been reached.

Configuration

The **SessionTimeout** limit (expressed in minutes) can be configured in the Configuration Index, under **Environment Settings**.

The default value is 60 minutes. Note that changing this setting requires restarting the application.

Important: hosted customers are unable to change the Configuration Index settings and need to contact Connective in case changes are required.

WYSIWYS session

Description

The WYSIWYS session (What you see is what you sign) is started as soon as a user:

- Clicks their URL to open their document/package.
- Opens their document in the WebPortal.

The WYSIWYS sessions ends when the user correctly closes their document.

Configuration

There is no specific configuration for the WYSISYS session. However, if the user opens the document in the WebPortal and doesn't do anything for the time specified in the **SessionTimeout** limit, they will be logged out, and the WYSIWYS session will also end.

Signing session

Description

The signing session starts as soon as the user clicks **Start signing** and the Signing Modal (displayed below) is opened.



Sign manually

Signing field 1 of marker



Sign in the field below.

Retry Next

The Signing session ends when:

- The document has been signed.
- The user presses Cancel.
- The **LockTimeOutInMinutes** limit has been reached.

Note 1: if the Signing Modal is closed incorrectly, the document will remain locked until the **LockTimeOutInMinutes** limit has been reached. Requesting a new signing URL won't unlock the document before the **LockTimeOutInMinutes** limit has been reached.

Note 2: signing methods that make use of third parties, such as iDIN signing and OpenID connect, may also have time restrictions. These time restrictions are unknown and cannot be configured in eSignatures.

Configuration

The **LockTimeOutInMinutes** limit (expressed in minutes) can be configured in the Configuration Index, under **Environment Settings**.

Important: it is recommended to keep this value at 15 minutes. If you do choose to modify it, never set it to less than 5 minutes. Otherwise signing issues may arise, especially with large documents.

SMS code validity session

Description

The validity duration of an SMS code can also be considered a session and is a sublevel of the Signing session. It is determined by the **CodeTimeOut** parameter.

Configuration

The **CodeTimeOut** parameter can be configured in the Configuration Index, under **Signing Option Settings > SMS Signing Options**.

Mail OTP code validity session

Description

The validity duration of an Mail OTP code can also be considered a session and is a sublevel of the Signing session. It is determined by the **CodeTimeOut** parameter.

Configuration

The **CodeTimeOut** parameter can be configured in the Configuration Index, under **Signing Options Settings > Mail Signing Options**.

One-time URL session

Description

eSignatures makes use of one-time URLs. A one-time URL can be considered a session as too. As soon as a user has clicked their one-time URL, it becomes expired.

Requesting a new one-time URL can also be considered a new session.

2.10 WebPortal Troubleshooting

In this section you'll find the issues that WebPortal users might have concerning eSignatures. The issues are bundled per topic where possible.

2.10.1 My account

[I can't access my account](#)

[I can no longer log in to my account](#)

I can't access my account

POSSIBLE CAUSE	SOLUTION
No internet connection.	Check your internet connection. Internet access is required to access your eSignatures account.
You might not be using the correct password.	Click Forgot Password? and follow the on-screen instructions to reset your password.

If you still can't access your account after you've tried the steps above, contact your system administrator.

I can no longer log in to my account

POSSIBLE CAUSE	SOLUTION
<p>You tried to do an action you don't have the user rights for, or tried to access a part of the WebPortal you don't have access to.</p>	<p>When you try to do an action you don't have user rights for, or access a part you don't have access to, the system will log you out and you won't be able to log in again.</p> <p>To solve this issue, delete your eSignatures environment cookie from your web browser. You will now be able to log back in.</p> <p>Tip: in case you don't know which cookie to delete, delete all cookies from your web browser.</p>

2.10.2 My documents

I can't access my document

POSSIBLE CAUSE	SOLUTION
No internet connection.	An internet connection is required to access eSignatures. Make sure your internet connection is restored before you try to access your document again.
The document you're trying to access has been revoked by the initiator.	In this case you should have received an email informing you about the document being revoked. The only solution to access a document that has been revoked is to contact the initiator and ask them to send the document again.
You might have accidentally rejected the document you were supposed to sign.	The only solution to access a document that has been rejected is to contact the initiator, ask them to send the document again, and redo the signing process.
Another signer may have rejected the document you were both required to sign.	You can no longer sign the document.
The document you're trying to access may have already been signed.	You no longer need to sign the document.

2.10.3 eSignatures is running slow

POSSIBLE CAUSE	SOLUTION
A large number of documents are loaded into the system.	Delete old documents you no longer need.

2.10.4 Signing documents

I'm not mandated to sign

I'm unable to bulk sign

I'm unable to sign using biometric signing

I'm unable to sign with itsme

I'm not able to sign a PDF/A document

The legal notice is not displayed correctly on my biometric signature pad

I'm not mandated to sign

When trying to sign with eID, BeLawyer or itsme, I get the message that I'm not mandated to sign.

POSSIBLE CAUSE	SOLUTION
The information in your contact doesn't match the certificate info on the eID card, BeLawyer card or itsme account.	Make sure the info in your contact matches the certificate info on the eID card, BeLawyer card or itsme account. When the MandatedSigningType is set to nameandbirthdate by the administrator in the Configuration Index, you need to enter the given names and birthdate in the contact info. If necessary, contact your system administrator.

I'm unable to bulk sign

POSSIBLE CAUSE	SOLUTION
You're trying to sign too many documents at the same time.	A maximum of 20 documents/packages can be signed per signing session.
You're using a signing method that's not compatible with bulk signing.	Bulk signing can only be used when signing with an eID. Make sure eID is the selected signing method.
Mandated signing is enabled in the Configuration Index.	Ask your system administrator to disable mandated signing.
There is a legal notice on one of the documents.	Bulk signing is not compatible with legal notices. Contact the initiator and ask them to resend the document without legal notices.

I'm unable to sign using biometric signing

POSSIBLE CAUSE	SOLUTION
You're using an unsupported signature pad.	eSignatures currently supports Wacom STU-430, Wacom STU-530 and Wacom STU-540. Use one of these models.
You don't have the required Wacom SDK installed.	Contact Wacom to obtain the following SDK: wacom-signature-sdk-x86-3.19.2.msi (32-bit Operating System) or wacom-signature-sdk-x64-3.19.2 (64-bit Operating System)
You're using an unsupported Wacom Bamboo Stylus.	eSignatures currently supports Wacom Bamboo Fineline (CS-600) Important: the Wacom Bamboo Fineline can only be used on iPad.

I'm not able to sign with itsme

POSSIBLE CAUSE	SOLUTION
You're using an unsupported Operating System: macOS Mojave v10.14.	Use a supported Operating System .
You're using an unsupported browser: Safari v12.0.	Use a supported browser .

I'm not able to sign a PDF/A document

In the rare case this happens, contact your system administrator and ask them to create a ticket on our support website.

The legal notice is not displayed correctly on my biometric signature pad

POSSIBLE CAUSE	SOLUTION
The legal notice is too long.	The Wacom signature pads display only a limited number of characters on the signing screen. The STU-430 approximately displays 24 characters over 1 line of text, while the STU-530 displays approximately 74 characters over 2 lines. If the legal notice contains more characters, they fall off the screen. So if you want the entire legal notice to be displayed correctly on the signing screen, keep the legal notice limited to those numbers. You're recommended to run some tests beforehand to see if the text fits.

2.11 Video Tutorials

In this section you'll find the available video tutorials about eSignatures.

- [Basic navigation](#)
- [How to upload documents](#)
- [How to sign with sms](#)
- [How to sign face to face](#)

3. End users

The **End users** section is intended for end users who exclusively sign documents via email or a mobile app.

3.1 Signing step-by-step (End users)

In this section you'll learn how to sign step-by-step.

Note that it has no importance whether you're signing single documents or packages that contain multiple documents. The same signing steps apply.

eSignatures now supports **asynchronous signing**. This means you no longer need to wait until all documents are signed before you may close your signing session. This comes in handy especially when signing high volumes of (large) documents. As soon as you've started the actual signing, you may close the signing session and continue working on other things.

Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (through OpenID Connect) and Biometric.

Attention: when documents have been sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.

Signing step-by-step

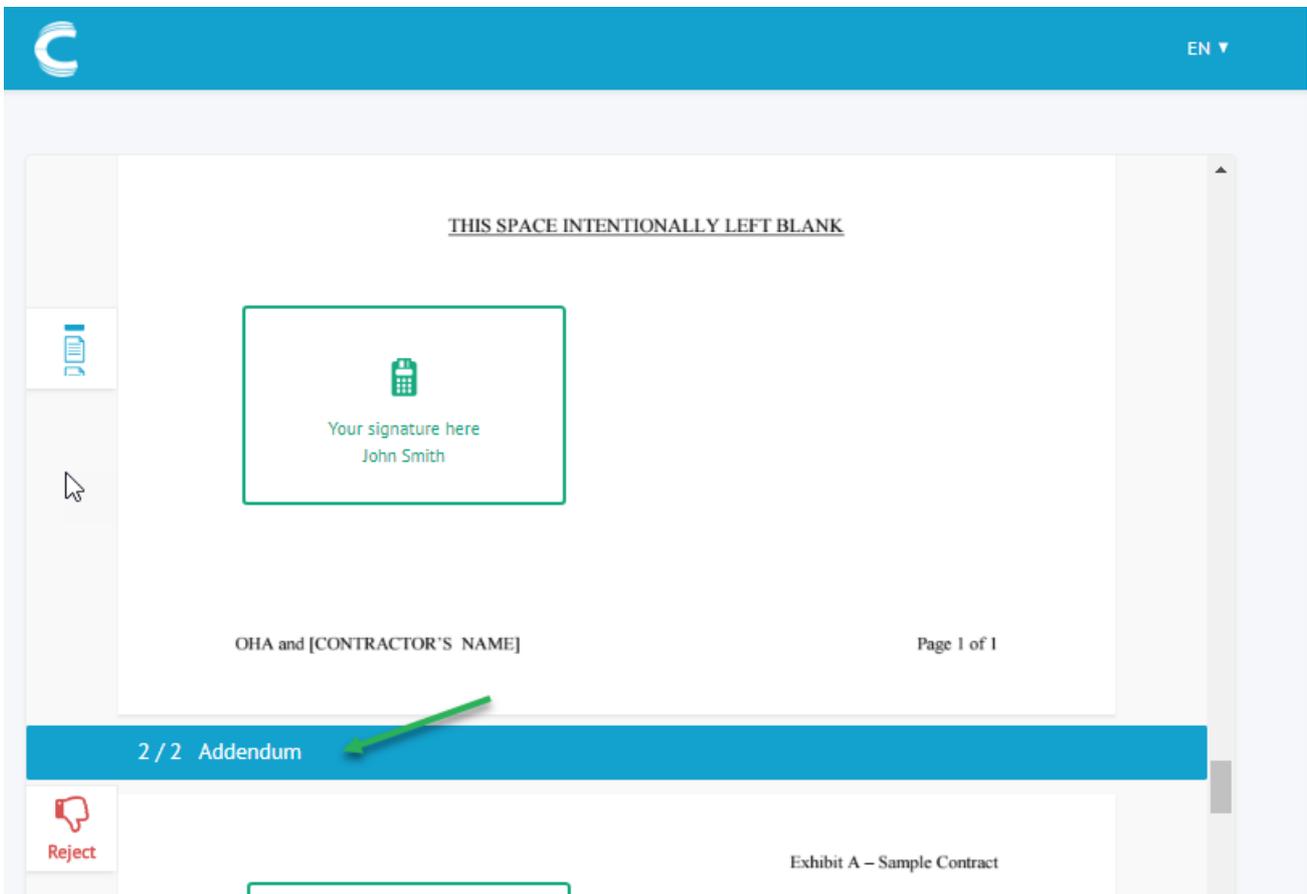
- Open the email you received from your eSignatures solution. In our standard demo environment you receive an email from **esigner@connective.be**.
- Click the secure link inside the email. The document will open in a new tab in your default web browser.

Important: this link will only work once. Once you've clicked it, you need to sign or reject the document. If the link expired, click **Request a new email** to receive a new link.

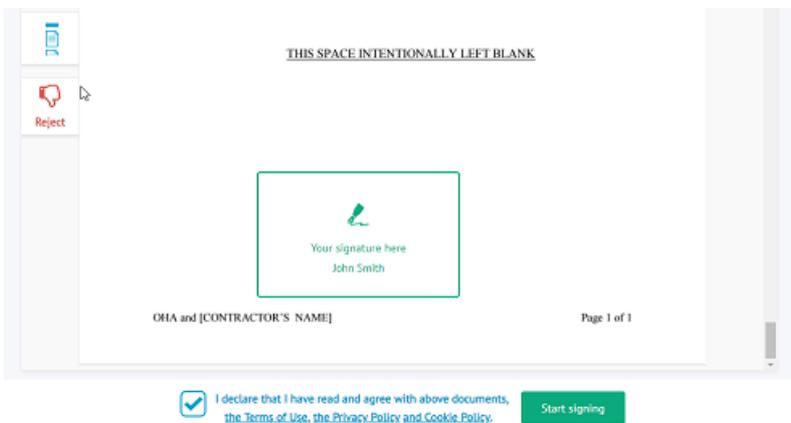
Please sign your document or package with name SAMPLE_CONTRACT Inbox x



- Read the entire document and scroll down to the last page. If your package contains multiple documents, the start of each document is indicated by a header. The header also indicates how many documents the package contains, and which document you are viewing.



- When you've scrolled down to the last page, the checkbox **I declare that I have read and agree with above documents, the Terms of Use, the Privacy Policy and Cookie Policy** now becomes available. Check it and then click **Start signing**.



Tip: click the links to consult the Terms of Use, Privacy Policy and Cookie Policy respectively.

Install the Connective browser package

Depending on the signing method that was defined for you, you might be prompted to install the Connective browser package. The Connective browser package is required on Windows and macOS when using any signing method that requires additional hardware:

- **eID signing:** requires an eID card reader
- **BeLawyer signing:** requires a transparent card reader
- **Biometric signing:** requires a biometric signing pad

✗ **The Connective Browser Package is not installed.**

This software is required for signing with a card reader or signing pad.



To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

Note: when using Internet Explorer, do *not* use Compatibility View. This is not supported by the Connective browser package.

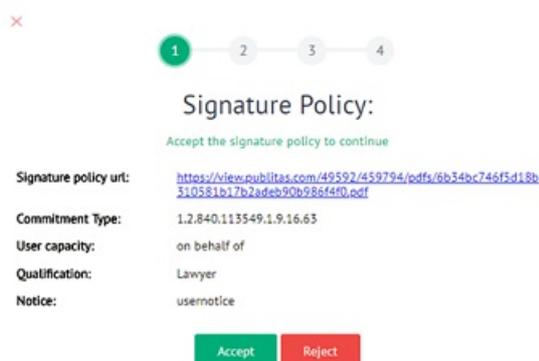
Choice of signing

- If multiple signing methods have been defined, you are prompted to **select a signing method**. Select your preferred signing method, and then click **Next**.
- The different signing methods are explained below.



Signature policy

- If your document contains a **signature policy**, you are prompted to accept it. A signature policy is a set of rules that details the terms and conditions of how a valid signature should be created and validated.
 - To check the signature policy before accepting it, click the link in the **Signature policy** screen.
 - If you agree with the signature policy, click **Accept**. If you reject, you won't be able to sign the document.



Legal notice

- If a legal notice was defined for your document, the **Legal notice** window opens.
- Type the content of the legal notice in the empty field. Copy-Paste is not supported.

Important: the legal notice you enter in the field must be exactly the same as the original legal notice, including spaces, cases, punctuation marks.

- Click **Next**.



Legal notice

Signing field 1 of SAMPLE_CONTRACT



Copy the legal notice below. (Note: the field is case sensitive.)

Read and approved

Next

Signing vs QuickSigning

- The signing window now opens, based on the signing method that was defined for you.
- If QuickSigning has been enabled, you'll be able to sign all signing fields inside the package using a single signature.



Quicksign manually

Sign in the field below.

Retry

Next

- If QuickSigning is not enabled, or the conditions are not met, you'll need to sign each signing field that is assigned to you.



Sign manually

Signing field 1 of marker



Sign in the field below.

Retry Next

Conditions for QuickSigning

- The following two signing methods cannot be combined with QuickSigning: sign manually+eID and biometric. If one of these signing methods has been selected for you, each field within the package must be signed individually.
- When using eID signing, a transparent eID card reader is required, i.e. a card reader without PIN pad. If you're using a PIN pad eID reader you'll be prompted to sign each field individually.
- If your package contains a legal notice, each field within the package must be signed individually.

Signing methods

One of the following signing methods may have been assigned to you. Click the links below for more information about each signing method.

Important: not all signing methods listed below may be available in your eSignatures solution, depending on the configuration.

Sign manually

- Use the mouse cursor to draw your signature in the signature field. To sign manually, a manual signature is needed as if signing a paper document.
- If you're not satisfied with the signature, click **Retry** to start over.
- When you are done, click **Next**.



Quicksign manually

Sign in the field below.

Retry Next

- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



Adding signatures to documents ...



We are adding signatures to your documents, you can close this window at any time

Close

- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign with eID

- Connect a [supported card reader](#) and insert your eID. The application now does all necessary checks.

Important notes:

Make sure the Connective browser package has been properly installed. Otherwise eID signing will not work. To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

To QuickSign using eID you need a transparent eID card reader, i.e. a card reader without PIN pad.

To use eID signing on an iOS or Android device, you need a rebranded [app](#) and use the Vasco 875 Bluetooth eID card reader. eID signing is not supported in a mobile web browser.



Sign with eID

Signing field 1 of marker



Connect a card reader and insert your card.

- When asked, enter your PIN. If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message appears on-screen.
- Click **Finish** to complete the process.

Sign manually + Sign with eID

- Use the mouse cursor to draw your signature in the signature field.
Tip: if you're not satisfied with the signature, click **Retry** to start over.
- When you're done, click **Next**.



Sign manually

Signing field 1 of marker



Sign in the field below.



- Connect a [supported card reader](#) and insert your eID. The application now does all necessary checks.



Sign with eID

Signing field 1 of marker



Connect a card reader and insert your card.

Important notes:

Make sure the Connective browser package has been properly installed. Otherwise eID signing will not work. To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

To QuickSign using eID you need a transparent eID card reader, i.e. a card reader without PIN pad.

To use eID signing on an iOS or Android device, you need a rebranded [app](#) and use the Vasco 875 Bluetooth eID card reader. eID signing is not supported in a mobile web browser.

- When asked, enter your PIN. If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message appears on-screen.
- Click **Finish** to complete the process.

Sign with SMS OTP

- Enter the 4 last digits of your phone number.

Important: if phone number confirmation is disabled in the Configuration Index, you won't need to confirm your phone number. You will receive the sms code directly.



Sign with one time SMS password

Signing field 1 of marker



Please fill in the final 4 digits of your phone number: +3247223

Next

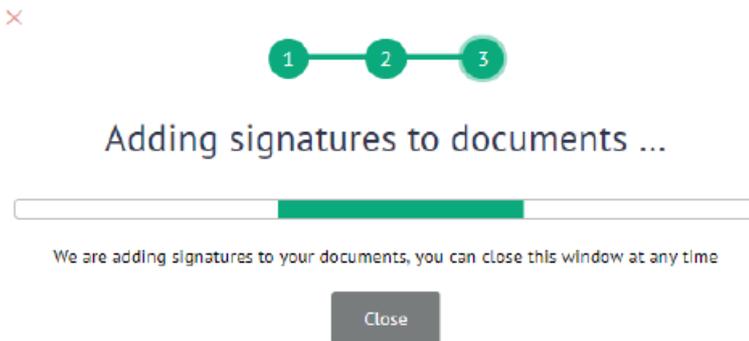
- Click **Next**. The password is sent by SMS to your phone.
- Enter the code you received and click **Next**.

Notes:

Make sure to enter the correct code. By default the number of attempts is limited to 10, but can be modified by an administrator in the Configuration Index.

The SMS code is valid for a limited amount of time. By default, this is 5 minutes, but can also be modified.

- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Tip: also check the [video tutorial](#) on sms signing:

Sign with email OTP

- Enter your full email address. A hint about the expected email address is shown between parentheses.

Important: if email address confirmation is disabled in the Configuration Index, you won't need to confirm your email address. You will receive the email code directly.



Sign with one time password via email

Signing field 1 of marker



Provide your **full email address** (documentatio*****@gmail.com).

Next

- Click **Next**. If you entered the correct email address, an email is sent to the email address you entered.
- Enter the code you received in the confirmation field.

Notes:

Make sure to enter the correct code. By default the number of attempts is limited to 10, but can be modified by an administrator in the Configuration Index.

The SMS code is valid for a limited amount of time. By default, this is 5 minutes, but can also be modified.

- Click **Next**.
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



Adding signatures to documents ...



We are adding signatures to your documents, you can close this window at any time

Close

- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign via signing pad (biometric)

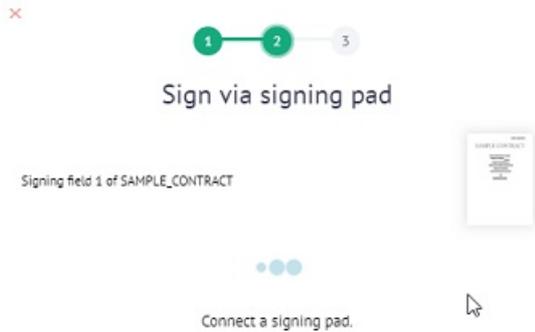
- Connect your signature pad to your computer using the provided USB cable.

Signing via signing pad requires a biometric signature pad. Such a signature pad allows to capture biometrical characteristics of your signature, like where the pen is located, when the pen tip is pressed down, and how hard it is pressed

down. These data are added to the signature, which would allow the signature pad manufacturer to verify the authenticity of the signature when required.

Important: eSignatures currently supports the following biometric signature pads: Wacom STU-430, Wacom STU-530 and Wacom STU-540. Make sure the necessary Wacom SDK is installed on your computer: wacom-signature-sdk-x86-3.19.2.msi (32-bit Operating System) or wacom-signature-sdk-x64-3.19.2.msi (64-bit Operating System).

Important: due to the technical setup of biometric signatures, each document within a package must be signed individually.



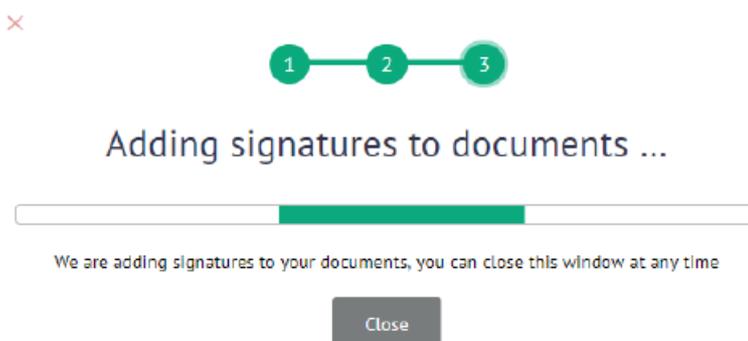
- When the device is successfully connected, "Please sign using your signature pad" appears on-screen.
- Draw your signature on the signature pad using the provided stylus.

To start over, tap **Clear**.

To cancel the operation, tap **Cancel**.



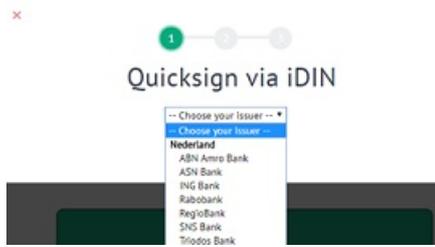
- When you're done, tap **OK** on the signature pad screen using the stylus.
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



- If asynchronous signing is not supported - because you will be redirected to another page after signing - the **Close** button will be unavailable. Please wait for the signing to finish to be redirected to the required page.

Sign with iDIN

iDIN signing allows users to sign using their Dutch bank card. The following issuers are currently supported: ABN Amro Bank, ASN Bank, ING Bank, Rabobank, RegioBank, SNS Bank, Triodos Bank.

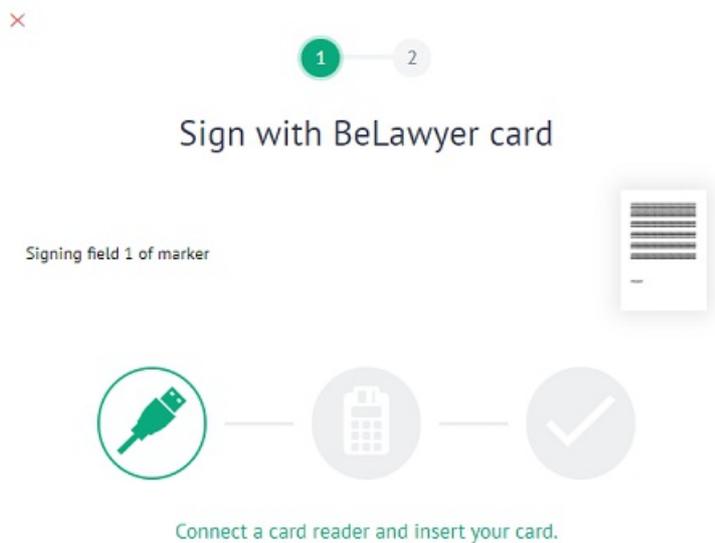


Sign with BeLawyer card

- Connect a [supported card reader](#) and insert your Belgian lawyer card. The application now does all necessary checks.

Important: Make sure the Connective browser package has been properly installed. Otherwise signing will not work. To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

- When asked, enter your PIN. If you're using a PIN pad card reader, press **OK** after entering the PIN. The document is now being signed. This may take a few seconds. When the document has been successfully signed, a confirmation message appears on-screen.



- Click **Finish** to complete the process

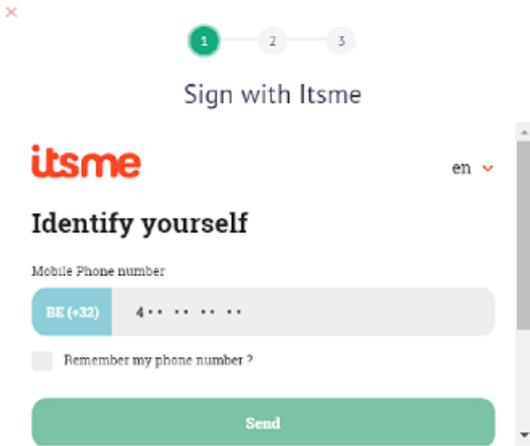
Sign with itsme

Attention:

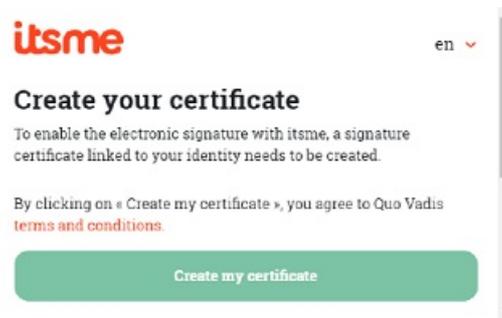
- Before you can sign with itsme, you must have [signed up for an ItsMe account](#) and the itsme app must be installed on your mobile phone.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

To sign with itsme:

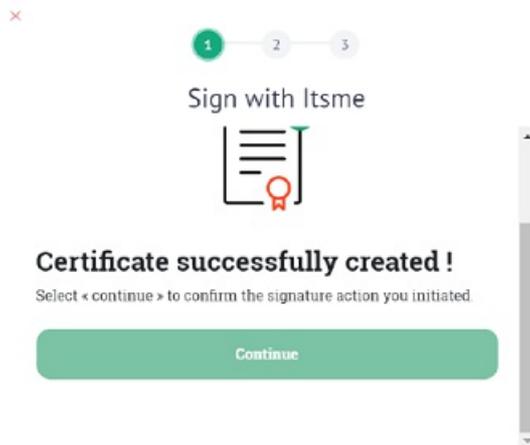
- Enter your **Mobile Phone number**, and click **Send**.



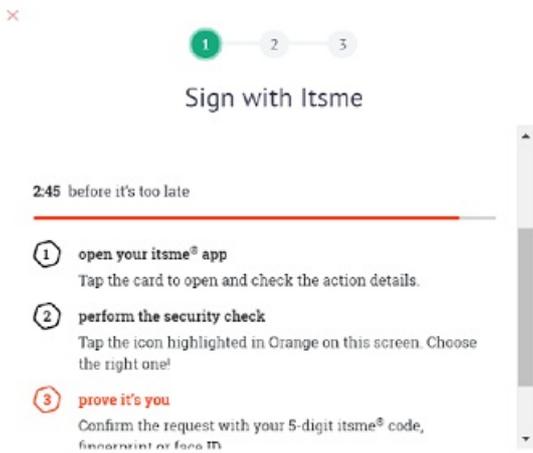
- When you're using itsme signing for the first time, you're prompted to create a certificate.
- Click **Create my certificate**. A message is now sent to your itsme app.



- Open your itsme app on your mobile phone, and click **Confirm**.
- Then enter your pincode, and click **OK**. The certificate will now be created.
- When the certificate has been created the following message appears in eSignatures.



- Click **Continue**.
- You're now prompted to return to your itsme app.
- Confirm your identity in the itsme app.



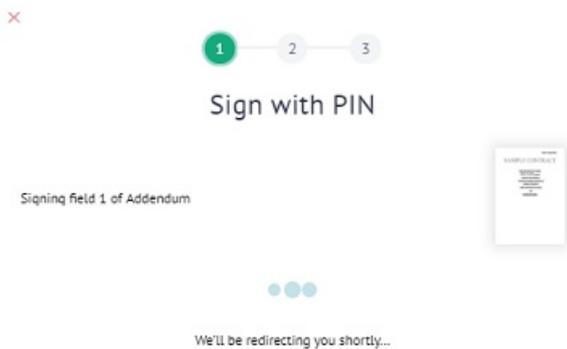
- Your document will now be signed with itsme.

Sign with custom signing type

A custom signing method - based on OpenID Connect - may have been configured that allows users to sign by pincode.

The name and icon of the signing type depends on the configuration done in the Configuration Index.

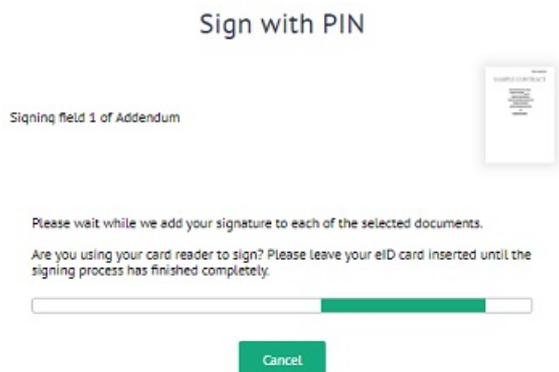
By way of example, we've named the custom signing method 'PINCODE'.



- Enter the pincode in the window that appears, and click **Validate**.

If you don't remember your pincode or haven't received one, contact your system administrator.

- Click **Yes** to confirm.
- Your document will now be signed.

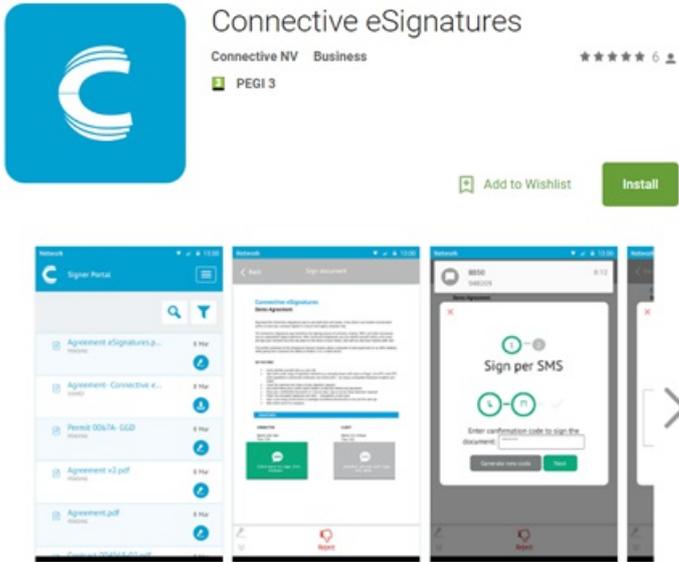


- When you are done, close the browser tab.

3.2 Signing in the Connective app

A demo **Connective app** for iOS and Android is available and can be used in combination with our standard demo environment of eSignatures. This app can be rebranded to fit your company's style.

In the Connective app you can sign documents in the same manner as on your computer. You also have an overview of all your documents and their status; so you can see which documents are pending, signed, rejected, expired and revoked. Note however that it is not possible to upload or delete documents via the app, or do advanced configurations. The app is solely meant for viewing and signing documents.



Note that you can also sign documents on a mobile device using a mobile web browser instead of the app. In that case however, signing methods that require additional hardware (eID signing, Biometric signing) are not supported. Also make sure you are using an [officially supported web browser](#).

3.2.1 How do I install the app?

Minimum System Requirements

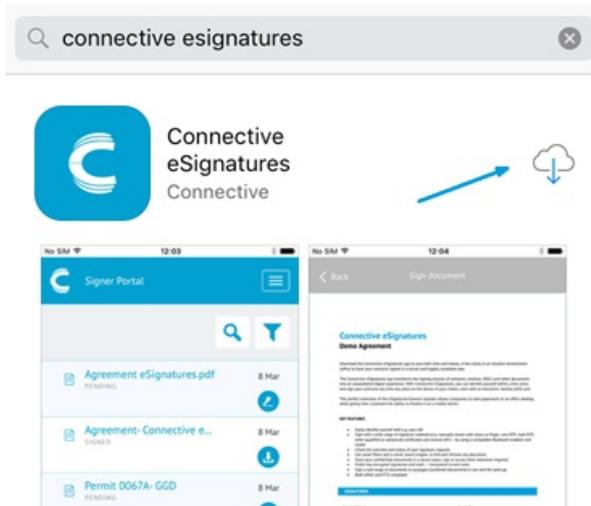
- Android 4.3 and up.
- iOS 9.3 or later. Compatible with iPhone, iPad and iPod Touch.

Installation on iOS (iPhone, iPad)

- Open the **App Store**.



- Tap **Search** at the bottom of the screen.
- Enter the app name in the Search field, and then tap **Search**. The name of our demo app is **Connective eSignatures**.
- Tap the download icon. Downloading the app takes a few seconds.



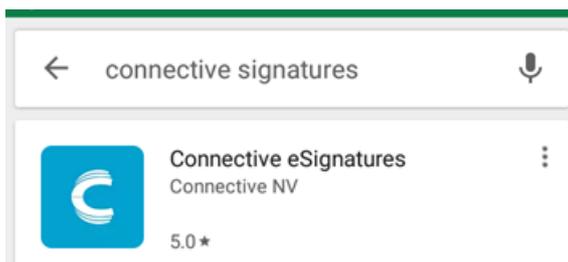
- When the download is complete, the **Open** button is displayed.
- Tap **Open** to open the app.



- Log in using the credentials you obtained from the administrator.

Installation on Android

- Open the **Play Store**.
- Type in the app name in the search field. The name of our demo app is **Connective eSignatures**.



- Select the app and then tap **Install**. Downloading the app takes a few seconds.



Connective eSignatures

Connective NV

 PEGI 3

INSTALL

- When the download is completed, the **Open** button is displayed.
- Tap **Open** to open the app.



Connective eSignatures

Connective NV

 PEGI 3

UNINSTALL

OPEN

- Log in using the credentials you obtained from your administrator.

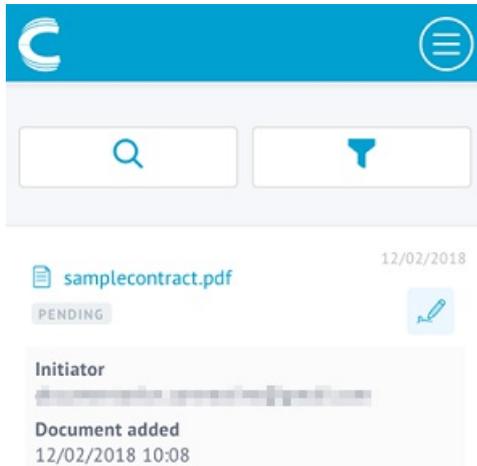
3.2.2 How do I sign in the Connective app?

1. Open the app

- Click the link in the email you received on behalf of the initiator. When you're prompted to open the app, tap **Open**.

2. Select a document to sign

- Tap the document you want to sign. Your document is opened in a new screen.



- Read the entire document and scroll down to the last page. The checkbox **I declare that I have read and agree with above documents, the Terms of Use, the Privacy Policy and Cookie Policy** now becomes available.. Check it and then click **Start signing**.
- The same signing steps apply like when signing a document on a computer. See [Signing a document step-by-step](#) for more info.

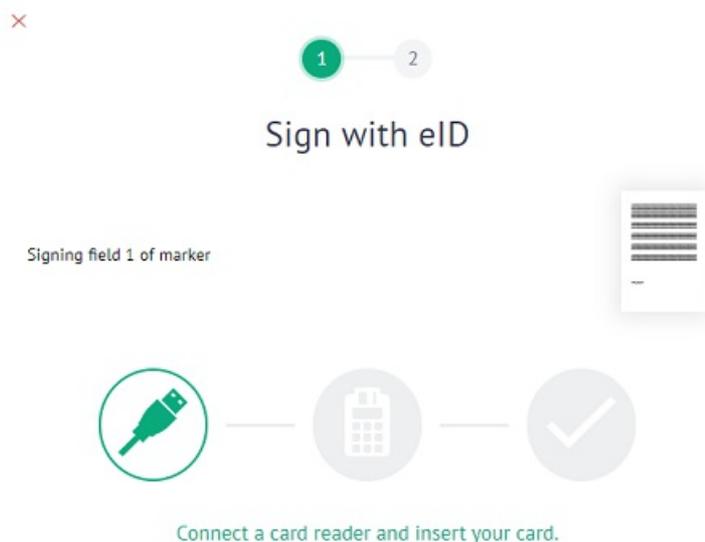
Note about signing with Belgian eID

To sign with eID you need the **Vasco Digipass 875** (a Bluetooth-enabled smart card reader).

Pairing the Vasco Digipass 875 to your mobile device

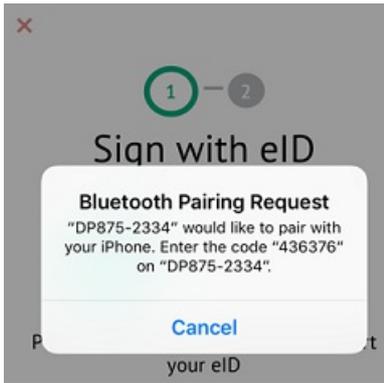
Before you can sign using such a card reader you need to pair it to your mobile device. This is a one-time action.

- Tap inside the signing field. The **Sign with eID** window now opens.



- Hold the Bluetooth card reader close to your mobile device and insert the eID card. If prompted to enable your location permissions, enable the location permissions on your mobile device.
- A message that Bluetooth is being connected should appear on the card reader screen.

- When the Bluetooth pairing request appears on your smartphone screen, enter the pairing code on the card reader and press **OK**.



- When the pairing is complete, you're prompted to enter the PIN code of your eID. Enter the PIN and press **OK**. Your document is now being signed. This may take some time, since all data is sent over Bluetooth.

3.2.3 How do I sign in a mobile web browser?

Signing documents in a mobile web browser only works if you haven't installed the Connective app yet.

To sign in a mobile web browser:

- Open your mail application on your iOS or Android device.
- Open the email you received on behalf of the initiator. In our standard demo solution you receive mails from: **esigner@connective.be**.
- Click the secure link inside the email.
- Now click **Continue to web signer** in the middle of the screen. Ignore the Connective App banner that might appear at the top of the screen.



- Your document now opens in your default mobile browser.
- The same signing steps apply like when signing a document on a computer. See [Signing a document step-by-step](#) for more info.

Attention: not all signing methods are supported in a mobile browser. If the signing method that was selected for you is not supported by your mobile browser, an error message will be displayed. If that happens, download the Connective app to sign your document or sign in on a computer.

3.3 End user FAQ

In this section we address the questions that end users might have concerning eSignatures.

The questions are bundled per topic where possible.

3.3.1 Signing documents

[How do I QuickSign a package?](#)

[Why do I need to install card reader software?](#)

[How do I install the card reader software?](#)

[Which smart card readers are supported?](#)

[Which biometric signature pads are supported?](#)

[Which operating systems are supported?](#)

[Which web browsers are supported?](#)

[How does asynchronous work?](#)

[What is the legal value of a digital signature?](#)

2.5.2 How do I QuickSign a package?

Signing a package functions in a similar way as signing a document. See [Signing a document step-by-step](#) for more information.

Why do I need to install card reader software?

Card reader software is required on Windows and macOS to use any signing method that requires additional hardware:

- **eID signing:** requires an eID card reader
- **BeLawyer signing:** requires a transparent card reader
- **Biometric signing:** requires a biometric signing pad

To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

If you're using an iOS or Android device and want to use a signing method that requires additional hardware, you need a mobile app. Contact your administrator to check if an app is available for your company. The Connective demo app in the App store and Google Play can only be used with the standard demo environment of eSignatures.

How do I install the card reader software?

To learn how to install the Connective browser package, see the [Browser package section](#) on the documentation website.

Which smart card readers are supported?

Windows and macOS

eSignatures supports the following brands of smart card readers on Windows and macOS:

CARD READER TYPE	EID SIGNING	BELAWYER SIGNING
Vasco Digipass 875	V	V
Vasco Digipass 870	V	V
Vasco Digipass 920	V	X
ACS ACR38	V	V
ACS AGP8202	V	V
Gemalto CT1100	V	V

Transparent readers (without PIN pad) from these brands should be supported too, provided they contain the correct drivers and the drivers have been installed on the user's computer.

Android and iOS

To sign with eID in a rebranded Connective app, the following smart card reader is required:

CARD READER TYPE	EID SIGNING
Vasco Digipass 875	V

Which biometric signature pads are supported?

The following signature pads are currently supported:

- Wacom STU-540
- Wacom STU-530
- Wacom STU-430

Make sure the necessary Wacom SDK is installed on your computer: [wacom-signature-sdk-x86-3.19.2.msi](#) (32-bit Operating System) or [wacom-signature-sdk-x64-3.19.2.msi](#) (64-bit Operating System).

Note: the Wacom Bamboo Fineline stylus is supported too as biometric signing method, but only on iPad.

Which operating systems are supported?

The following operating systems are supported:

Microsoft Windows

- Microsoft Windows 10 (64-bit)
- Microsoft Windows 8.1 (64-bit)
- Microsoft Windows 7 (64-bit)

Mac OS

- macOS Mojave
- macOS High Sierra
- macOS Sierra

Android

- Oreo 8.0-8.1
- Nougat 7.0-7.1.2
- Marshmallow 6.0-6.0.1

iOS

- iOS 11
- iOS 10
- iOS 9

Other operating systems are officially *not* supported.

Note: itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

Which web browsers are supported?

The following web browsers are supported:

Windows

- Microsoft Edge 44.17763 and 42.17134
- Internet Explorer 11

Earlier versions of Internet Explorer are not supported.

Note: do *not* use Compatibility View when using a signing method that requires the Connective browser package. This is not supported by the Connective browser package.

- Google Chrome n-2
- Mozilla Firefox n-2

Mac OS

- Safari n-2
- Google Chrome n-2
- Mozilla Firefox n-2

Android

- Google Chrome n-2

iOS

- Safari n-2

Other web browsers or browser versions are officially not supported.

Note that not all signing methods are supported in a mobile web browser. All signing methods that require additional hardware are only supported in the app.

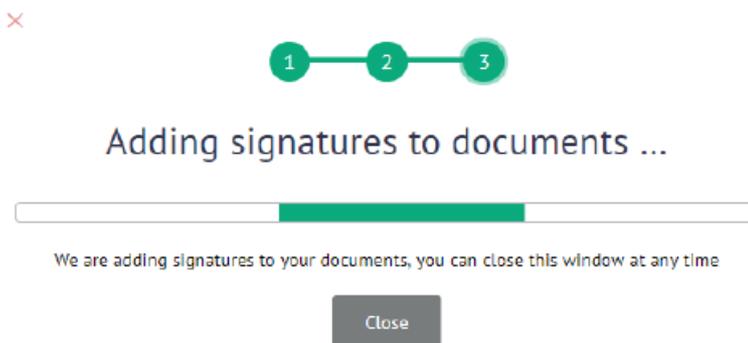
Note: itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

How does asynchronous signing work?

Asynchronous signing allows you to close the signing session and let the signing run in the background. This way, you can work on other things while your documents are being signed. This comes in handy especially when signing large documents that take quite some time to be signed.

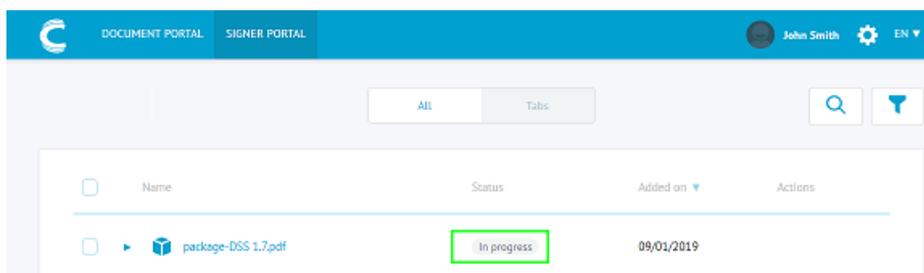
Asynchronous signing is available for the following signing types: manual, SMS OTP, Mail OTP, iDIN, pincode (though OpenID Connect) and Biometric.

- [Access the documents](#) that require your signature.
- Read the entire document and scroll down to the last page.
- When you've scrolled down to the last page, the checkbox **I declare that I have read and agree with above documents, the Terms of Use, the Privacy Policy and Cookie Policy** now becomes available. Check it and then click **Start signing**.
- Go through all the steps (if any) that require user action, such as entering a legal notice, accepting a signature policy, etc.
- The signing modal now opens. Place your signature, and then click **Next**. Note that the way of placing your signature varies depending on the signing type that was selected for you.



- You may click **Close** to the signing session, while the signing continues in the background. The status of the document is now changed to **In progress**. **Note:** the **In progress** state is only visible in the **Signer Portal**, not in the Document Portal.

Attention: when documents have been sent through the eSignatures API and a RedirectUrl has been configured for the signer, the signer will still have to wait until the signing has finished. The purpose of a RedirectUrl after all is that the signer is redirected to another URL after signing. Therefore, the Close button will be unavailable, and the signer will be informed they will be redirected.



- When the signing is completed, the status changes to **Signed**.

Note: if for some reason the asynchronous signing should fail, the document will get the status **Failed**. If this happens, end users should notify the initiator (the person who sent the document), who in turn should contact the system administrator to see what went wrong.

API users can also use the **Resubmit poison queue** call to resubmit the failed documents and retry the signing. In this case the end user won't have to resign the documents. Since they already placed their signature, the system will retry the signing automatically.

What is the legal value of a digital signature?

To answer that question, we first need to make sure we're talking about the same thing. After all, the terms '**electronic signature**' and '**digital signature**' are often used interchangeably. But there's a difference.

An **electronic signature** simply captures a person's intent to agree to the content of an electronic document or a set of data. It can be a signature manually drawn on a desktop screen, but also merely the image of your signature pasted in a Word document, or your mail signature. A **digital signature**, on the other hand, relies on a cryptography-based technology, which provides an extra level of security and integrity of the document.

So a digital signature is always an electronic signature while an electronic signature is not always a digital signature.

Under eIDAS - the European legislation that oversees electronic identification and trust services for electronic transactions - there are three categories of electronic signatures. All three categories can be legally effective. The difference between them is the evidence needed to reassure a court that the signature is genuine and intentionally applied to a particular document. When using an eIDAS compliant solution like eSignatures, the signatures are legally valid across European borders.

Below you'll find an overview of the three categories of electronic signatures, and their differences.

Categories of electronic signatures

1. Basic electronic signature (BES)
2. Advanced electronic signature or **digital signature** (AES)
3. Qualified advanced electronic signature or **Qualified digital signature** (QES)

As you can see, only categories 2 and 3 qualify as **digital signatures**.

The difference between these categories are mainly based on 4 key elements:

1. **Authenticity:** Is the signature uniquely linked to the signer?
2. **Identity:** Are you capable to identify the signer?
3. **Integrity:** Is the signature linked to the data signed in such a way that any subsequent change in the data is detectable?
4. **Authentication:** How confident are you that the signature is created under the sole control of the signer?

SIGNATURE TYPE	BASIC (BES)	ADVANCED (AES)	QUALIFIED (QES)
Definition	All electronic types of signatures that prove acceptance or approval by the signer by using some sort of certificate. This can be a signature manually drawn on a desktop screen (& digitally saved), a click on an "I accept" button, etc. This type is not a digital signature.	Advanced electronic signatures must meet specific requirements providing a higher level of signer ID verification, security, and tampersealing (meaning the document cannot be changed once it is signed).	Qualified electronic signatures or non-repudiation Digital Signatures are the only electronic signature type to have special legal status in EU. Unlike the other signatures, the burden of proof lies with the party that disputes the signature(s), not with the initiator. This makes it legally equivalent to a written signature. It is backed by a certificate issued by a Qualified Trust Service Provider (QTSP) that is on the EU Trust List (EUTL) and thus certified by an EU member state.
Authenticity	Not mandatory that the signature is linked to the signer.	Certain that the signature is uniquely linked to the signer.	Certain that the signature is uniquely linked to the signer.
Identity	Checking the identity of the signer is not mandatory.	Certain that the signature is uniquely linked to the signer.	100% Capable of identifying the signer. Initial face-to-face verification or another equivalent process is required.

SIGNATURE TYPE	BASIC (BES)	ADVANCED (AES)	QUALIFIED (QES)
Integrity Certain that content cannot be changed after signature?	Yes	Yes	Yes
Legal validity	The burden of proof lies with the party that initiated the signature.	The burden of proof lies with the party that initiated the signature.	This type is non-repudiative. The burden of proof lies with the party that disputes the signature.

Conclusion

In conclusion, all three categories are legally valid, but only **qualified digital signatures** are the legal equivalent to a written signature and lay the burden of proof with the party that disputes the signature. Qualified digital signatures are the most advanced and secure type of electronic signatures. They comply with the most demanding regulatory requirements as they provide the highest levels of assurance about each signer's identity and the authenticity/integrity of the documents they sign.

3.4 End user Troubleshooting

In this section you'll find the issues that end users might have concerning eSignatures.

The issues are bundled per topic where possible.

3.4.1 Emails

I don't receive any emails with documents to sign

I replied to an email sent by eSignatures but never received a response

I don't receive any emails with documents to sign

POSSIBLE CAUSE	SOLUTION
The emails end up in your Spam or Junk folder.	Check your Spam or Junk folder. If the emails sent by your eSignatures solution end up in your Spam or Junk folder, add the correct email address to your list of trusted email addresses. Contact your system administrator if necessary. In our standard demo solution, the email address to add is esigner@connective.be
The eSignatures solution doesn't have your correct email address.	Contact your system administrator and communicate your correct email address.

I replied to an email sent by eSignatures but never received a response

The eSignatures email address is used to send documents on behalf of the initiator, it may not be a correspondence address. This depends on the system configuration.

If you need to contact the person on whose behalf the email was sent, you need his personal email address.

I get an empty screen when trying to sign

POSSIBLE CAUSE	SOLUTION
You're not using an officially supported web browser.	Use an officially supported web browser .

I'm not mandated to sign

When trying to sign with eID, BeLawyer or itsme, I get the message that I'm not mandated to sign.

POSSIBLE CAUSE	SOLUTION
The information in your contact doesn't match the certificate info on the eID card, BeLawyer card or itsme account.	Make sure the info in your contact matches the certificate info on the eID card, BeLawyer card or itsme account. When the MandatedSigningType is set to nameandbirthdate by the administrator in the Configuration Index, you need to enter the given names and birthdate in the contact info. If necessary, contact your system administrator.

I'm not able to sign with itsme

POSSIBLE CAUSE	SOLUTION
You're using an unsupported Operating System: macOS Mojave v10.14.	Use a supported Operating System .
You're using an unsupported browser: Safari v12.0.	Use a supported browser .

I'm not able to sign a PDF/A document

In the rare case this happens, contact your system administrator and ask them to create a ticket on our support website.

The legal notice is not displayed correctly on my biometric signature pad

POSSIBLE CAUSE	SOLUTION
The legal notice is too long.	The Wacom signature pads display only a limited number of characters on the signing screen. The STU-430 approximately displays 24 characters over 1 line of text, while the STU-530 displays approximately 74 characters over 2 lines. If the legal notice contains more characters, they fall off the screen. So if you want the entire legal notice to be displayed correctly on the signing screen, keep the legal notice limited to those numbers. You're recommended to run some tests beforehand to see if the text fits.

API documentation

This section documents the technical specifications of eSignatures 5.2.7 API v3.

Revisions

DATE	OWNER	TOPIC
2018-07-11	DGI	Update to version 5.2 BETA
2018-11-19	DGI	Added examples of complex signing flow and document signing fields
2018-11-22	BJA	Added information about XML signing, signing type list
2018-12-06	DGI	Correction: renamed LawyerBe to BeLawyer
2018-12-06	DGI	New Set Process Information v3.1 call
2018-12-21	DGI	Corrections Get Enabled signing types call, Mandated signing on itsme
2019-01-07	DGI	Added info on phone number format
2019-01-09	DGI	Poison queue
2019-02-06	DGI	Revision
2019-02-13	DGI	Update to version 5.2.1
2019-02-26	TDE	Update to version 5.2.2
2019-03-07	BJE	Update to version 5.2.3
2019-03-19	DGI	Correction in 5.14.7
2019-03-26	DGI	Update to version 5.2.4
2019-03-27	DGI	Added Check version call + corrected units to px
2019-04-04	DGI	Update to version 5.2.5
2019-04-29	DGI	Update to version 5.2.6
2019-05-08	DGI	Added note on F2FRedirectUrl
2019-05-14	DGI	Update to version 5.2.7
2019-09-12	DGI	Correction on itsme limitations
2020-01-29	DGI	Error code description correction

1. Introduction

This document describes the technical specifications of Connective eSignatures version 5.2 API v3.

The eSignatures API is a REST API that allows external applications to integrate with and use the features listed in this document to create and manage signing flows.

1.1 Disclaimers

Only the described use cases are supported. All other use cases, even though possibly technically feasible with the API, are explicitly not supported and should not be implemented as such.

In case of discrepancies between examples and the description of the parameters section, the description of the parameters section prevails.

The descriptions of error codes in this document or the actual error message fields returned by the API are subject to change. External applications should rely on the returned error code values, anything else is only for diagnostic purposes.

All documents uploaded to the API must comply with the standards corresponding with that format. Conversions are based on a best-effort approach. The behavior of uploading non-compliant documents and / or conversions in an environment where required fonts, color profiles etc. are missing is undefined.

The URLs of the REST API in this document are changed to v3 since eSignatures version 5.0. The support period for the API with v2 URLs outlined in the eSignatures version 4.2 API Technical Documentation is governed by the Connective eSignatures support Contract.

Mixing v2 and v3 calls for the same document in the same environment is not supported. The API calls with v1 URLs are no longer supported since eSignatures version 5.0.

Be aware that the agile nature of JSON for the REST services supports adding optional parameters to the request or new parameters to the responses. The only actions that are considered breaking changes of the API are adding required parameters, changing existing parameters in the requests, or parameters missing from responses.

Optional fields that are not used must be left out of the request message. An empty string or dummy value for optional fields is also a value, and hence may trigger an error in the current version or a future version.

There is no dedicated error code for packages which exceed a given file size other than HTTP code 404.13 returned by IIS. However, as a practical guideline a document must not exceed 30 MB. Packages must not be larger than 150 MB in total and should not contain more than 15 documents. Note that large files might affect signing performance, depending on the user's Internet connections.

1.2 REST-service

The services are plain REST-based services, maintaining no state whatsoever. All data exchanges, in and out, are handled in the JSON data format and using the UTF-8 encoding.

The default URL is:

`https://[servername]:[port]/webportalapi/v3/`

Important notes:

Legacy Mode

If the API keeps returning HTTP 403 Forbidden errors, then the **Legacy Mode** setting might be enabled in the Configuration Index. This setting allows one-time URLs to be used indefinitely but will block API v3 calls (note that API v2 is never affected by this setting). Another explanation of HTTP 403 Forbidden errors might be invalid credentials.

Flows in Legacy Mode

When legacy mode is set to disabled in an environment, and you decide to enable it later on, the API flows that have been created when legacy mode was disabled will no longer work.

MTLS

Important: when the setting **IsMtlsEnabled** is enabled in the Configuration Index, API users need to be sure that an HTTP/1.1 connection is established and that the following header is sent in all eSignatures API calls:

```
Expect: 100-continue
```

The client should then wait for the HTTP 100 Continue response before sending the actual payload.

Failing to do so will cause problems with large payloads (e.g. when uploading PDF documents). In such a case the payload will fill the server's receiving buffer before proper mutual authentication is finished, the connection will never be established and the request will time out as a result.

If for some reason such HTTP/1.1 connection is not feasible, API users can also try to send a HEAD request using their HTTP client object before doing the actual file upload request with the same client. This workaround is not recommended however, since the client may decide to reset the connection in between the two different calls.

2. Authentication and Security

2.1 Authentication

The eSignatures API supports 'Basic Authentication' via a **username** and **password** combination that must be placed in the header of every request.

The default credentials, which have to be changed through configuration at installation time are:

PortalTA

ConnectiveXYZ123

Note: depending on the Configuration Index settings, the use of an Mtls Client certificate might be required.

2.2 Security through one-time URLs

Since eSignatures version 5.0 and higher, the URL lifetime is limited to one-time use. It is therefore necessary to always retrieve the newest URLs by doing an external API request just before redirecting the end user to their documents because once a URL has been used it becomes invalid. Only requesting a new one will give access to your documents.

When end users open a download URL they already opened, they will either see a warning or be redirected to a given URL (if available). The only way to proceed is to request a new URL and provide it to the user. This can be done by fetching all details using the API call in section [5.7 Get Package Status](#) or by using the API call in section [5.12 Send Package Reminders](#) which will send all signers an email containing the new URL. Making those calls multiple times may return the same URL each time if it has not been used yet.

When end users open a download URL they already opened, they will see a warning. On the same screen they can ask for an email to be sent to the original signer or receiver email address (if available) which will then contain a new URL.

Trying to download a document which has been deleted will always result in an authentication error. This cannot be recovered from as all information has been purged from eSignatures. It is therefore recommended to download documents as soon as possible to send them to an archiving system.

3. Quick Overview

With Connective eSignatures version 5.2 all signing flow creations can be done the same way, irrespective of the number of documents. Everything becomes a "package" of documents, even when some packages might contain just a single document.

Note: section [5.7 Instant Package Creation](#) has information about a call which can be used in common scenarios where a single document will suffice. This section gives an overview of how more complex scenarios can be handled.

□

The first call ([section 5.1](#)) creates a package with a unique id. This is a container for documents.

When adding documents ([section 5.2](#)), no signer information is sent but eSignatures will mark the places where the end user might want to sign. The eSignatures application will send back one unique id per location in the document together with a "label". The client application must then map these unique ids to the persons in charge of signing before making the next request below (the labels might help to find the right ones).

When all documents are added, the 'Set Process Information' request ([section 5.4](#)) will configure the signing flow:

- Signer details will be specified, including the unique id of the location(s) to sign
- Receiver information
- ...

When all information is present, the 'Set Package Status' request will make the package available for signing. Its return value contains all information needed to start signing.

4. Error Handling Responses

The eSignatures API v3 uses HTTP error codes to give a rough idea of whether a call succeeded or failed, and a system of error codes in the response body to give more information incase things went wrong.

The used error codes for each call are listed in that call's section. The meaning of each error code is further described in [section 11](#).

The error responses are JSON containing the following list of objects:

PARAMETER	CONTENT / DESCRIPTION	TYPE
Errors (array of objects)	List of the errors	Array
ErrorCode	Error code with variable information	String
Message	Error message detail text, not localized	String

The error code field usually contains information for an external system. Such information is separated by a colon (':') character. Everything *before* the colon character is of the form:

```
Group.Subgroup.Id
```

Everything *after* the colon character is variable and depends on the context of the error. If there are multiple values, then the first character will be a square opening bracket and the last character will be a square closing bracket.

Example response:

```
{
  "Errors": [
    {
      "ErrorCode": "Request.RequiredFieldIsMissing:DocumentName",
      "Message": "Required data field [DocumentName] is missing"
    },
    {
      "ErrorCode": "Request.RequiredFieldIsMissing:DocumentLanguage",
      "Message": "Required data field [DocumentLanguage] is missing"
    },
    {
      "ErrorCode": "Document.InvalidTargetFileType:[Pdf,PdFA1A,PdFA2A]",
      "Message": "Supported file types are [\"Pdf\", \"PdFA1A\", \"PdFA2A\"] but requested type was DocX"
    }
  ]
}
```

Error codes can be reused: if the HTTP response code is HTTP 400 Bad Request, an error code like "Document.InvalidTargetFileType" indicates a value that is not supported by eSignatures. If the HTTP response code is HTTP 409 Conflict, an error code like "Document.InvalidTargetFileType" means that the request value can currently not be used because the configuration forbids it.

Note: any HTTP 500 Server Error or other 50x responses might deviate from the format described in this section as they are not part of the API.

5. Available Package Services

eSignatures version 5.2 prefers to work with packages, which are containers for documents. They allow to show several documents to the end user and allow to place multiple signatures spread over those documents.

A newly created package will have the "Draft" status and will not be available for signing until all documents are added and the package status is changed to "Pending". Adding documents to a package can only be done when the package is still in the initial "Draft" state.

Notes:

- A package must not exceed 150 MB.
- A package must not contain more than 15 documents and each document must not exceed 30 MB.
- An .xml file must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance, depending on the user's Internet connection.

When all signers have reviewed and signed their documents, the package will have the "finished" status.

5.1 Create Package

5.1.1 Description

This call creates an empty package, allowing documents to be added to it.

Notes:

- A package must not exceed 150 MB.
- A package must not contain more than 15 documents and each document must not exceed 30 MB.
- An .xml file must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.

5.1.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages](https://[servername]:[port]/webportalapi/v3/packages)

5.1.3 HTTP Method

POST

5.1.4 MIME Type

application/json

5.1.5 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Initiator	Required	Email address of a registered user.	String
PackageName	Required	Package name, seen in the eSignatures Portal and when downloading as zip file. Note: do not add an extension to the PackageName value. Important: the Package name must not contain any of the following characters: slash, backslash, question mark, percent, asterisk, colon, pipe, double quote, less than, greater than.	String
CallbackUrl	Optional	REST API URL that will be called each time an action has been completed for this package, if no URL is supplied no call back is performed. See section 5.1.11 Package Callback Details below.	String
CorrelationId	Optional	Id that indicates which packages are correlated. The CorrelationId can be used in a GET call to retrieve the Audit proof file (a signed .xml file) about all correlated packages. See section 7. Audit proofs for more information about the audit proofs and the corresponding calls. Important: the Audit proofs setting must be enabled in the Configuration Index to use this parameter. If it is disabled, entering the CorrelationId will return an error.	String
DocumentGroupCode	Optional	The 'Code' which identifies a document group in which the package should be shown. Default is "00001" to signify "My Documents". See section 6.1: Get DocumentGroups .	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
ExpiryTimestamp	Optional	The date and time when this package expires and can no longer be signed. Documents in packages all use the value given here. Format is ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z	String with Date+Time +Offset
ExternalPackageReference	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String
ExternalPackageData	Optional	JSON data which could be used by customer-specific extensions.	String
F2FRedirectUrl	Optional	<p>URL to which the end user is redirected after all fields have been signed with 'face to face' signing. This field must be a valid absolute url.</p> <p>Attention: don't confuse the F2FRedirectUrl with the 'regular' RedirectUrl. The F2FRedirectUrl only applies to face to face signing. The RedirectUrl applies to regular signing and is set in the Set Process Information call. See section 5.4 Set Process Information > 5.4.14: Redirect URL Details for more info.</p> <p>Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a redirect url however is to redirect the signer to a new URL after the signing has finished. Therefore, when a F2FRedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.</p>	String
NotificationCallbackUrl	Optional	REST API URL that will be called when an end user requests to be notified. If no URL is supplied, no call back is performed. See section 5.1.12 Notification Callback Details below.	String

5.1.6 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId	Unique identifier of the package.	String
CreationTimestamp	Date and time when the package was created according to the server. Format is ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z	String

5.1.7 Example request

```
{
  "PackageName": "Contracts Mr. Doe",
  "Initiator": "info@mail.com",
  "DocumentGroupCode": "00001",
  "ExternalPackageReference": "2019-CR-5891"
}
```

5.1.8 Example response

```
{
  "PackageId": "25892e17-80f6-415f-9c65-7395632f0223",
  "CreationTimestamp": "2019-02-28T14:05:11+00:00"
}
```

5.1.9 Response codes

RESPONSE STATUS CODE	DESCRIPTION
201 Created	The package was created successfully. The URL for the new package is available in the Location header.
400 Bad request	The package could not be created due to invalid parameters in the request.
409 Conflict	The package could not be created due to an invalid document group code or unknown email address for the initiator.

5.1.10 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldsMissing
400	Package.ExpiryTimestampInvalid
400	Package.ExpiryTimestampInPast
400	Url.Invalid
409	DocumentGroup.NotFoundWithCode
409	User.NotFound

5.1.11 Package Callback Details

The **Callback URL** is used to contact external systems. When certain status changes happen, the given URL will be used to send an HTTP POST request, informing the external system that a change has taken place. The remote server is called only once, even when the server is offline or responds with a HTTP 500 error.

A callback may happen in the following cases:

- The package has its status changed to "Pending" through the eSignatures Portal
- The package is revoked through the eSignatures Portal
- One of the signers completed signing all their fields
- All signers have finished signing
- The package is rejected by one of the signers
- The package has its status changed to "Failed" through the eSignatures Portal

Note: API requests explained in this document will never trigger a callback. It is only when an end user triggers an action in the eSignatures Portal or signing screen that a callback will happen. Other triggers might be added in the future.

Currently the POST request sends an empty body with content type application/json. The external system must then fetch the current state of the package by inspecting the query parameters of the POST request and invoking the "Get Package Status" call from [section 5.8](#) with the provided package id.

The base URL (without parameters) is appended with the following query parameter:

- PackageId (the id of the package as returned by the Create Package call)

The callback url must contain a **valid, absolute URL with no parameters**.

For example, if the **CallbackUrl** parameter contains the following URL:

<https://myhost.example.org/services/callback>

then the effective callback URL will be:

<https://myhost.example.org/services/callback?PackageId=6aaa6664-22f8-4a4e-8b02-a1babd8eabb1>

Important: For performance and scalability purposes, a Callback timeout has been introduced and is set to **100 seconds**. This way, if the driving application doesn't respond to eSignatures' callback, a timeout will be forced, and the rest of the flow will be finished as if the expected **200 OK** message were received. If eSignatures were to wait indefinitely for a response to finish the package, its performance would drop drastically.

For this reason, it's highly recommended that the client's callback service is developed in such a way that it sends its response as soon as possible. Any other actions done by the callback service must not depend on the response being sent but should function asynchronously.

5.1.12 Notification Callback Details

The Notification Callback parameter, if specified, will in certain cases override the usual behavior of sending out emails and triggers a remote service instead. The remote service must then retrieve information about the package and choose what to do (see [section 5.8](#) for the Get Package Status call).

While the normal callback is called for major state changes, the notification callback can be called multiple times without any apparent change. For that reason, the callback includes information about the type of notification which was requested.

The remote server is called only once per user action; there is no retry unless the end user requests a new notification.

Note: this callback system may at one point be superseded by a more generic extension mechanism for notifications. At that point only the more generic mechanism will receive improvements.

Note: eSignatures waits for this remote service to complete its job before returning control to the end user. Therefore the remote service needs to give a response within seconds.

A callback consists of a POST request to the specified URL with content-type application/json. The following parameters are available in the JSON body:

PARAMETER	CONTENT / DESCRIPTION	TYPE
packageId	Unique identifier of the package	String
actorId	Identifier of the actor for which the notification was triggered	String
notificationType	Which kind of notification was requested	String
language	The language which the end user was told to use (see the Language parameter in the Stakeholder object of the Set Process Information call).	String

A callback may currently happen in the following cases:

(Note that the number in the left column is the id.)

NOTIFICATIONTYPEKEY	NOTIFICATIONTYPE	USE CASE
SendSignerUrl	0	The single-use signing link has been used already and the end user requested a new link.
SendDownloadUrl	1	The single-use download link has been used already and the end user requested a new link.

Note: since this list may grow in the future, the remote service will need to ignore other types without returning an error response.

Note: other use cases will not send a callback until explicitly listed in the table above. The existing emails sent when a package is created, revoked, etc. will depend on the SendNotifications parameter which is separate from the NotificationCallbackUrl.

Note: due to circumstances, the notification type ended up being a number rather than a string. The current numeric values will remain supported until eSignatures 6, but know that eSignatures 5.1 has included a new parameter NotificationTypeKey which contains a string-typed identifier.

5.2 Add document to package

5.2.1 Description

This call will add a document to an existing package.

Notes:

- A package must not exceed 150 MB.
- A package must not contain more than 15 documents and each document must not exceed 30 MB.
- An .xml must not contain more than 2 million characters per file. A package must not contain more than 15 .xml files.
- Large files might affect signing performance depending on the user's Internet connection.
- A PDF document's physical dimensions must not exceed 3.99 m by 3.99 m.

Signature field locations for PDFs can be set either using coordinates in the document or the id of an object in the document. See [section 10.2](#) for more info about such objects.

The supported upload document formats are docx, doc, pdf, plain text, and xml. **Note:** some of these formats might be disabled in the eSignatures configuration.

The response on this request will return a unique document GUID and unique ids for each of the proposed signing field locations.

Note: it is possible to add a document to a package where the "Set Process Information" call has already been run (see [section 5.4](#)). However, it is then necessary to run the "Set Process Information" call again before changing the package status to Pending.

Remarks:

- *Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A. When using itsme as signing method, it is mandatory to use PdfA1A or PdfA2A as TargetType. Note however that Connective does not perform any checks whether this TargetType has been selected.*
- *Rotated PDFs should not be used together with text markers. Detected signature locations will not be rotated to match the PDF text direction but will be placed near the text marker on a best-effort approach.*
- *When you upload PDF documents that contain Text Fields of which the name/id complies with the Text Field format you have configured in the Configuration Index, the Text Fields will be converted to empty signature fields in the output document and the original Text Field will not be displayed. This is intended behavior.*
Note: the remark above does not apply in case you upload a document that already contains one or more signatures – whether they have been created in eSignatures or another signing application.
- *When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the signature field names only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain text fields.*
- *Packages currently can't contain both XML documents and PDF documents on which signatures will be placed. The type of package is determined by the first uploaded document.*

5.2.2 URL

`https://[servername]:[port]/webportalapi/v3/packages/{id}/documents`

5.2.3 HTTP Method

POST

5.2.4 MIME Type (JSON + Base64)

application/json

5.2.5 MIME Type (Multipart)

multipart/form-data

This call expects the same input and will deliver the same output as the non-multipart version above, but the Document variable in the JSON must **not** contain a base-64 encoded pdf file. Instead the call will expect the document to be included as a different "part" of the request:

REQUEST ENTITY	CONTENT TYPE	DESCRIPTION
Data	application/json;charset=utf8	Json reques
Document	application/octet-stream	Document which needs to be signed
Representation	application/octet-stream	A PDF to be shown together with the document to be signed. See the Representation parameter below for its restrictions.

Note: eSignatures v5.1 and earlier looked for a part named 'pdf'. This is deprecated but still works.

5.2.6 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.2.7 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Document	Conditional	Attached document in base64 encoded format. <i>Required</i> unless Multipart format is used.	String
DocumentLanguage	Required	Language to use in signature texts. Currently supported: en, nl, de, fr, es. This is also the language that will be used for legal notices when LegalNoticeCode is filled for an Actor.	String
DocumentName	Required	Name of the document to be shown in the eSignatures Portal. Note: do not add an extension to the DocumentName value. Important: the Document name must not contain any of the following characters: slash, backslash, question mark, percent, asterisk, colon, pipe, quote, less than, greater than.	String
SigningFields	Required	See section 5.2.7.1 below.	Array of objects

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
CorrelationId	Conditional	<p>Id that indicates which documents within this package are correlated with documents that have been signed in the past in other packages.</p> <p>This CorrelationId can later be used to retrieve all Audit proofs related to this document across many packages. See section 7 for more information.</p> <p>Important: the CorrelationId value must be unique within the same Package.</p> <p>Important: The Audit proofs setting must be enabled in the eSignatures Configuration Index to use this parameter.</p>	String
DocumentType	Optional Required for XML signing	<p>Type of document that will be signed.</p> <p>Supported values: application/pdf (default) application/xml Word processing and text files are always converted to PDF.</p>	String
ExternalDocumentReference	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String
PdfErrorHandling	Optional	<p>How to deal with PDFs containing minor flaws. See section 4 below for more info. Values:</p> <p>Ignore DetectWarn DetectFail DetectFixWarn DetectFixFail</p>	Object
Representation	Optional Forbidden for PDF signing	Attached representation document in base64 format. This must be PDF data.	String
RepresentationType	Conditional	Type of the representation document. Must be present when Representation is filled. Only "application/pdf" is supported.	String
TargetType	Optional	<p>The TargetType defines if an extra conversion to PDF/A needs to be done before signing. Values: Pdf PdfA1A PdfA2A</p> <p>Notes: This will only work if the Document Conversion settings have been enabled in the Configuration Index. Existing signatures will be removed unless the PDF is of the specified type. When using itsme as signing method, it is mandatory to use PdfA1A or PdfA2A as TargetType. Note however that Connective does not perform any checks whether this TargetType has been selected.</p>	String

5.2.7.1 Signing Field Position

PDF signing

The location of each signature field on a PDF can be set by either using coordinates in the document (**PageNumber, Width, Height**, etc.) or by using the id of an object in the document (the **MarkerOrFieldId** parameter.) See [section 10](#) for more info.

When placing a field on a PDF it should at least be 112 px wide and 70 px high. **Note:** it is recommended to use slightly higher

values than the minimum ones. E.g. 75 px x 120 px. Using the absolute minimum values may lead to round-off errors during conversions.

When using the **MarkerOrFieldId** parameter, then all other parameters except **Label** are forbidden. The label will by default be generated from the text marker id part, text field id or signature field id. Specifying the **Label** parameter overrides the default label.

When you choose not to use the **MarkerOrFieldId** parameter, then you must use all coordinates parameters: **PageNumber, Width, Height, Left, Top**.

XML signing

XML signing does not allow for coordinates or markers to form a visual signature. Instead, all signature data will become part of the XML document and will be added at the end. This is determined by the XML signing specification eSignatures uses. It's the specification that determines how the signature is placed between the XML tags.

Note that all parameters except **Label** are forbidden. The **Label** parameter is required.

Note: every **Label** value should be unique within a document. This also implies that when a **MarkerOrFieldId** parameter is used without **Label** as override that this marker id should be unique in the document and not clash with other labels.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
SigningFields(array of objects)	Required (1-n)	One or more signing locations in the document	Array of objects
PageNumber	Conditional	Number of the page on which to add a signing location. First page is number 1. Zero (0) is not supported. Negative page numbers work as described in section 10.1.	Integer
Width	Conditional	Width Minimum value is 112. It is recommended to use slightly higher values than the minimum ones.	String
Height	Conditional	Height Minimum value is 70. It is recommended to use slightly higher values than the minimum ones.	String
Left	Conditional	Position from the left of the page. Minimum value is 1. Negative values are not supported.	String
Top	Conditional	Position from top of the page. Minimum value is 1. Negative values are not supported.	String
Label	Conditional	Text string which identifies this location for later use	String
MarkerOrFieldId	Conditional	Unique reference to a signing field, text marker or textfield. See section 10.2 for more details.	String

5.2.8 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
DocumentId	Unique id for the document	String

PARAMETER	CONTENT / DESCRIPTION	TYPE
CreationTimestamp	Date and time the flow was created. Format: YYYY-MM-DDThh:mm:ss+zz:zz	String
Locations	See table below	Array of objects

PARAMETER	CONTENT / DESCRIPTION	TYPE
Locations (array of objects)	Represents a possible location for a signature	Array of objects
Id	Unique id for this location	String
Label	Detected or specified label	String
PageNumber	The page on which the location was found. Numbering starts with 1 and the highest possible index is equal to the number of pages in the document.	Integer

5.2.9 Example Request

```
{
  "Document": "JVBERi...rest-of-the-document",
  "DocumentName": "Invoice",
  "DocumentLanguage": "nl",
  "ExternalDocumentReference": "INV-2019-04-01-0038",
  "SigningFields": [
    {
      "PageNumber": 1,
      "Width": "100",
      "Height": "200",
      "Left": "100",
      "Top": "200",
      "Label": "ThisIdentifiesJohnDoeHisSignatureLabel"
    }
  ]
}
```

5.2.10 Example Response

```
{
  "DocumentId": "e0cb4de4-673d-49fc-9bd1-7c81248984f9",
  "CreationTimestamp": "2019-03-28T08:54:38+00:00",
  "Locations": [
    {
      "Id": "8a96613f-b6ed-4227-9bde-c20d3ee0c9d6",
      "Label": "ThisIdentifiesJohnDoeHisSignatureLabel",
      "PageNumber": 1
    }
  ]
}
```

5.2.11 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The document was correctly added to the package. The response contains more information about the locations where a signer can place a signature.

RESPONSE STATUS CODE	DESCRIPTION
400 Bad Request	The request contained parameters which could not be accepted.
404 Not Found	The package id could not be found in the database.
409 Conflict	When certain document conversions are forbidden, when the input document has issues, or when marker ids are not matched.

5.2.12 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldsMissing
400	Document.InvalidTargetFileType
400	Document.NameInvalidLength
400	SigningField.MarkerAndCoordinatesCannotBeMixed
400	SigningField.MarkerNotUnique
400	SigningField.InvalidWidthCoordinate
400	SigningField.InvalidHeightCoordinate
400	SigningField.InvalidPage
400	Document.PasswordProtected
400	Pdf.UploadDoesNotComplyToSpec
400	PdfErrorHandling.InvalidType
400	Document.UnsupportedLanguage
409	Package.ApiVersionMismatch
409	Package.InvalidStatus
409	User.NotFound
409	Document.InvalidSourceFileType
409	Document.InvalidTargetFileType
409	SigningField.InvalidWidthMarker
409	SigningField.InvalidPage

HTTP CODE	CODE
409	SigningField.InvalidHeightMarker

5.2.13 PDF Error Handling Details

Some PDFs might have minor flaws which prohibit signing. Depending on the request parameters and the configuration settings, PDFs are either only checked or also modified to remove those flaws.

Note: The PDF will never be fixed if it already contains signatures, otherwise these signatures would become invalid. The presence of signatures and a PDF flaw might then trigger an error or warning depending on the choices below.

The **PdfErrorHandling** parameter defines the behavior, though the configuration settings might define the behavior if this parameter is not specified. Here are the different actions for the parameter:

- **Ignore**

Ignore means no checks or fixes will be done. Any document will be accepted but this might later be impossible to sign or result in a PDF with signature validation errors should a PDF flaw be present. This is the default value if this parameter is not specified and the eSignatures configuration has no different value.

- **DetectWarn**

When there is an issue, it will be detected and a warning is added to the eSignatures log file. The upload will still proceed. The upload will still proceed.

- **DetectFail**

When there is an issue, an error is added to the response and the upload is stopped.

- **DetectFixWarn**

When there is an issue, the system will detect and try to fix it. When it's not possible to fix it, a warning is added to the eSignatures log but the upload will still proceed.

- **DetectFixFail**

When there is an issue, the system will detect and try to fix it. When it's not possible to fix the document, an error is added to the response and the upload is blocked.

Note: these actions – 'Ignore' excluded – influence the speed of the system in different ways. See appendix II of the eSignatures Configuration Guide for an overview of the steps a document goes through when the other options are selected.

5.3 Get Signing Locations

5.3.1 Description

This call provides an overview of all signing locations inside the documents within a package. You can use this call in case the separate location lists returned by the different Add document calls were not stored for instance.

Note however that this call provides a subset of the responses that are returned by the Add document calls.

5.3.2 URL

https://[servername]:[port]/webportalapi/v3/packages/{id}/locations

5.3.3 HTTP Method

GET

5.3.4 MIME Type

Not applicable

5.3.5 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package. Use the value you receive as PackageId parameter in 5.1.6 .	String

5.3.6 Response parameters

The root level has currently only one parameter: 'Documents'.

PARAMETER	CONTENT / DESCRIPTION	TYPE
Documents	Zero or more document objects	Array of objects

PARAMETER	CONTENT / DESCRIPTION	TYPE
Documents (array of objects)	Zero or more document objects	Array of objects
DocumentId	Unique id for the document	String
ExternalDocumentReference	External reference for identification. Make sure to use a unique value.	String
Locations	List of detected or created signature locations	Array of objects

PARAMETER	CONTENT / DESCRIPTION	TYPE
Documents --> Locations (object)	Possible locations for signatures.	Object
Id	Unique id for this location, to be used when referring to it later.	String
Label	Detected or specified label.	String
PageNumber	The page on which the location was found. Numbering starts with 1 and the last index is equal to the number of pages.	Integer

5.3.7 Example Response

```

{
  "Documents": [
    {
      "DocumentId": "7c0af947-e3db-417a-900a-c25852be3d97",
      "ExternalDocumentReference": "INV-2019-04-23-0037",
      "Locations": [
        {
          "Id": "ae554ac8-bacc-4e8a-81a1-46af780142ea",
          "Label": "SIG01",
          "PageNumber": 1
        }
      ]
    },
    {
      "DocumentId": "176232c3-c97b-4b4a-91e3-c2f347c92e9f",
      "ExternalDocumentReference": "INV-2019-04-23-0038",
      "Locations": [
        {
          "Id": "2dd4ed18-ce80-49a8-aa75-f177045b2488",
          "Label": "SIG02",
          "PageNumber": 1
        }
      ]
    }
  ]
}

```

5.3.8 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The signing locations from the package are returned successfully.
404 Not Found	The package Id given does not exist.
409 Conflict	The package with the specified id was made with an old version of the api.

5.3.9 Error codes

HTTP CODE	CODE
404	Package.NotFound
409	Package.ApiVersionMismatch

5.4 Set Process Information

5.4.1 Description

This webservice method updates the people involved in the process (stakeholders) and assigns them steps which need to be taken.

Important: an API v3.1 version of this call is also available. See section 5.5 for more information. Any new Set Process Information features that are added to the API in the future will be introduced on that API v3.1 call while the call in this section will not change.

Note: all stakeholders must be specified each time this call is made, otherwise they will get deleted.

5.4.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/process](https://[servername]:[port]/webportalapi/v3/packages/{id}/process)

5.4.3 HTTP Method

PUT

5.4.4 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.4.5 Request parameters

The root level has currently only one parameter: 'Stakeholders'.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders	Required (1-n)	Information about the people who are involved with this package.	Array of objects
Stakeholder (array of objects)	Required (1-n)	Information about the people who are involved this package.	Array of objects
Actors	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
EmailAddress	Required	Email address	String
FirstName	Required	First name	String
Language	Required	UI language of this stakeholder. Currently supported: en, nl, de, fr, es.	String
LastName	Required	Last name	String
BirthDate	Conditional	Date of birth in format: YYYY-MM-DD Note: activating mandated signer validation in the API or configuration might make this required. See section 5.4.12 below.	String
ExternalStakeholderReference	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE

5.4.5.1 Actor

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders --> Actors (array of objects)	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
Type	Required	Signer Receiver Note: actor of type Signer will become a receiver as well	String
OrderIndex	Required for Signer	This number specifies in which order actors need to execute their action. If this number is the same for all actors, then the order in which they sign doesn't matter; they sign in parallel. Incrementing numbers indicate a sequential signing flow: the actor with the lowest OrderIndex value must sign first, the one with the second lowest must sign second and so on. You can also design a complex signing flow: assign the same OrderIndex value to multiple actors who may sign in parallel and assign a different OrderIndex value to actors who must sign before or after the parallel signing.	Int
LocationIds	Required for Signer	The location ids where a signature must be placed by this person.	Array of strings
SigningTypes	Required for Signer	One or more signing type info objects. See section 8.	Array of objects
Phonenumber	Optional Only Signer	Phone number to receive an SMS OTP. Note: always add the country code in front of the phone number. E.g. +32xxxxxxxx. It is recommended to use the plus sign as international dialing prefix instead of using "00". Important: never use spaces in the phone number format.	String
RedirectURL	Optional Only Signer Required if IsBackButtonEnabled is set to false in the Configuration Index.	Url to which the end user is redirected after signing. This field must be a valid absolute url. See section 5.4.14 Redirect Details below. Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a RedirectUrl however is to redirect the signer to a new URL after the signing has finished. Therefore, when a RedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.	String
SendNotifications	Optional Only Signer	eSignatures can send e-mails to the actors whose action is required, such as signing. Such notifications can be enabled or suppressed by setting this parameter as 'true' or 'false' (the default is 'false'). This parameter is set per actor.	Boolean
UserRoles	Conditional Only Signer Forbidden for XML signing	Information about the signer's function. This field must match the language used in the documents to be legally valid. Can currently only be passed when signature policy is used, as seen in section 5.4.13 below.	Array of strings

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
LegalNoticeCode	Optional Only Signer Forbidden for XML signing	LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3 The 3 values will point to the 3 legal notices built into the application. These can be altered in the Configuration Index. The language in which the legal notice is displayed, depends on the DocumentLanguage	String
LegalNoticeText	Optional Only Signer Forbidden for XML signing.	Custom legal notice text in case none of the three predefined legal notices apply. The text should be written in the same language as the one used in the documents of the package.	String

5.4.5.2 Signing Type Specific Information

The following object defines what kinds of signing types are allowed and it can define extra validation steps for that signing type.

Note: when any of the optional parameters are specified, all signing type objects need to contain the same values for those parameters. This is especially important for **MandatedSignerValidation**. When you are using choice of signing (i.e. when the signer may choose from different signing types), and you set the **MandatedSignerValidation** parameter to **MatchId** or **NameAndBirthDate** for one signing type, you must also set that parameter to the same value for all the other signing types for that actor, even though those signing types may not support **MandatedSignerValidation**.

This restriction will be solved in a future version of eSignatures, making all kinds of combinations possible.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders --> Actors --> SigningTypes (array of objects)	Required (1-n)	This object specifies the signing type and its related properties.	Array of objects
SigningType	Required	The signing type. See section 9 .	String
CommitmentTypes	Conditional Forbidden for XML signing	One or more OIDs of commitment types. Can only be passed when signature policy is used. See section 5.4.13 below.	String
MandatedSignerValidation	Optional	Type of validation to execute during eID or other smart card signing session. See section 5.4.12 below. Values: Disabled NameAndBirthDate MatchId	String
MandatedSignerIds	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See section 5.4.12. Required when MandatedSignerValidation is of type MatchId .	Array of strings
SignaturePolicyId	Optional	Signing policy details for the signature. See section 5.4.13 below.	String

5.4.6 Example request 1

In this example request, one actor named John Doe will sign one document inside a package, choosing Beld or BeLawyer as signing type.

```
{
  "Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john.doe@example.org",
      "Language": "en",
      "BirthDate": "1972-09-24",
      "ExternalStakeholderReference": "C0004105",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 1,
          "LocationIds": [
            "68f35693-5530-4770-b8d8-76284719e524", "c2e325a4-7b1d-42a6-8179-5377707d007c"
          ],
          "SigningTypes": [
            {
              "SigningType": "BeId",
              "MandatedSignerValidation": "MatchId",
              "MandatedSignerIds": [
                "72092400465", "72092630155"
              ]
            },
            {
              "SigningType": "BeLawyer",
              "MandatedSignerValidation": "MatchId",
              "MandatedSignerIds": [
                "83fc726f-9e4a-486f-9d99-87c6604bde7d", "7ab8594b-4cd0-4c7e-862e-3cc226622149"
              ]
            }
          ],
          "PhoneNumber": "+32477123456",
          "UserRoles": [
            "Lawyer"
          ],
          "SendNotifications": true,
          "RedirectUrl": "https://www.mycompany.com"
        }
      ]
    }
  ]
}
```

5.4.7 Example request 2

This example request displays a complex signing use case: John Doe will use Beld to sign his recruitment contract at MyCompany. Once he has signed, Jane Jefferson (the CEO of MyCompany) and Bob Smith (the HR Officer of MyCompany) must countersign the contract. The **OrderIndex** value determines John Doe must sign first; he has the lowest **OrderIndex** value. Then, the two other actors must sign. The order in which they do has no importance – since the **OrderIndex** value is the same for both, but higher than the one for John Doe.

```
{
  .."Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john.doe@example.org",
      "Language": "en",
      "SendNotifications": true,
      "ExternalStakeholderReference": "Employee"
```

```

ExternalStakeholderReference : Employee ;
"Actors": [
  {
    "Type": "Signer",
    "OrderIndex": 1,
    "LocationIds": [
      "68f35693-5530-4770-b8d8-76284719e524"
    ],
    "SigningTypes": [
      {
        "SigningType": "BeId",
        "MandatedSignerValidation": "Disabled"
      }
    ]
  }
],
},
{
  "FirstName": "Jane",
  "LastName": "Jefferson",
  "EmailAddress": "jane.jefferson@mycompany.org",
  "Language": "en",
  "SendNotifications": false,
  "ExternalStakeholderReference": "CEO",
  "Actors": [
    {
      "Type": "Signer",
      "OrderIndex": 2,
      "LocationIds": [
        "76284719e524-5530-4770-b8d8-68f35693"
      ],
      "SigningTypes": [
        {
          "SigningType": "BeId",
          "MandatedSignerValidation": "Disabled"
        }
      ]
    }
  ]
},
{
  "FirstName": "Bob",
  "LastName": "Smith",
  "EmailAddress": "bob.smith@mycompany.org",
  "Language": "en",
  "SendNotifications": true,
  "ExternalStakeholderReference": "HR Officer",
  "Actors": [
    {
      "Type": "Signer",
      "OrderIndex": 2,
      "LocationIds": [
        "55304486e524-5530-4770-b8d8-68f35748"
      ],
      "SigningTypes": [
        {
          "SigningType": "BeId",
          "MandatedSignerValidation": "Disabled"
        }
      ]
    }
  ]
}
}

```

5.4.8 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
None	Empty object	

5.4.9 Example response

```
{ }
```

5.4.10 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The process information has successfully been added to the package.
400 Bad Request	There are invalid parameters in the request.
404 Not Found	When an unknown package id is passed.
409 Conflict	Some parameters conflict with the data found in the database or configuration.

5.4.11 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldIsMissing
400	Stakeholder.BirthDayInFuture
400	Stakeholder.UnsupportedLanguage
400	SigningField.InvalidPage
400	SigningType.Invalid
400	Stakeholder.EmailAddressInvalid
400	SignaturePolicy.MissingUserRole
400	SignaturePolicy.MissingCommitmentType
400	Signer.ParameterCannotBeUsedWithoutSignaturePolicy
400	Actor.TypeInvalid
400	Actor.MissingPhoneNumber
409	MandatedSigner.BirthDateMissing
409	MandatedSigner.MandatedSignerIdMissing
409	SignaturePolicy.NotFound
409	CommitmentType.NotAllowed

HHTP CODE	CODE
409	PhoneNumber.Invalid
409	Location.NotFound
409	SignaturePolicy.ShouldBeSameForActor
409	CommitmentType.ShouldBeSameForActor
409	MandatedSigner.MultipleValidationTypesNotAllowed
409	SigningType.Disabled

5.4.12 Mandated signer validation

Smartcards (or secure elements in general) hold a signing certificate which is directly tied to the end user. These certificates may have a unique serial number as part of the certificate's subject information which can then be checked by the eSignatures solution to see if the end user is mandated to sign during a particular session. Any other card or token will show a "You are not mandated to sign" warning during signing.

The extra validation can be enabled by passing the **MandatedSignerValidation** parameter value "MatchId" or through settings in the configuration. If the configuration was changed to do the mandated signer validation check by default, then the validation can still be disabled for a single API request by passing "Disabled" as value of the **MandatedSignerValidation** parameter.

Two kinds of validation are possible:

- MatchId

If this kind of validation is enabled then one or more ids need to be passed in the **MandatedSignerIds** parameter. This way only certificates can be used whose Subject information contains one of the allowed ids.

- NameAndBirthDate

This validation type is useful when the full national registry number of a Belgian eID holder is not known but their name and birthdate is. This check will try to match up the stakeholder's last name, first name and birthdate with the certificate to find a match. In this case the **MandatedSignerIds** parameter is not necessary.

Important: When you are using choice of signing (i.e. when the signer may choose from different signing types), and you set the **MandatedSignerValidation** parameter to **MatchId** or **NameAndBirthDate** for one signing type, you must also set that parameter to the same value for all the other signing types for that actor, even though those signing types may not support **MandatedSignerValidation**.

This restriction will be solved in a future version of eSignatures, making all kinds of combinations possible.

Note: the API accepts numeric values ("0" = Disabled, "1" = NameAndBirthDate, "2" = MatchId) but this is deprecated and will be removed in API v4 / eSignatures 6. It is recommended to use text ids.

5.4.13 Signature Policies and Commitment Types

Digital signatures can reference a signing policy which details the terms and conditions of how a valid signature should be created and validated.

Since such a policy needs to be drafted before it can be used, these policies are specified using a single Id which needs to be configured in the eSignatures database. Please ask the Connective customer services team for more information when a policy is required.

Commitment types are limited to the following set of 6 types defined in the ETSI CADES standards:

TYPE OID	DESCRIPTION
1.2.840.113549.1.9.16.6.1	Proof of origin
1.2.840.113549.1.9.16.6.2	Proof of receipt
1.2.840.113549.1.9.16.6.3	Proof of delivery
1.2.840.113549.1.9.16.6.4	Proof of sender
1.2.840.113549.1.9.16.6.5	Proof of approval
1.2.840.113549.1.9.16.6.6	Proof of creation

5.4.14 Redirect URL Details

The **redirect url** is used to redirect the end user to the originating web application pointed to by that URL. If there is no URL, the end user will have to close the current tab by clicking the Close Tab button of the browser.

The base URL (without any parameters) must be given in the Set Process Information call and is appended with the following query parameters:

- SessionID (unique identifier of the package signing session)
- ExternalReference (as found in the Stakeholder object of the Set Process information call)
- Status (SIGNED, REJECTED or INVALIDTOKEN)
- PackageExternalReference (as found in the ExternalPackageReference parameter of the Create Package call/Create Instant Package call)

The **RedirectUrl** parameter must contain a **valid, absolute URL with no existing query parameters**.

For example, if the **RedirectUrl** parameter contains the following redirect url:

<https://myserver.example.org/rental/services/testredirect>

then the effective redirect URL will be:

<https://myserver.example.org/rental/services/testredirect?SessionID=5a2aff04-3cfa-4278-9480-64ac39f74734&ExternalReference=user1@example.org&Status=REJECTED&PackageExternalReference=dossier-3592>

Note: the end user can **counterfeit** the redirect URL because the redirection happens *client-side*! Either establish a second secure channel by means of the Callback URL, or verify through session state that the end user returned from the correct session id. Afterwards a call to Get Package Status (see [section 5.8](#)) must be done to verify that the document was actually signed or rejected.

Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a RedirectUrl however is to redirect the signer to a new URL after the signing has finished. Therefore, when a RedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.

5.5 Set Process Information (v3.1)

5.5.1 Description

This webservice method updates the people involved in the process (stakeholders) and assigns them steps which need to be taken.

The API v3.1 version of this call allows you to set multiple legal notices within a single signing session. You can, for instance, set a certain legal notice on one location (i.e. document), and a different legal notice on another location. Or you can set a general legal notice on Actor level which will be applied to all locations where that actor must sign, unless a different legal notice is set for specific locations.

Important: any new Set Process Information features that are added to the API in the future will be introduced on this API v3.1 call. This is the only call which has **v3.1** in the URL, all others will keep using **v3**.

Note: all stakeholders must be specified each time this call is made, otherwise they will get deleted.

5.5.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3.1/packages/{id}/process](https://[servername]:[port]/webportalapi/v3.1/packages/{id}/process)

5.5.3 HTTP Method

PUT

5.5.4 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.5.5 Request parameters

The root level has currently only one parameter: 'Stakeholders'.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders	Required (1-n)	Information about the people who are involved with this package.	Array of objects
PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholder (array of objects)	Required (1-n)	Information about the people who are involved with this package.	Array of objects
Actors	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
EmailAddress	Required	Email address	String
FirstName	Required	First name	String
Language	Required	UI language of this stakeholder. Currently supported: en, nl, de, fr, es.	String
LastName	Required	Last name	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
BirthDate	Conditional	Date of birth in format: YYYY-MM-DD Note: activating mandated signer validation in the API or configuration might make this required. See section 5.4.12.	
ExternalStakeholderReference	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

5.5.5.1 Actor

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders --> Actors (array of objects)	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
Type	Required	Signer Receiver Note: actor of type Signer will become a receiver as well	String
OrderIndex	Required for Signer	This number specifies in which order actors need to execute their action. If this number is the same for all actors, then the order in which they sign doesn't matter; they sign in parallel. Incrementing numbers indicate a sequential signing flow: the actor with the lowest OrderIndex value must sign first, the one with the second lowest must sign second and so on. You can also design a complex signing flow: assign the same OrderIndex value to multiple actors who may sign in parallel and assign a different OrderIndex value to actors who must sign before or after the parallel signing.	Int
Locations	Required for Signer	The locations where a signature must be placed by this person.	Array of string
SigningTypes	Required for Signer	One or more signing type info objects. See section 9.	Array of objects
Phonenumber	Optional Only Signer	Phone number to receive an SMS OTP. Note: always add the country code in front of the phone number. E.g. +32xxxxxxx. It is recommended to use the plus sign as international dialing prefix instead of using "00". Important: never use spaces in the phone number format.	String
RedirectURL	Optional Only Signer Required if IsBackButtonEnabled is set to false in the Configuration Index.	Url to which the end user is redirected after signing. This field must be a valid absolute url. See section 5.4.14 Redirect Details below. Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a RedirectUrl however is to redirect the signer to a new URL after the signing has finished. Therefore, when a RedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.	String
SendNotifications	Optional Only Signer	eSignatures can send e-mails to all the people who can sign. Such notifications can be enabled or suppressed by setting this parameter as 'true' or 'false' (the default is 'false').	Boolean

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
UserRoles	Conditional Only Signer Forbidden for XML signing	Information about the signer's function. This field must match the language used in the documents to be legally valid. Can currently only be passed when signature policy is used, as seen in section 5.4.13.	Array of strings
LegalNoticeCode	Optional Only Signer Forbidden for XML signing	LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3 The 3 values will point to the 3 legal notices built into the application. These can be altered in the Configuration Index. The language in which the legal notice is displayed depends on the DocumentLanguage . Note: a LegalNoticeCode or LegalNoticeText set on Location level takes precedence over this value.	String
LegalNoticeText	Optional Only Signer Forbidden for XML signing.	Custom legal notice text in case none of the three predefined legal notices apply. The text should be written in the same language as the one used in the documents of the package. Note: a LegalNoticeCode or LegalNoticeText set on Location level takes precedence over this value.	String

5.5.5.2 Location

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders => Actors => Locations (array of objects)	Required (1 n)	This object specifies the locations where a signature must be placed.	Array of objects
Id	Required	The id of the location where a signature must be placed by this person.	String
LegalNoticeCode	Optional	LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3 The 3 values will point to the 3 legal notices built into the application. These can be altered in the Configuration Index.	String
LegalNoticeText	Optional	Custom legal notice text in case none of the three predefined legal notices apply. The text should be written in the same language as the one used in the documents of the package.	String

5.5.5.3 Signing Type Specific Information

The following object defines what kinds of signing types are allowed and it can define extra validation steps for that signing type.

Note: when any of the optional parameters are specified, all signing type objects need to contain the same values for those parameters. This is especially important for **MandatedSignerValidation**. When you are using choice of signing (i.e. when the signer may choose from different signing types), and you set the **MandatedSignerValidation** parameter to **MatchId** or **NameAndBirthDate** for one signing type, you must also set that parameter to the same value for all the other signing types for that actor, even though those signing types may not support **MandatedSignerValidation**.

This restriction will be solved in a future version of eSignatures, making all kinds of combinations possible.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders --> Actors --> SigningTypes (array of objects)	Required (1-n)	This object specifies the signing type and its related properties.	Array of objects
SigningType	Required	The signing type. See section 9 .	String
CommitmentTypes	Conditional Forbidden for XML signing	One or more OIDs of commitment types. Can only be passed when signature policy is used. See section 5.4.13.	Array of strings
MandatedSignerValidation	Optional	Type of validation to execute during eID or other smart card signing session. See section 5.4.12. Values: Disabled NameAndBirthDate MatchId	String
MandatedSignerIds	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See section 5.4.12. Required when mandated signer validation is of type MatchId.	Array of strings
SignaturePolicyId	Optional	Signing policy details for the signature. See section 5.4.13.	String

5.5.6 Example request 1

In this example request, one actor named John Doe will sign on two locations (i.e. 2 documents) using Beld or Manual as signing type.

For location 1, a specific legal notice is defined which takes precedence over the legal notice set on Actor level. For location 2, no specific legal notice is defined, and the legal notice set on Actor level will be used. Note that the legal notice set on Location level will take precedence over the legal notice set on Actor level.

```

{
  "Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": " john.doe@example.org ",
      "Language": "en",
      "BirthDate": "1972-09-24",
      "ExternalStakeholderReference": "C0004105",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 2,
          "Locations": [
            {
              "Id": "68f35693-5530-4770-b8d8-76284719e524",
              "LegalNoticeCode": "LEGALNOTICE1"
            },
            {
              "Id": "2e000002-fae3-46e1-ba01-5b592bf8dc32",
            }
          ],
          "LegalNoticeCode": "LEGALNOTICE2"
        },
        {
          "SigningTypes": [
            {
              "SigningType": "beid",
              "MandatedSignerValidation": "MatchId",
              "MandatedSignerIds": [
                "72092400412", "72031516425"
              ]
            },
            {
              "SigningType": "manual",
            }
          ],
          "SendNotifications": true,
        }
      ]
    }
  ]
}

```

5.5.7 Example request 2

This example request displays a complex signing use case: John Doe will use Beld to sign his recruitment contract at MyCompany. Once he has signed, Jane Jefferson (the CEO of MyCompany) and Bob Smith (the HR Officer of MyCompany) must countersign the contract. The **OrderIndex** value determines John Doe must sign first; he has the lowest **OrderIndex** value. Then, the two other actors must sign. The order in which they do has no importance – since the **OrderIndex** value is the same for both, but higher than the one for John Doe.

```

{
  .."Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john.doe@example.org",
      "Language": "en",
      "SendNotifications": false,
      "ExternalStakeholderReference": "Employee",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 1,
          "Locations": [
            {

```

```

        "Id": "68f35693-5530-4770-b8d8-76284719e524",
    },
    {
        "Id": "2e000002-fae3-46e1-ba01-5b592bf8dc32",
    }
],
"SigningTypes": [
    {
        "SigningType": "BeId",
        "MandatedSignerValidation": "Disabled"
    }
]
}
]
},
{
    "FirstName": "Jane",
    "LastName": "Jefferson",
    "EmailAddress": "jane.jefferson@mycompany.org",
    "Language": "en",
    "SendNotifications": true,
    "ExternalStakeholderReference": "CEO",
    "Actors": [
        {
            "Type": "Signer",
            "OrderIndex": 2,
            "Locations": [
                {
                    "Id": "76284719e524-5530-4770-b8d8-68f35693",
                }
            ],
            "SigningTypes": [
                {
                    "SigningType": "BeId",
                    "MandatedSignerValidation": "Disabled"
                }
            ]
        }
    ]
},
{
    "FirstName": "Bob",
    "LastName": "Smith",
    "EmailAddress": "bob.smith@mycompany.org",
    "Language": "en",
    "SendNotifications": true,
    "ExternalStakeholderReference": "HR Officer",
    "Actors": [
        {
            "Type": "Signer",
            "OrderIndex": 2,
            "Locations": [
                {
                    "Id": "55304486e524-5530-4770-b8d8-68f35748",
                }
            ],
            "SigningTypes": [
                {
                    "SigningType": "BeId",
                    "MandatedSignerValidation": "Disabled"
                }
            ]
        }
    ]
}
]
}

```

5.5.8 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
None	Empty object	

5.5.9 Example response

```
{ }
```

5.5.10 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The process information has successfully been added to the package.
400 Bad Request	There are invalid parameters in the request.
404 Not Found	When an unknown package id is passed.
409 Conflict	Some parameters conflict with the data found in the database or configuration.

5.5.11 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldIsMissing
400	Stakeholder.BirthDayInFuture
400	Stakeholder.UnsupportedLanguage
400	SigningField.InvalidPage
400	SigningType.Invalid
400	Stakeholder.EmailAddressInvalid
400	SignaturePolicy.MissingUserRole
400	SignaturePolicy.MissingCommitmentType
400	Signer.ParameterCannotBeUsedWithoutSignaturePolicy
400	Actor.TypeInvalid
400	Actor.MissingPhoneNumber
409	MandatedSigner.BirthDateMissing
409	MandatedSigner.MandatedSignerIdMissing
409	SignaturePolicy.NotFound
409	CommitmentType.NotAllowed

HHTP CODE	CODE
409	PhoneNumber.Invalid
409	Location.NotFound
409	SignaturePolicy.ShouldBeSameForActor
409	CommitmentType.ShouldBeSameForActor
409	MandatedSigner.MultipleValidationTypesNotAllowed
409	SigningType.Disabled

5.5.12 Mandated signer validation

These parameters haven't changed. See section 5.4.12.

5.5.13 Signature Policies and Commitment Types

These parameters haven't changed. See section 5.4.13.

5.5.14 Redirect URL Details

These parameters haven't changed. See section 5.4.14.

5.6 Set Package Status (Pending / Revoked)

5.6.1 Description

Once the package is created and filled with documents, the status needs to be changed to "Pending". This will make the package visible for each of the stakeholders in their Signer Portal.

If the status was changed to Pending but the package needs to be deleted, then the status must be changed to Revoked first so that it is declared unavailable for signing. If the package was created with the **SendNotifications** parameter set to true, then all signers will get a notification that they can no longer sign the specified package.

5.6.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/status](https://[servername]:[port]/webportalapi/v3/packages/{id}/status)

5.6.3 HTTP Method

PUT

5.6.4 MIME Type

application/json

5.6.5 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.6.6 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Status	Required	Pending Revoked	String

5.6.7 Response parameters

The response is currently the same as that of the Get Package Status call. See [section 5.8](#).

5.6.8 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The package was successfully changed to the new status.
400 Bad Request	When invalid parameters are passed.
404 Not Found	When an unknown package id is passed.
409 Conflict	When the package doesn't contain any documents or the status doesn't allow revocation.

5.6.9 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldsMissing

HTTP CODE	CODE
403	User.PermissionDenied
404	Package.NotFound
409	Package.InvalidStatus
409	Package.ApiVersionMismatch
409	Package.ContainsNoPackageSigners
409	Package.ContainsNoDocuments
409	Package.ContainsDocumentWithLocationsWithNoSigners
409	Package.ContainsNoDocumentSigners
409	Package.ContainsDocumentWithNoSigners

5.7 Instant Package Creation

5.7.1 Description

This call creates a package with a single document in it and instantly prepares it for signing.

The response is **nearly** the same as [Get Package Status](#) (with the addition of the package id), saving an extra call.

Notes:

- A document must not exceed 30 MB.
- A document's physical dimensions must not exceed 3.99 m by 3.99 m.
- An .xml document must not contain more than 2 million characters per file.
- Large files might affect signing performance depending on the user's internet connection.

Signature field locations can be set either using coordinates in the document or the id of an object in the document. See [section 10](#) for more info.

The supported upload document formats are docx, doc, pdf, plain text, and xml.

Note: some of these formats might be disabled in the eSignatures configuration.

Remarks:

- *Uploading PDF/A documents is only allowed if the format is PDF/A_2A or PDF/A_1A. When using itsme as signing method, it is mandatory to use PdfA1A or PdfA2A as TargetType. Note however that Connective does not perform any checks whether this TargetType has been selected.*
- *Rotated PDFs should not be used together with text markers. Detected signing fields will not be rotated to match the PDF text direction but will be placed near the text marker on a best-effort approach.*
- *When you upload PDF documents that contain Text Fields of which the name/id complies with the Text Field format you have configured in the Configuration Index, the Text Fields will be converted to empty signature fields in the output document and the original Text Field will not be displayed. This is intended behavior.*

Note: in case you upload a document that already contains one or more signatures – whether they have been created in eSignatures or another signing application – the remark above does not apply.

- *When uploading PDF documents that already contain signature fields - which you created in a PDF solution such as Adobe Acrobat Pro DC - make sure the signature field names only contain letters and numbers, or a combination of both. Any special characters such as accented letters, slashes, dots, etc. are not supported and must not be used. The same limitations apply when uploading PDF documents that contain text fields.*

5.7.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/instant](https://[servername]:[port]/webportalapi/v3/packages/instant)

5.7.3 HTTP Method

POST

5.7.4 MIME Type (JSON + Base64)

application/json

5.7.5 MIME Type (Multipart)

multipart/form-data

This call expects the same input and will deliver the same output as the non-multipart version above, but the Document or Representation parameter in the JSON must **not** contain base-64 encoded file data. Instead the call will expect the document to

be included as a different "part" of the request.

REQUEST ENTITY	CONTENT TYPE	DESCRIPTION
Data	application/json; charset=utf 8	Json request
Document	application/octet-stream	Document that needs to be signed
Representation	application/octet-stream	A PDF to be shown together with the document to be signed. See the Representation parameter below for its restrictions.

Note: eSignatures v5.1 and earlier looked for a part named 'pdf'. This is deprecated but still works.

5.7.6 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Document	Conditional	Attached document in base64 encoded format. <i>Required</i> unless Multipart format is used.	String
DocumentLanguage	Required	Language to use in signature texts. Currently supported: en, nl, de, fr, es. This is also the language that will be used for legal notices when LegalNoticeCode is filled.	String
DocumentName	Required	Name of the document and package to be shown in the eSignatures Portal. Note: do not add an extension to the DocumentName value. Important: the name must not contain any of the following characters: slash, backslash, question mark, percent, asterisk, colon, pipe, double quote, less than, greater than.	String
Initiator	Required	Email address of a registered user	String
Stakeholders	Conditional	Information about the people who need to sign this package. See section 5.7.6.1. below Normally <i>required</i> , but using a template will make this parameter <i>forbidden</i> .	Array of objects
CallbackUrl	Optional	REST API URL that will be called each time an action has been completed for this package, if no URL is supplied no call back is performed. See section 5.1.11 Package Callback Details .	String
CorrelationId	Optional	Id that indicates which packages or documents are correlated. The CorrelationId can later be used to retrieve all related Audit proofs. See section 7 for more info. In this request the parameter will be used as packaged <i>and</i> document correlation id. Important: the Audit proofs setting must be enabled in the Configuration Index to use this parameter.	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
DocumentGroupCode	Optional	The 'Code' which identifies a document group in which the document should be shown. Default is "00001" to signify "My Documents". See section 6.1: Get DocumentGroups .	String
ExpiryTimestamp	Optional	The date and time when this package expires and can no longer be signed. Note that this must be an ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z	Date+Time+Offset
ExternalDocumentReference	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal. Make sure to use a unique value.	String
ExternalPackageReference	Optional	Reference given by the calling application, this parameter will not be used by the eSignatures Portal. Make sure to use a unique value.	String
ExternalPackageData	Optional	JSON data which could be used by customer-specific extensions.	String
F2FRedirectUrl	Optional	Url to which the end user is redirected after all fields have been signed with 'face to face' signing. This field must be a valid absolute url. Attention: don't confuse the F2FRedirectUrl with the 'regular' RedirectUrl. The F2FRedirectUrl only applies to face to face signing. The RedirectUrl applies to regular signing and is set in the Set Process Information call. See section 5.4.14: Redirect URL Details for more info. Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a redirect url however is to redirect the signer to a new url after the signing has finished. Therefore, when a F2FRedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.	String
NotificationCallbackUrl	Optional	REST API URL that will be called when an end user requests to be notified. If no URL is supplied no call back is performed. See section 5.1.12 Notification Callback Details .	String
PdfErrorHandling	Optional	How to deal with PDFs containing minor flaws. See section 4 for more info. Values: Ignore DetectWarn DetectFail DetectFixWarn DetectFixFail	String
Representation	Optional Only allowed for XML signing	Attached representation document in base64 format. This must be PDF data.	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
RepresentationType	Conditional	Type of the representation document, must be present when Representation is filled. Only "application/pdf" is supported.	String
SigningTemplateCode	Optional Forbidden for XML signing	<p>The template configured in the portal will provide all necessary information.</p> <p>The SigningTemplateCode can either be retrieved in the portal or via the Get signing templates call. See 6.2 Getting Signing Templates (Paginated) for more info.</p> <p>When you use this parameter, further use of the StakeHolders parameter will be forbidden.</p>	String
TargetType	Optional	<p>The TargetType defines if an extra conversion to PDF/A needs to be done before signing. Values: Pdf PdfA1A PdfA2A</p> <p>Notes This will only work when the Document Conversion settings have been enabled in the Configuration Index. When using itsme as signing method, it is mandatory to use PdfA1A or PdfA2A as TargetType. Note however that Connective does not perform any checks whether this TargetType has been selected.</p>	String

5.7.6.1 Stakeholder information

The following parameters specify who will sign or receive this package:

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders (array of objects)	Required (1-n)	Information about the people who are involved with this package.	Array of objects
Actors	Required (1-n)	Array of actor objects. See below.	Array of objects
EmailAddress	Required	Email address of the signer.	String
FirstName	Required	First name	String
Language	Required	UI language of the stakeholder. Currently supported: en, nl, de, fr, es.	String
LastName	Required	Last name	String
BirthDate	Conditional	<p>Date of birth in format YYYY-MM-DD</p> <p>Note: activating mandated signer validation in the API or configuration might make this required. See section 5.4.12.</p>	
ExternalStakeholderReference	Optional	Reference given by the calling application, this parameter will not be used by the eSignatures Portal.	String

5.7.6.2 Actor

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders => Actors (array of objects)	Required (1-n-)	This object gives information about what the stakeholder must do.	Array of objects
Type	Required	Signer Receiver Note: actor of type Signer will automatically become a receiver as well.	String
OrderIndex	Required for Signer	This number specifies in which order actors need to execute their action. If this number is the same for all actors, then the process can happen in parallel. Incrementing numbers indicate a sequential flow and repeats indicate parallel work again.	Integer
SigningFields	Required for Signer	Define the locations where this actor should sign. See section 5.7.6.3 below.	Array of objects
SigningTypes	Required (1-n) for Signer	One or more signing type info objects. See section 9.	Array of objects
Phonenumber	Optional Only Signer	Phone number to receive an SMS OTP. Note: always add the country code in front of the phone number. E.g. +32xxxxxxxx. It is recommended to use the plus sign as international dialing prefix instead of using "00". Important: never use spaces in the phone number format.	String
RedirectUrl	Optional Only Signer	Url to which the end user is redirected after signing. This field must be a valid absolute url. See section 5.4.14 Redirect URL Details . Note: during asynchronous signing, the signer has the possibility to close the signing session - by means of a Close button - while the signing continues in the background. The purpose of a RedirectUrl however is to redirect the signer to a new URL after the signing has finished. Therefore, when a RedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.	String
SendNotifications	Optional Only Signer	eSignatures can send e-mails to all the people who are allowed to sign. Such notifications can be enabled or suppressed by setting this parameter as ' true ' or ' false ' (the default is ' false ').	Boolean
UserRoles	Conditional Only Signer	Role or function of the signer. Note: only allowed if signature policy id is present. See section 5.4.13 below.	Array of strings
LegalNoticeCode	Optional Only Signer Forbidden for XML signing	LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3 The 3 values will point to the 3 notices built into the application. These can be altered in the Configuration Index.	String
LegalNoticeText	Optional Only Signer Forbidden for XML signing	Custom legal notice text in case none of the three predefined legal notices apply. The text should be written in the same language as the one used in the documents of the package.	String

5.7.6.3 Signing Field Position

Signature field locations can be set on PDFs either using coordinates in the document or using the id of an object in the document. See [section 10](#) for more info. When placing a field on a PDF it should at least be 112 px wide and 70 px high. **Note:** it is recommended to use slightly higher values than the minimum ones. E.g. 75 px x 120 px. Using the absolute minimum values may lead to round-off errors during conversions.

If *no* **MarkerOrFieldId** parameter is specified, then all the other fields will be **required**. When the **MarkerOrFieldId** parameter is specified then all other parameters are forbidden.

Invisible PDF Signing

When the **SigningType** parameter is set to **Server** for all actors, it is possible to sign the document without adding a visible signing field. In this case, the PDF will be signed, and the signature can be checked but it will not be visible on the document.

To leave the visible signing field out, all parameters of the **SigningFields** parameter must be omitted.

Note that this feature is only available in Instant Packages

XML Signing

XML signing does not allow for coordinates or markers to form a visual signature, instead all signature data will become part of the XML document and be added at the end. Therefore, this parameter should be **empty** when signing XML. The API will make one location for each actor in the call.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholder--> Actors--> SigningFields (array of objects)	Required (1-n)	One or more signing locations in the document	Array of objects
PageNumber	Conditional	Number of the page on which to add a signing location. First page is number 1. Negative page numbers work as described in section 10.1 .	Integer
Width	Conditional	Width Minimum value is 112. It is recommended to use slightly higher values than the minimum ones.	String
Height	Conditional	Height Minimum value is 70. It is recommended to use slightly higher values than the minimum ones.	String
Left	Conditional	Position from the left of the page. Minimum value is 1. Negative values are not supported.	String
Top	Conditional	Position from top of the page. Minimum value is 1. Negative values are not supported.	String
MarkerOrFieldId	Conditional	Unique reference to a signing field, text marker or textfield. See section 10.2 for more details.	String

5.7.6.4 SigningType Specific Information

The following object defines what kinds of signing types are allowed and it can define extra validation steps for that signing type.

Note: when any of the optional parameters are specified, all signing type objects need to contain the same values for those parameters. This is especially important for **MandatedSignerValidation**. When you are using choice of signing (i.e. when the signer may choose from different signing types), and you set the **MandatedSignerValidation** parameter to **MatchId** or

NameAndBirthDate for one signing type, you must also set that parameter to the same value for all the other signing types for that actor, even though those signing types may not support **MandatedSignerValidation**.

This restriction will be solved in a future version of eSignatures, making all kinds of combinations possible.

Important: because of this restriction, it is not possible to combine **BeLawyer** and **itsme** signing in choice of signing when using **MandatedSignerValidation**. This is because **BeLawyer** requires **MatchId** as value, while **itsme** requires **NameAndBirthDate** as value.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Stakeholders => Actors => SigningTypes (array of objects)	Required (1-n)	This object specifies the signing type and its related properties.	Array of objects
SigningType	Required	The signing type used in this actor's session. See section 9 .	String
CommitmentTypes	Conditional Forbidden for XML signing	One or more OIDs of commitment types. Can only be passed when signature policy is used. See section 5.4.13 .	Array of strings
MandatedSignerValidation	Optional	Type of validation to execute during eID other smart card, or itsme signing session. See section 5.4.12 . Values: Disabled NameAndBirthDate MatchId Note that MatchId is not supported for itsme.	String
MandatedSignerIds	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See section 5.4.12 . Required when mandated signer validation is of type MatchId.	Array of string
SignaturePolicyId	Optional Forbidden for XML signing	Signing policy details for the signature. See section 5.4.13 .	String

5.7.7 Example Request

5.7.7.1 Regular Request

```

{
  "Document": " Base64 of your document",
  "DocumentLanguage": "en",
  "DocumentName": "Instant doc",
  "Initiator": "John.Doe@mail.com",
  "DocumentGroupCode": "00052",
  "ExpiryTimestamp": "2019-01-22T13:00:28+00:00",
  "ExternalDocumentReference": "MyDocument",
  "F2FRedirectUrl": "https://yourF2FSignUrl",
  "TargetType": "PdfA2A",
  "PdfErrorHandling": "DetectFixFail",
  "Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Smith",
      "EmailAddress": "john.smith@mail.com",
      "BirthDate": "1994-09-23",
      "Language": "en",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 1,
          "SigningFields": [{"Width": "150", "Height": "100", "PageNumber": 1, "Left": "100", "Top": "400"}],
          "SigningTypes": [
            [
              {
                "SigningType": "manual",
              },
              {
                "SigningType": "BeId",
                "MandatedSignerValidation": "MatchId",
                "MandatedSignerIds": "94092306689",
              }
            ],
            "Phonenumber": "+32499111213",
            "SendNotifications": true,
          }
        ]
      ]
    }
  ]
}

```

5.7.7.2 Request with SigningTemplateCode

In this example you'll notice that the stakeholder block is not present (since that info is covered by the template).

```

{
  "Document": " Base64 of your document",
  "DocumentLanguage": "en",
  "DocumentName": "Instant doc",
  "Initiator": "Jon.Doe@mail.com",
  "DocumentGroupCode": "00052",
  "ExpiryTimestamp": "2019-06-25T15:00:28+00:00",
  "ExternalDocumentReference": "MyDocument",
  "RedirectUrl": "https://www.mycompany.com",
  "TargetType": "PdfA2A",
  "PdfErrorHandling": "DetectFixFail",
  "SigningTemplateCode": "00001",
}

```

5.7.8 Example Response

```

{
  "PackageId": "9b4b3d2b-f495-40ca-b6fd-b6fb9b0d916b",
  "PackageName": "Instant doc",
  "Initiator": "John.Doe@mail.com",
  "ExpiryTimestamp": null,
  "F2FSigningUrl": "https://yourF2FSignUrl",
  "PackageStatus": "Pending",
  "PackageDocuments": [
    {
      "DocumentId": "67c82b46-1871-4cf6-a6b7-f5a701bd7e98",
      "ExternalDocumentReference": "MyDocument",
      "Name": "Instant doc"
    }
  ],
  "Stakeholders": [
    {
      "EmailAddress": "john.smith@mail.com",
      "StakeholderId": "6ea96cd0-9933-452f-9b7f-4325980642cd",
      "Actors": [
        {
          "ActorId": "1d4f4cf4-f22b-49a0-ba7a-d4f57e415f5d",
          "ActionUrl": "https://ActionUrl.com",
          "ActionType": "Signer",
          "Reason": null,
          "ActorStatus": "Available"
        },
        {
          "ActorId": "a3005f08-cc33-44fe-a17f-89fb362ce9a6",
          "ActionUrl": "https://ActionUrl.com",
          "ActionType": "Receiver",
          "Reason": null,
          "ActorStatus": "Available"
        }
      ]
    }
  ]
}

```

5.7.9 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId	Unique id of the package	String
<Other parameters>	The other parameters in the response are currently the same as those of the Get Package Status call. See section 5.8 .	

5.7.10 Response codes

RESPONSE STATUS CODE	DESCRIPTION
201 Created	The package can be made ready for signing. The response contains more information about the locations where a signer can place a signature.
400 Bad request	The request contained parameters which could not be accepted.
404 Not Found	The package id could not be found in the database.
409 Conflict	When certain document conversions are forbidden, when the input document has issues, or when marker ids are not matched.

5.7.11 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldsMissing
400	Document.UnsupportedLanguage
400	Package.ExpiryTimestampInPast
400	Document.InvalidTargetFileType
400	SigningField.LabelNotUnique
400	SigningField.MarkerAndCoordinatesCannotBeMixed
400	SigningField.MarkerNotUnique
400	SigningField.InvalidHeightCoordinate
400	SigningField.InvalidWidthCoordinate
400	Document.PasswordProtected
400	Pdf.UploadDoesNotComplyToSpec
400	Actor.TypeInvalid
400	SigningType.Invalid
400	Stakeholder.EmailAddressInvalid
400	Stakeholder.BirthDayInFuture
400	Stakeholder.UnsupportedLanguage
400	PdfErrorHandling.InvalidType
400	Actor.MissingPhoneNumber
400	SigningField.InvalidPhoneNumber
400	Signer.ParameterCannotBeUsedWithoutSignaturePolicy
400	SignaturePolicy.MissingCommitmentType
400	SignaturePolicy.MissingUserRole
400	SigningType.MissingSignaturePolicy
400	SigningTemplate.ForbiddenParameter
400	Url.Invalid

HTTP CODE	CODE
400	Package.ExpiryTimestampInvalid
400	Package.ExpiryTimestampInThePast
409	SigningTemplate.NotFoundWithCode
409	MandatedSigner.MandatedSignerIdMissing
409	MandatedSigner.BirthDateMissing
409	DocumentGroup.NotFoundWithCode
409	User.NotFound
409	SigningType.Disabled
409	MandatedSigner.MultipleValidationTypesNotAllowed
409	SignaturePolicy.NotFound
409	Document.InvalidSourceFileType
409	SigningField.InvalidHeightMarker
409	SigningField.InvalidPage
409	SigningField.InvalidWidthMarker
409	Document.InvalidTargetFileType

5.7.12 Signing Template Restrictions

Customer-specific integrations may want to avoid continuously specifying signer and receiver information by reusing a previously saved template.

Using these templates requires that there is a one-time setup in the eSignatures Template Portal. Templates can be edited by users of the eSignatures Portal when they have the appropriate permission. If that is done, the unique id of the template needs to be looked up in the UI or through the Getting Signing Templates call so that it can be passed through the **SigningTemplateCode** parameter in the call documented above.

When a template contains multiple signers with the same email address then the end user will see all those fields in the same package signing session. If those signers had different signing types then the end user will at signing time be able to choose one of the signing types for the entire session.

Note: signing templates work based on markers PDF documents. Because XML has no such options the use of signing templates **cannot** be combined with XML signing.

Note that no Signers, Receivers, or their locations can be passed to the API when a template code is specified. Doing so will make this call return an error response.

5.8 Get Package Status

5.8.1 Description

Retrieves the current state of the package and its documents.

5.8.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/status](https://[servername]:[port]/webportalapi/v3/packages/{id}/status)

5.8.3 HTTP Method

GET

5.8.4 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.8.5 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageName	Description for the Package shown to the eSignatures Portal user as file name.	String
CreationTimestamp	Date and time when the package was created according to the server. Format is ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z	String
Initiator	Initiator field of the package as it was passed in at creation time.	String
ExpiryTimestamp	UTC formatted time at which the document expires. Can be null.	String
ExternalPackageReference	Returns the external reference id of the package as it was passed in at creation time.	String
F2FSigningUrl	Link to the package which allows to pick from all the signing session at once.	String
PackageStatus	Status of the package as a whole: Draft Pending Finished Rejected Revoked Expired Failed Note: a package has the status Failed when a background operation has failed and left a message on the Poison Queue.	String
PackageDocuments	Details for each of the documents in the package.	Array of objects
Stakeholders	Details for each of the persons which will interact with the package.	Array of objects

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageDocuments (array of object)	Details for each of the documents in the package	Array of objects
DocumentId	Unique id of the document	String
ExternalDocumentReference	Returns the external reference of this document as it was passed in through the Add document to package call.	String
DocumentName	Name of the document	String

PARAMETER	CONTENT / DESCRIPTION	TYPE
Stakeholders (array of objects)	Details for each of the persons which will interact with the package.	Array of objects
EmailAddress	Email address of the signer.	String
ExternalStakeholderReference	External reference identifying this person in the external system.	String
StakeholderId	Unique id	String
Actors	See below	Array

PARAMETER	CONTENT / DESCRIPTION	TYPE
Stakeholders => Actors (object)	Details of all steps to take.	Array of objects
ActionId	Unique id for this combination of action, stakeholder and document.	String
ActionUrl	URL that this person can open to interact with the package. Can be null.	String
ActorStatus	Draft (package has status Draft) Inprogress (package is being signed) Available (ready for execution) Finished Rejected (signing cannot continue) Failed (signing has failed)	String
Type	Signer Receiver	String
CompletedBy	The name of the end user who completed the action. This can only be properly filled when an authenticated signing method is used like Beld or Idin. Can be null, will never be present for a Receiver.	
CompletedTimestamp	Timestamp of the moment on which this action was completed. Format is ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z Can be null, will never be present for a Receiver	

PARAMETER	CONTENT / DESCRIPTION	TYPE
Reason	Returns the text entered by the person who changed the status of a package to a final state (e.g. a reject). Can be null, will never be present for a Receiver.	String

5.8.6 Example response

```
{
  "PackageName": "package-docu1.pdf",
  "Initiator": "signer@mail.com",
  "ExpiryTimestamp": null,
  "F2FSigningUrl": "http://myserver/signinit?packageSignId=6bc402eb-6cbd-423a-bf00-1157e8d68f37&f2f=True",
  "PackageStatus": "Draft",
  "PackageDocuments": [
    {
      "DocumentId": "ee61b9b1-5651-472b-9bef-a2d35352de50",
      "DocumentName": "docu1"
    },
    {
      "DocumentId": "e4b9342b-c587-4419-b057-f235d517159c",
      "DocumentName": "docu2"
    }
  ],
  "Stakeholders": [
    {
      "EmailAddress": "john.doe@example.org",
      "StakeholderId": "d2fff468-5236-4610-b5ed-3d25770e528f",
      "Actors": [
        {
          "ActionUrl": "http://myserver/signinit?packageSignId=79b404ff-2570-47d2-95b9-d6e01a37254a&token=NkpYfe5B0K2880d6oT37uS3zKT-AW_Uq2eh19qLfw_FJe2EwGv2aWwHGx8c1yYU_",
          "ActionType": "Signer",
          "Reason": null,
          "ActorStatus": "Available"
        },
        {
          "ActorId": "79b404ff-2570-47d2-95b9-d6e01a37254b",
          "ActionUrl": "https://www.myactionurl.com",
          "ActionType": "Receiver",
          "Reason": null,
          "ActorStatus": ""
        }
      ]
    }
  ]
}
```

5.8.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The package was returned successfully.
404 Not Found	The package with the specified id could not be found.
409 Conflict	The package with the specified id was made with an old version of the api.

5.8.8 Error codes

HTTP CODE	CODE
404	Package.NotFound
409	Package.ApiVersionMismatch

5.9 Update Signer Actors

5.9.1 Description

Once the package has its process set there might be small changes needed to the actions which a stakeholder needs to take. Note that this call can only be used for packages that are pending signing. For packages in draft you can do a new **Set Process Information** call. Currently the only change allowed this way is to restrict the set of signing types a stakeholder might use during a given signing session. Repeated calls can only further restrict the set of signing types, they can never add the signing types which were never allowed or removed in a previous call.

5.9.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/signers/{actorid}](https://[servername]:[port]/webportalapi/v3/packages/{id}/signers/{actorid})

5.9.3 HTTP Method

PUT

5.9.4 MIME Type

application/json

5.9.5 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package.	String
actorid	Required	Unique id of the actor which needs updates.	String

5.9.6 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
SigningTypes	Required	Signing types to be used during the actor's signing session.	Array of string

5.9.7 Response parameters

PARAMETER	CONTENT / DESCRIPTION
None	Empty object

5.9.8 Example response

```
{ }
```

5.9.9 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The package was successfully changed to the new status.
400 Bad Request	When invalid parameters are passed in.
404 Not Found	When an unknown package or actor id is passed in, or the actor is not a signer.
409 Conflict	When the signing types were never allowed in the first place or when the package status is wrong.

5.9.10 Error codes

HTTP CODE	CODE
400	Request.RequiredFieldIsMissing
404	Package.NotFound
404	Actor.NotFound
409	Package.InvalidStatus
409	Package.ApiVersionMismatch
409	Package.ContainsNoPackageSigners
409	Signer.SigningTypesRestricted

5.10 Download Package

5.10.1 Description

The package containing the signed documents can be downloaded by an external system using this call.

The package will be downloaded as .zip file.

Note: The package can only be downloaded if its status is **Finished**. If the package is in any other state, the request will fail.

5.10.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/download](https://[servername]:[port]/webportalapi/v3/packages/{id}/download)

5.10.3 HTTP Method

GET

5.10.4 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package	String

5.10.5 Response

The eSignatures API will return a zip file containing all documents. Each file in the zip file will use the value passed in **DocumentName**, optionally suffixed with a number to keep the filenames unique.

5.10.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The package gets downloaded successfully.
404 Not Found	The package cannot be found.
409 Conflict	The package hasn't been fully signed.

5.10.7 Error codes

HTTP CODE	CODE
404	Package.NotFound
404	Document.NotFoundInStore
409	Package.InvalidStatus

5.11 Download Document from Package

5.11.1 Description

The signed documents in a package can be downloaded one by one by an external system using this call. Each document will be downloaded as a PDF, or as an XML file stream, depending on the value of the **DocumentType** parameter.

Note: The documents can only be downloaded when the package has status **Finished**. If the package is in any other status, the request will fail.

5.11.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/download/{documentId}](https://[servername]:[port]/webportalapi/v3/packages/{id}/download/{documentId})

5.11.3 HTTP Method

GET

5.11.4 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id for the signing package.	String
documentId	Required	Unique id of the document contained in the package.	String

5.11.5 Response

The eSignatures API will return a PDF document, or an XML file stream, depending on the value of the **DocumentType** parameter.

5.11.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The document gets downloaded successfully.
404 Not Found	The document cannot be found or is not part of the specified package.
409 Conflict	The package hasn't been fully signed.

5.11.7 Error codes

HTTP CODE	CODE
404	Document.NotFound
404	Package.ContainsNoDocumentWithId
404	Document.NotFoundInStore
409	Package.InvalidStatus
409	Document.InvalidStatus

5.12 Package Expiry Extension

5.12.1 Description

A package may have the status Expired when a package passed a value for the **ExpiryTimestamp** parameter in the Create Package call. Such a package can no longer be signed.

The "extend expiry" call allows to specify a new date and time in the future on which the package will expire, thus allowing signing again until that date and time.

An expiration timestamp can currently be set on a package which has the status Draft, Expired or Pending. An expiration timestamp cannot be removed once set, but it is possible to choose a date and time far in the future instead.

5.12.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/expirytimestamp](https://[servername]:[port]/webportalapi/v3/packages/{id}/expirytimestamp)

5.12.3 HTTP Method

PUT

5.12.4 MIME Type

application/json

5.12.5 Template parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Id	Required	Unique id for the signing package	String

5.12.6 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
ExpiryTimestamp	Required	The new date and time when this package expires. This must be an ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z	Date-time

5.12.7 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
None	Empty response body	

5.12.8 Example response

Successful response

```
{}
```

5.12.9 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The update package expiry timestamp was set.
400 Bad request	The request contained parameters which could not be accepted.
400 Not Found	The package id could not be found in the database.

RESPONSE STATUS CODE	DESCRIPTION
409 Conflict	When the status of the document does not allow to extend the expiration period.

5.12.10 Error codes

HTTP CODE	CODE
400	Package.ExpiryTimestampInvalid
400	Package.ExpiryTimestampsRequired
404	Package.NotFound
409	Package.InvalidStatus

5.13 Send Package Reminders

5.13.1 Description

Company policy might require that a document is handled within a given timespan. Triggering the “send reminders” call will look up everybody who hasn’t signed and send them an extra notification as a reminder.

Note that only the next available signer(s) in the workflow are notified. This means signers waiting for someone else to sign first in a serial workflow will not receive a notification.

5.13.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}/reminders](https://[servername]:[port]/webportalapi/v3/packages/{id}/reminders)

5.13.3 HTTP Method

POST

5.13.4 MIME Type

application/json

5.13.5 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
PackageId	Required	Unique id of the package	String

5.13.6 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
None	Empty response body	

5.13.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The reminders have been sent.
404 Not Found	The package with the given id could not be found.
409 Conflict	The package did not have status Pending.

5.13.8 Error codes

HTTP CODE	CODE
404	Package.NotFound
409	Package.InvalidStatus

5.14 Packagelist (Paginated)

5.14.1 Description

Get a list of packages with their current status. This can be useful for periodic cleaning.

This method uses paging so only a limited number of records is returned at a time (currently limited to maximum 50 records). Query parameters are then used to specify how many items to return and which page to continue with. Further parameters (sorting, sort order) can be seen below.

Note that the ContinuationToken is meant to be opaque to clients – either give an empty value for the first page or pass a token which was returned from a previous request. Trying to generate it on the client may not be supported in future versions when the format changes.

Note that currently the response will always contain a continuation token, even if there are no more records. The client is responsible for counting the number of items seen before and comparing it with the Total parameter in the response to know when to stop making calls.

5.14.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages](https://[servername]:[port]/webportalapi/v3/packages)

5.14.3 HTTP Method

GET

5.14.4 MIME Type

application/json

5.14.5 Query parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
CreatedBeforeDate	Optional	Displays only the packages created before this date. ISO 8601 date format.	Date
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
MaxQuantity	Optional	Maximum number of records.	Int
SortField	Optional	Sort the returned records by the field which is specified. (Reserved for future use)	String
SortOrder	Optional	'ASC' or 'DESC' to specify in which way sorting happens. (Reserved for future use)	String
Status	Optional	Returns only those packages which have the requested status. The possible statuses are: draft, pending, finished, rejected, revoked, failed.	String

5.14.6 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
ContinuationToken	Token which allows the client to retrieve the next set of records.	String
MaxQuantity	The highest number of records that the result may contain. Useful when no quantity was given in the request (default subject to change).	Int
Total	Number of records found in the database. The client can use the quantity and total to calculate the number of calls needed to retrieve all records.	Int

PARAMETER	CONTENT / DESCRIPTION	TYPE
Items	List of packages.	Array of objects
PARAMETER	CONTENT / DESCRIPTION	TYPE
Items (array of objects)	Array, 0..*	Array of objects
PackageId	Package id for this package	String
PackageStatus	Status of the package as a whole: Draft Pending Finished Rejected Revoked Failed Note: a package has the status Failed when a background operation has failed and left a message on the Poison Queue.	String
ExternalPackageReference	Reference given by the calling application. This will not be used by eSignatures.	String

5.14.7 Example response

Take for example the following GET request:

```
https://[host]/webportalapi/v3/packages?MaxQuantity=2&ContinuationToken=
```

```
{
  "ContinuationToken": "68036c7f-db6c-4dd0-bc1d-cb7337b2259f",
  "Items": [
    {
      "Id": "c3940cf6-fa80-441f-8971-55af6d00fb9d",
      "PackageStatus": "DRAFT",
      "ExternalPackageReference": "INVOICE-18-0048"
    },
    {
      "Id": "0eeaa964-4197-41aa-9d6e-875b8e6d1a92",
      "PackageStatus": "PENDING",
      "ExternalPackageReference": "INVOICE-18-0009"
    }
  ],
  "MaxQuantity": 2,
  "Total": 21
}
```

5.14.8 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The packagelist gets returned successfully.
400 Bad Request	A request parameter was invalid.

5.14.9 Error codes

HTTP CODE	CODE
400	Pagination.MaxQuantity.OutOfBounds

5.15 Delete Package

5.15.1 Description

eSignatures does not automatically delete packages from the database once they have reached a final state. They are stored indefinitely.

To delete packages from the database, users can use the Delete Package call.

Note: deleting a package can only be done when the package has status Draft or one of the final states Finished, Rejected or Revoked.

5.15.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{id}](https://[servername]:[port]/webportalapi/v3/packages/{id})

5.15.3 HTTP Method

DELETE

5.15.4 MIME Type

application/json

5.15.5 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
id	Required	Unique id of the package	String

5.15.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The package was deleted.
404 Not Found	The package with the given id could not be found.
409 Conflict	The package's status is still in a non-final state (e.g. Pending, Expired) so it can't be deleted yet.

5.15.7 Error codes

HTTP CODE	CODE
404	Package.NotFound
409	Package.InvalidStatus
409	Package.ExpiryTimestampInThePast

5.16 Delete Document from Package

5.16.1 Description

Sometimes a package might contain a document which shouldn't have been added. In that case it is possible to delete it from the package when the package still has the Draft status.

When the package has some other status it is only possible to revoke it and start from scratch.

5.16.2 URL

`https://[servername]:[port]/webportalapi/v3/packages/{id}/documents/{documentId}`

5.16.3 HTTP Method

DELETE

5.16.4 MIME Type

application/json

5.16.5 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
packageId	Required	Unique id of the package	String
documentId	Required	Unique id of the document inside the package	String

5.16.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
404 Not Found	No document could be found for the given id.
409 Conflict	The package's status is different from [Draft] or the package was made with an old version of the api.

5.16.7 Error codes

HTTP CODE	CODE
404	Document.NotFound
404	Package.NotFoundForDocument
409	Package.ApiVersionMismatch
409	Package.InvalidStatus

6. Miscellaneous Services

6.1 [Get DocumentGroups](#)

6.2 [Get Signing Templates \(Paginated\)](#)

6.3 [Get Enabled Signing Types](#)

6.1 Get DocumentGroups

6.1.1 Description

Some requests need to identify a Document Group in which a document is to be used. This request allows to list the names of document groups and most importantly their associated codes.

There is always at least one document group: "My Documents" (the name could be different) with Code "00001". This group is special because the documents in this group are only visible to the eSignatures Portal user who uploaded the document (in case of an upload made through the API from this document it will be the user whose email was given as **Initiator**).

Please note that the Code field is a **string**. Its value may look numeric but its leading zeroes are part of the value.

6.1.2 URL

https://[servername]:[port]/webportalapi/v3/documentgroups

6.1.3 HTTP Method

GET

6.1.4 MIME Type

application/json

6.1.5 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
DocumentGroups	List of all document groups	Array of objects

PARAMETER	CONTENT / DESCRIPTION	TYPE
DocumentGroups (array of objects)	Document group details	Array of objects
Name	Name or description of the document group	String
Code	A unique value identifying each document group	String

6.1.6 Example response

```
{
  "DocumentGroups" : [{
    "Name" : "My Documents",
    "Code" : "00001"
  }, {
    "Name" : "HR",
    "Code" : "00002"
  }, ]
}
```

6.1.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The documentgroups gets returned successfully.

6.1.8 Error codes

HTTP CODE	CODE
None	No error messages possible

6.2 Get Signing Templates (Paginated)

6.2.1 Description

Some requests need to identify a signing template so that they don't have to specify all signers and details each time a document is uploaded. This request allows to list the names of signing templates and their associated codes.

Signing templates can be created in the eSignatures Portal by a user with the appropriate permissions.

Please note that the Code field is a **string**. Its value may look numeric but its leading zeroes are part of the value. The code should be looked up every time it gets used because deleting a template and creating a new one means the Code field is set to a unique value.

This method uses paging so only a limited number of records is returned at a time (currently limited to maximum 50 records). Query parameters are then used to specify how many items to return and which page to continue with. Further parameters (sorting, sort order) can be seen below.

Note: signing templates can only be used in Instant Package Creation calls.

Note: the ContinuationToken is meant to be opaque to clients – either give an empty value for the first page or pass a token which was returned from a previous request. Trying to generate it on the client may not be supported in future versions when the format changes.

Note: currently the response will always contain a continuation token, even if there are no more records. The client is responsible for counting the number of items seen before and comparing it with the Total parameter in the response to know when to stop making calls.

6.2.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/signingtemplates](https://[servername]:[port]/webportalapi/v3/signingtemplates)

6.2.3 HTTP Method

GET

6.2.4 MIME Type

application/json

6.2.5 Query parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
MaxQuantity	Optional	Maximum number of records.	Int
SortField	Optional	Sort the returned records by the field which is specified. (Reserved for future use)	String
SortOrder	Optional	'ASC' or 'DESC' to specify in which way sorting happens. (Reserved for future use)	String
Tags	Optional, 0...*	Labels (Tags) to define the correct template. Multiple labels (tags) are possible.	String

6.2.6 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
ContinuationToken	Token which allows the client to retrieve the next set of records.	String

PARAMETER	CONTENT / DESCRIPTION	TYPE
MaxQuantity	The highest number of records that the result may contain. Useful when no quantity was given in the request (default subject to change).	Int
Total	Number of records found in the database. The client can use the quantity and total to calculate the number of calls needed to retrieve all records.	Int
Items	List of signing template items.	Array of objects
PARAMETER	CONTENT / DESCRIPTION	TYPE
Items (array of objects)	Array, 0..*	Array of objects
Code	Unique id of the signing template.	String
Name	Signing template name.	String
CreationTimestamp	The creation date and time of the signing template. Uses the ISO 8601 date-time format, e.g. 2018-01-23T12:34:00.000Z	DateTime

6.2.7 Example response

Take for example the following GET request:

```
https://[host]/webportalapi/v3/signingtemplates?MaxQuantity=2&ContinuationToken=
```

```
{
  "ContinuationToken": "45b8530d-c54b-46ba-bbfe-7b80209c7e4e",
  "Items": [
    {
      "Code": "c3940cf6-fa80-441f-8971-55af6d00fb9d",
      "Name": "MyTemplate",
      "CreationTimestamp": "2017-01-25T13:35:49.844Z"
    },
    {
      "Code": "0eeaa964-4197-41aa-9d6e-875b8e6d1a92",
      "Name": "DepartmentTemplate",
      "CreationTimestamp": "2017-01-25T13:36:23.513Z"
    }
  ],
  "MaxQuantity": 2,
  "Total": 21
}
```

6.2.8 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The template list of templates gets returned successfully.
400 Bad request	A request parameter was invalid.

6.2.9 Error codes

HTTP CODE	CODE
400	Pagination.MaxQuantity.OutOfBounds

6.3 Get Enabled Signing Types

6.3.1 Description

This call retrieves the list of signing types that have been enabled in the eSignatures Configuration Index and are thus available in the eSignatures Portal. While section 9. [Signing Types](#) lists all possible signing types, this call returns only those signing types for which the IsEnabled flag is set to "On" and the signing type configuration is complete.

This call is useful for integrators who need to know which signing types are currently enabled in a given eSignatures installation, otherwise that list of signing types would need to be duplicated in the integrator's configuration database as well.

6.3.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/signingtypes](https://[servername]:[port]/webportalapi/v3/signingtypes)

6.3.3 HTTP Method

GET

6.3.4 MIME Type

application/json

6.3.5 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
SigningTypes	List of enabled signing types	Array of objects
PARAMETER	CONTENT / DESCRIPTION	TYPE
SigningTypes (array of objects)	Array, 1..*	
Name	Name of the signing type for use in Set Process Information or Create Instant package calls.	String

6.3.6 Example response

```
{
  "EnabledSigningTypes":
  [
    {
      "Name": "Manual"
    },
    {
      "Name": "BeId"
    }
  ]
}
```

6.3.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The signing types list gets returned successfully.

6.3.8 Error codes

HTTP CODE	CODE
None	No errors

6.4 Get Version

6.4.1 Description

This call checks the version and build number of eSignatures.

6.4.2 URL

https://[servername]:[port]/webportalapi/v3/version

6.4.3 HTTP Method

GET

6.4.4 MIME Type

application/json

6.4.5 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
ProductVersion	Number of the version	String
FileVersion	Number of the file (build)	String

6.4.6 Example response

```
{
  "ProductVersion": "5.2.4",
  "FileVersion": "5.2.4.48326.02"
}
```

6.4.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The version and build number get returned successfully.

6.4.8 Error codes

HTTP CODE	CODE
None	No error messages possible

7. Audit proofs

When the Audit proofs setting is enabled in the Configuration Index, eSignatures keeps Audit proofs about each signature that is placed.

The following information is collected in base64 format about each signature:

Signing events information

- Start time of signing, end time of signing
- Time when reminder was sent, content of the reminder, recipient(s) of the reminder
- Time when an SMS OTP was sent, content of the SMS, recipient of the SMS
- Time when a mail OTP was sent, content of the mail, recipient of the mail
- Time when an invitation was sent to sign the package, content of the invitation, recipient(s)

Signature information

- Signature certificate that was used to place the signature, its certificate chain and certificate revocation information (OCSP / CRL)
- Timestamp certificate, its certificate chain and certificate revocation information (OCSP / CRL)
- The signed PDF when the package is fully signed
- If the setting **Intermediate States Saved** is enabled in the Configuration Index, a copy of the document gets added in base64 format after each signature. **Note:** the base64 of PDFs is stored in the Proofs folder in the repository, and no longer in the database to reduce its load.
- Any extra proofs which were added by the client through the Post extra proof call described in section 7.3.

All this data is stored in xml files which are signed at the end. The signed xml files can then be retrieved through one of the following four calls:

1. The Audit proofs of a specific document within a package, based on the document id.
2. The Audit proofs of a specific package, based on the package id.
3. The Audit proofs of all packages containing a specific package correlation id.
4. The Audit proofs of all documents containing a specific document correlation id.

Important: if a package is deleted via the API or the eSignatures Portal then the audit proofs are deleted as well.

7.1 Limitations

The Audit proof is available for documents and packages created in API v3 or in the eSignatures Portal. The correlation id parameters are not available in API v2.

Audit proofs can only be retrieved when the package or document is fully signed, or when all packages / documents in a correlated set are fully signed.

Note: The Audit proof feature has a *significant* impact on the eSignatures database. The bigger the documents, the more database space will be used. The impact grows exponentially with bigger documents.

Note: The Audit proof feature *also has an impact* on the signing speed of eSignatures. The bigger the documents, the longer the signing time will be. Small documents (<1MB) do not seem to impact the signing speed but might do so once they contain enough signatures and signature revocation data.

Note: Retrieving the Audit proof xml for a set of packages/documents with a correlation id is only supported when all the packages in the set have been fully signed or are otherwise in a "final" state.

7.2 Retrieve Package or Document Audit Proofs Xml

7.2.1 Description

A package's audit proofs xml can be retrieved when the package is fully signed. The same applies for a document: it only works when the containing package is fully signed.

7.2.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{packageId}/auditproof/download](https://[servername]:[port]/webportalapi/v3/packages/{packageId}/auditproof/download)

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{packageId}/auditproof/download/{documentId}](https://[servername]:[port]/webportalapi/v3/packages/{packageId}/auditproof/download/{documentId})

7.2.3 HTTP Method

GET

7.2.4. MIME Type

application/xml

7.2.5. Response parameters

The API will return a signed xml containing the Audit proofs (if the package status is right). See section [7.5](#) for the format.

7.2.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The Audit proofs xml is returned.
404 Not Found	The documentation/package id could not be found.
409 Conflict	The package is not fully signed.

7.2.7 Error codes

HTTP CODE	CODE
	TBD

7.3 Add Proof from External Source

7.3.1 Description

This call allows API users to add extra proofs to a location on a document.

This call can be done multiple times for the same location and even when the package is fully signed (in this case the API user is responsible for making sure that any proof is added before retrieving the signed Audit proofs xml).

7.3.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packages/{packageld}/auditproof/proofs](https://[servername]:[port]/webportalapi/v3/packages/{packageld}/auditproof/proofs)

7.3.3 HTTP Method

POST

7.3.4 MIME Type

application/json

7.3.5 Request parameters

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
Content	Mandatory	The actual content of the proof.	Base64 string
LocationId	Mandatory	Location of the signature for which the proof content was generated.	Guid
Name	Mandatory	Name of the proof.	String
Type	Mandatory	A machine-readable "type" key. Can be freely chosen.	String
Description	Optional	Brief human-readable description of the proof.	String
IpAddress	Optional	IP address of the end user for which the proof was added.	String

7.3.6 Example request

```
{
  "locationId": "740745da-eda9-4520-a7b8-0b5b930667d3",
  "name": "pdfSignature-1523563886023.log",
  "description": "Traces de la signature du pdf",
  "type": "OTHER",
  "content": "Your signing code is 045002 In-Base-64",
  "ipaddress": "192.168.0.3"
}
```

7.3.7 Response codes

RESPONSE STATUS CODE	DESCRIPTION
204 No content	The proof was created for the requested location.
400 Bad Request	The request contained parameters which could not be accepted.
404 Not Found	The documentId or the LocationId could not be found.
409 Conflict	The proof could not be added due to some other reason.

7.3.8 Error codes

HTTP CODE	CODE
	TBD

7.4 Retrieve Package or Document Correlation Audit Proofs Xml

7.4.1 Description

Correlation ids are used to track packages or documents across several passes through the eSignatures application. On a document it states the "external identity" of that document so that it can be established what happened to it, whereas for a package it depends on how you interpret the data.

Either way, retrieving the audit proofs for such an identifier will combine all available audit proofs into one big XML. Items will be grouped into packages and documents to indicate how they were uploaded several times of the lifespan of the set.

Important: combined audit proofs xmls should only be retrieved when all packages containing the package correlation id or all documents with the given correlation id are fully signed, or otherwise in a final state.

7.4.2 URL

[https://\[servername\]:\[port\]/webportalapi/v3/packagecorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/webportalapi/v3/packagecorrelations/{correlationId}/auditproof/download)

[https://\[servername\]:\[port\]/webportalapi/v3/documentcorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/webportalapi/v3/documentcorrelations/{correlationId}/auditproof/download)

7.4.3 HTTP Method

GET

7.4.4 MIME Type

application/xml

7.4.5 Response parameters

The API will return a signed xml containing the Audit proofs (if the status of all items in the set is right). See section 7.5 for the format.

7.4.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The Audit proofs xml is returned.
404 Not Found	The document/package id could not be found.
409 Conflict	The package is not fully signed.

7.4.7 Error codes

HTTP CODE	CODE
	TBD

7.5 Audit Proofs XML Format

The XSD below describes the structure of the XML that will be returned:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Content">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="uploads" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="upload" maxOccurs="unbounded" minOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:dateTime" name="start"/>
                    <xs:element type="xs:dateTime" name="end"/>
                    <xs:element name="signatures">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="signature" maxOccurs="unbounded" minOccurs="1">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="proofs">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="proof" maxOccurs="unbounded" minOccurs="0">
                                        <xs:complexType>
                                          <xs:sequence>
                                            <xs:element type="xs:string" name="name"/>
                                            <xs:element type="xs:string" name="type"/>
                                            <xs:element type="xs:byte" name="index"/>
                                          </xs:sequence>
                                        </xs:complexType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        <xs:attribute type="xs:string" name="locationId" use="optional"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="indexes">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="index" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:byte" name="identifier"/>
                    <xs:element type="xs:string" name="content"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="documentCorrelationId" type="xs:string" use="optional"/>
  <xs:element name="documentId" type="xs:string" use="optional"/>
</xs:schema>
```

```
</xs:sequence>
  <xs:attribute type="xs:string" name="packageCorrelationId" use="optional"/>
  <xs:attribute type="xs:string" name="packageId" use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

8. Queuing Mechanism

Now that eSignatures supports asynchronous signing, a **queuing mechanism** has been introduced.

The queuing mechanism works as follows: for each package that is signed asynchronously, a message is placed in the command queue. The Connective WebSigner Worker tries to process each message in the command queue. If an error occurs, the Worker will retry to process the message at a later time. The interval at which the Worker retries is configurable in the eSignatures Web.config file. If the message still can't be processed after a number of attempts (also configurable in Web.config), it will be put in the poison queue.

Messages typically end up in the poison queue when the package has the wrong status (e.g. deleted, revoked, expired, rejected) and the signer still tries to sign, or due to infrastructure issues. Typical infrastructure issues are timestamp server downtime, mail server downtime, storage issues, etc.

Once a message ends up in the poison queue, it will no longer be picked up and processed. It will stay there until the API user takes further action.

The actions the eSignatures API user can take, are the following:

- **Get Poison Queue**, to retrieve the contents of the poison queue.
- **Resubmit Poison Queue**, to resubmit the contents of the poison queue to the command queue. eSignatures will retry to sign the packages.
- **Clear Poison Queue**, to clears the contents of the poison queue.

Each of the actions are detailed in the sections below.

Important: queuing mechanisms are not designed to be polled. If you want to check if a package has the status 'failed', you should use the **Get Package Status** call.

8.1 Get Poison Queue

8.1.1 Description

This call retrieves the contents of the poison queue.

The contents of the poison queue are paginated and contain the **MessageId**, **SigningSessionId** and **PackagId**, amongst others. This allows you to have a clear view where the problem occurred.

Technical administrators may want to set the **IncludeStackTrace** parameter to true, to obtain the error messages from the code in the poison queue response.

When you examine the poison queue contents and notice that the issues are caused by infrastructure issues, such as timestamp server downtime, mail server downtime, storage issues, etc., you can use the **Resubmit Poison Queue call (8.2)** to resubmit the contents. If the infrastructure issues have been solved, the signing should succeed at a next attempt. This prevents administrators from having to send documents again and signers from having to sign again, which is especially convenient when a large number of documents have been sent.

If, however, the failed packages are not caused by infrastructure issues, but by an incorrect status or corrupt data, resubmitting them won't solve the problem. They will end up in the poison queue again. In this case, the initiator needs to check their documents and send them again. Afterwards they may clear the poison queue (8.3).

8.1.2 URL

https://[servername]:[port]/webportalapi/v3/poisonqueue

8.1.3 HTTP Method

GET

8.1.4 MIME Type

application/json

8.1.5 Query parameters

PARAMETER	OCCURRENCE	CONTENT/DESCRIPTION	TYPE
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
IncludeStackTrace	Optional	Whether to include full details of the failed message. This is useful for technical administrators to view the error messages from the code.	Boolean
MaxQuantity	Optional	Maximum number of records.	Int
SortField	Optional	Sort the returned records by the field which is specified. (Reserved for future use)	String
SortOrder	Optional	'ASC' or 'DESC' to specify in which way sorting happens. (Reserved for future use)	String

8.1.6 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
ContinuationToken	Token which allows the client to retrieve the next set of records.	String
MaxQuantity	The highest number of records that the result may contain.	Int

PARAMETER	CONTENT / DESCRIPTION	TYPE
Total	Number of records found in the database. The client can use the quantity and total to calculate the number of calls needed to retrieve all records.	Int
Items	List of items in the poison queue.	Array of objects
PARAMETER	CONTENT / DESCRIPTION	TYPE
Items (array of objects)	Array, 0.. *	Array of objects
MessageId	Unique identifier of the poison queue message	String
Payload	Contents of the message	
DeliveryTime	The date and time when this package was delivered to the poison queue. Format is ISO 8601 date-time. E.g. 2019-01-23T12:34:00.000Z	Date/Time
Retries	The number of times eSignatures has tried to sign the package.	Number
Exception	Exception that is thrown when signing has failed. This is left out by default. It can be included by setting the IncludeStackTrace parameter to true.	String
CorrelationId	This parameter is built in for future use.	String

8.1.7 Example response

```
{
  "ContinuationToken": "1",
  "Items": [
    {
      "MessageId": "4fb792d8-7c15-470a-8a7b-3c5350043ff7",
      "Payload": {
        "Id": "4fb792d8-7c15-470a-8a7b-3c5350043ff7",
        "SigningSessionId": "90b2297b-841f-4e80-a389-7aa13370bb9c",
        "PackageId": "bc6f1af7-d3dc-4630-8200-a365a8f42d02"
      },
      "DeliveryTime": "2019-01-10T10:22:31.266949",
      "Retries": 4,
      "Exception": "Connective.Common.Exceptions.VNext.ItemNotFoundExceptionVNext: Exception of type 'Connective.Common.Exceptions.VNext.ItemNotFoundExceptionVNext' was thrown.\r\n at",
      "CorrelationId": null
    }
  ],
  "MaxQuantity": 20,
  "Total": 1
}
```

8.1.8 Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	The contents of the poison queue are returned successfully.

8.1.9 Error codes

N/A.

8.2 Resubmit Poison Queue

8.2.1 Description

This call resubmits the contents of the poison queue to the command queue. eSignatures then retries to sign the packages.

This call is useful when infrastructure issues occurred, which prevented the signing from succeeding. For instance, the timestamp server or mail server was down for a period of time, which caused all packages to fail during that timeframe. Resubmitting the contents of the poison queue after the infrastructure issues have been solved will prevent signers from having to sign again.

If, however, the failed packages are not caused by infrastructure issues, but by an incorrect status or corrupt data, resubmitting them won't solve the problem. They will end up in the poison queue again. In this case, the initiator needs to check their packages, make sure they aren't corrupt and send them again. Afterwards, they may clear the poison queue (8.3).

8.2.2 URL

https://[servername]:[port]/webportalapi/v3/poisonqueue

8.2.3 HTTP Method

POST

8.2.4 MIME Type

application/json

8.2.5 Response parameters

N/A.

8.2.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
204 No content	The contents of the poison command queue are resubmitted successfully to the command queue.

8.2.7 Error codes

N/A.

8.3 Clear Poison Queue

8.3.1 Description

This call clears the contents of the poison queue.

It is recommended to use this call only after you have checked the contents of the poison queue and know which problems occurred.

8.3.2 URL

`https://[servername]:[port]/webportalapi/v3/poisonqueue`

8.3.3 HTTP Method

DELETE

8.3.4 MIME Type

application/json

8.3.5 Response parameters

N/A.

8.3.6 Response codes

RESPONSE STATUS CODE	DESCRIPTION
204 No content	The contents of the poison command queue are cleared successfully.

8.3.7 Error codes

N/A.

9. Signing Types

Actors can use one or more of the following signing types, depending on whether they have been enabled in the Configuration Index (see section 6.3 for a call to retrieve the currently enabled types):

- Manual

The end user needs to “manually” draw a signature using the mouse or stylus.

- Beld

The end user must sign using a Belgian eID.

This signing type requires a third-party smart card reader.

Note: in the previous version of the documentation the “Beld” signing type was called “Digital”.

- ManualBeld

The end user will sign using a Belgian eID but before doing so draw their signature using the mouse or stylus.

Note: in the previous version of the documentation the “ManualBeld” signing type was called “ManualDigital”.

- SmsOtp

A one-time code will be sent to the signer's phone. Depending on the configuration, users may need to complete the last digits of their phone number which was passed in the original request or entered in the Portal's Contact screen. Entering the right code received per sms will initiate the signing.

- Server

The eSignatures solution will sign this field autonomously as soon as the package is set to status ‘Pending’.

Note: this signing type can only be used when **all** locations use the Server signing type, and it cannot be used in combination with choice of signing. It is also restricted to the Instant Package Creation call.

- MailOtp

A one-time code will be sent to the signer's email address. Depending on the configuration, users may need to complete their email address which was passed in the original request or entered in the Portal's Contact screen. Entering the right code received per email will initiate the signing.

- Idin

Signing by means of Dutch bank card through iDIN.

- BeLawyer

The end user must sign using a Belgian lawyer card. This signing type requires a third-party smart card reader.

- Biometric

Signing by means of a biometric signature pad. The drivers of a supported biometric signature pad must be correctly installed on your computer.

- Itsme

Signing via the Belgian Mobile ID itsme app.

Attention: when using the itsme signing type, the following conditions must be met:

- The TargetType must be **PdfA1A** or **PdfA2A**. See the parameter's description in 5.7.6. Note that this is the user's responsibility. Connective does not check whether this TargetType has been selected in combination with itsme.
- XML documents cannot be signed with the itsme signing type due to Belgian Mobile ID's terms and conditions.
- A Signature Policy is required. The signature policy must be passed through the **SignaturePolicyId** parameter. See **Appendix IV** of **Connective – eSignatures 5.2.x – Configuration Documentation** to learn how to use a signature policy. You can also choose to leave the **SignaturePolicyId** parameter empty. In that case, the default signature policy that the system admin configured in the Config Index will be used.

- **OpenIdConnect:[Unique Name]**

As of eSignatures 5.1 you can add a custom signing type through OpenID Connect. To do so, the **OpenIDConnect** settings must be enabled and correctly configured in the Configuration Index. See **Connective – eSignatures 5.2 – Configuration Documentation**.

Important: always use the '**OpenIdConnect:**' prefix, then add the **Name** that was configured in the Configuration Index in the **OpenIdConnect** settings group.

10. PDF Signing Locations

Important: when leaving the invisible signing fields out, as explained in section 5.7.6.3, this entire chapter doesn't apply.

Signing fields can be added to a PDF in 2 different manners:

- [Via coordinates](#)
- [Via an id reference](#)

Note: all fields should at least be 112 px wide and 70 px high, so they can be reliably filled. Smaller fields might have their contents scaled to tiny size or have their content cut off. Also note that it is recommended to use slightly higher values than the minimum ones. E.g. 75 px x 120 px. Using the absolute minimum values may lead to round-off errors during conversions.

10.1 Signing location with coordinates

The location of a signature can be determined by using the coordinates (Left, Top, Height, Width and Page number in the Add Document to Package call).

Starting from eSignatures 5.1 the page number can be negative to count backwards from the last page. For example, -1 places the signature field on the last page, -2 places it on the second-last, and so on. Make sure the number (both negative and positive) does not exceed the number of pages in your document however.

10.1.1 Example request

In this example, a signature field of 100 px wide and 200 px high will be placed in the top left corner of the page, at 200 px from the top and 100 px from the left.

```
{
  "Document": "JVBERi...rest-of-the-document",
  "DocumentName" : "Invoice",
  "DocumentLanguage" : "en",
  "ExternalDocumentReference" : "INV-2018-04-01-0038",
  "SigningFields" : [
    {
      "PageNumber" : 1,
      "Width" : "100",
      "Height" : "200",
      "Left" : "100",
      "Top" : "200",
      "Label" : "Coordinates Sig"
    }
  ]
}
```

10.1.2 Example response

```
{
  "DocumentId": "e0cb4de4-673d-49fc-9bd1-7c81248984f9",
  "CreationTimestamp": "2018-03-28T08:54:38+00:00",
  "Locations": [
    {
      "Id": "8a96613f-b6ed-4227-9bde-c20d3ee0c9d6",
      "Label": "Coordinates Sig",
      "PageNumber": 1
    }
  ]
}
```

10.2 Signing location with id reference

When a document is added, the eSignatures API will search for signing fields or text markers in the document. The visual location of these fields or markers will then decide the location of the signature in the PDF. The **MarkerOrFieldId** parameter is used to reference such a field or marker. By doing so, the coordinate parameters of the Add Document To Package call will become forbidden. The eSignatures configuration has options to restrict which text fields or text strings are matched.

Note: identifiers need to be unique! A single document should not contain signing fields, text fields or text markers with the same identifier.

10.2.1 PDF signature fields

If the PDF contains dedicated (empty) signature fields, then the id of such a field will be matched against the value given in the parameter **MarkerOrFieldId** to see if the id of the field is equal.

10.2.2 PDF text fields

A PDF can contain text input fields (form-field). The name property of the input field should equal the id given in the parameter **MarkerOrFieldId**.

Note: text fields are only used when the document has not been signed yet, and their names must correspond to the format defined in the configuration (this name can be different from the hover text and it can only be reliably seen when opening the PDF in Acrobat Professional's "Form Edit" mode).

10.2.3 Text markers

In addition to the field types above the document will be scanned for special markers in the text. The format of such markers can be finetuned in the Configuration Index but it will always be of a form like #XXX000_H_W# (see below to see what all parts mean). The **MarkerOrFieldId** parameter will then be matched against the XXX000 part of the text marker to see if they are equal, the "#" does not need to be passed.

Example of markers

```
#SIG01_100_200#  
  
#SIG02_100_200#  
  
#SIG03_100_200#
```

SIG01_100_200# will be replaced by a field of height 100 px and width 200 px

Remark:

- Signing fields placed over the marker are transparent, it's required to change the font color of markers to the background color. This restriction needs to be handled for all signature markers.
- Calculation for dimensions and location is based on (pixels = (mm * 72dpi) / 25.4).

BEWARE! You must type the markers in one fluent go. Otherwise the solution will recognize each stop and delete as a specific character, thus failing the equality check.

Markers should have the format **#XXX000_H_W#**:

- X: to indicate that it is a signature field location
- 0: numerical identification of the signature field (by default '01', '02', '03' ...)
- H: height of the signature field.
- W: width of the signature field. Example: #SIG01_100_200#

10.2.4 Example request

```
{
  "Document": "JVBERi...rest-of-the-document",
  "DocumentName" : "Invoice",
  "DocumentLanguage" : "en",
  "ExternalDocumentReference" : "INV-2018-04-01-0038",
  "SigningFields" : [
    {
      "MarkerOrFieldId": "SIG01",:
      "Label" : "Marker Sig"
    }
  ]
}
```

10.2.5 Example response

```
{
  "DocumentId": "e0cb4de4-673d-49fc-9bd1-7c81248984f9",
  "CreationTimestamp": "2018-03-28T08:54:38+00:00",
  "Locations": [
    {
      "Id": "8a96613f-b6ed-4227-9bde-c20d3ee0c9d6",
      "Label": "Marker Sig",
    }
  ]
}
```

11. Error Code Descriptions

The following list describes all the error codes in more detail. The codes may contain one or more placeholders which will be discussed in each section.

11.1 Actor

Actor.NotFound:'actorId'

The given actor identifier could not be found in the database.

Actor.TypeInvalid:'FieldValue'

The Actor Type can only contain one of two possible values: "Signer" or "Receiver".

11.2 CommitmentType

CommitmentType.NotAllowed

See [section 5.4.13](#). The possible Commitment Types are limited to the ones described in this section.

CommitmentType.ShouldBeSameForActor

All the CommitmentTypes should be the same for all signingtype objects inside an actor object.

11.3 Document

Document.InvalidTargetFileType:'supported types'

The requested document type cannot be used for conversion, either because it is unsupported in eSignatures or because it has been disabled through configuration.

The placeholder includes the (comma-separated) list of supported types or configured types (which of the two is returned can be seen by the HTTP error code in which it is returned: HTTP 400 Bad Request indicates the requested type is unsupported).

Document.InvalidSourceFileType:'supported types'

The input document cannot be added because it was detected as a kind of document format which is either unsupported or disabled through configuration.

The placeholder includes the (comma-separated) list of supported types or configured types.

Document.PasswordProtected:'type of document'

The uploaded document is password protected, types of documents are pdf, word.

Document.NotFoundInStore:'document id'

The document with id 'document id' could not be found in the document store.

Document.NameInvalidLength:'document name length'

The name of the given document was longer than 150 characters.

Document.UnsupportedLanguage:'Given language'

The given document language is not supported.

11.4 Location

Location.NotFound:'FieldValue'

The provided location(s) could not be matched to one of the locations provided in the previous step: Add document to package.

11.5 MandatedSigner

MandatedSigner.BirthDateMissing

See [section 5.4.12](#): when the Mandated Signer Validation type is NameAndBirthday and the provided signing type is either Beld, ManualBeld or itsme, the Stakeholder's BirthDate must be provided in the Request.

MandatedSigner.MandatedSignerIdMissing

See [section 5.4.12](#): when the Mandated Signer Validation type is MatchId and the provided signing type is either Beld, ManualBeld or BeLawyer, a MandatedSignerId must be provided in the Request.

MandatedSigner.MultipleValidationTypesNotAllowed

See [section 5.4.12](#): you may only set a single MandatedSignerValidation Type within an actor. You must choose between **MatchId** and **NameAndBirthDate**.

11.6 Package

Package.NotFound:'package id'

The package with id 'package id' could not be found.

Package.NotFoundForDocument:'document id'

There was no package found containing document with id 'document id'.

Package.ApiVersionMismatch

The specified package was created with an old version of the api and cannot be used in the newest version of the api.

Package.InvalidStatus:'status' The operation on the specified package could not be performed because the package has an invalid status of 'status'.

Package.ApiVersionMismatch

The operation on the specified package could not be performed because the package has an invalid Api Version.

Package.ContainsNoPackageSigners:'packageId'

Package with id ['package id'] contains no packageSigners.

Package.ContainsNoDocuments:'packageId'

Package with id ['package id'] contains no documents

Package.ContainsDocumentWithLocationsWithNoSigners:'packageId'

Package with id ['package id'] contains a document with id ['document id'] with a location ['location id'] with no document-signers.

Package.ContainsNoDocumentSigners:'packageId'

Package with id ['package id'] contains no document-signers

Package.ContainsDocumentWithNoSigners:'packageId'

Package with id ['package id'] contains a document with id ['document id'] with no signers

Signer.SigningTypesRestricted:'signingType'

Signer cannot have a 'signingType' that is not already in the list of designated signingTypes.

11.7 Pagination

Pagination.MaxQuantity.OutOfBounds

The requested number of items that would be returned falls out of the bounds of the defined range.

11.8 Pdf

Pdf.UploadDoesNotComplyToSpec

Uploaded or converted document doesn't comply to the pdf specification.

11.9 PdfErrorHandling

PdfErrorHandling.InvalidType:'Given type'

Invalid value for pdf error handling method.

11.10 PhoneNumber

PhoneNumber.Invalid: 'phonenumber'

The provided phone number is not a valid mobile phone number.

PhoneNumber.PrefixDisabled: 'phonenumber'

This validation error will only be triggered if one of the provided Signing Types is SmsOtp. As described, the allowed country prefixes should be enabled in the configuration.

11.11 Request

Request.RequiredFieldsMissing:'FieldName'

The request could not be completed because a required parameter is missing.

The placeholder includes the name of the missing field. E.g. FirstName, EmailAddress, Actors, etc.

11.12 SignaturePolicy

SignaturePolicy.NotFound

See section 5.4.13. The available Signature Policies will have to be configured in a Mapping Table in advance. Mapping will be done based on the Oid of the Signature Policy.

SignaturePolicy.ShouldBeSameForActor

All the SignaturePolicyId's should be the same for all signingtype objects inside an actor object.

11.13 SigningField

SigningField.LabelNotUnique

Labels of signing field locations need to be unique. Note that using markers without passing a custom label will use that marker id as a label and that label will be included in the validation.

SigningField.MarkerAndCoordinatesCannotBeMixed

See section [5.2.7.1](#): a signing field location cannot specify both coordinates and a marker or field id.

SigningField.MarkerNotUnique

Markers must be unique within a single PDF and each marker can only be used once.

SigningField.InvalidHeightCoordinate

Signing field height needs to be larger than 70 px.

SigningField.InvalidWidthCoordinate

Signing field width needs to be larger than 112 px.

SigningField.InvalidHeightMarker:'markerOrFieldId'

The given marker id contains a height which is smaller than 70 px.

SigningField.InvalidWidthMarker:'markerOrFieldId'

The given marker id contains a width which is smaller than 112 px.

SigningField.InvalidPage

One of the signing fields to be placed on a document uses coordinates, but the specified pagenumber exceeds the number of pages in the document. The page number must be between 1 and the maximum number of pages of the document (inclusive), or when counting from the end of the document it needs to be between -1 and -(maximum number of pages) (inclusive). When a 0 or a value outside the range gets passed then this error will be returned.

11.14 SigningType

SigningType.Invalid:'FieldValue'

The passed in signing type parameter is not a valid signing type.

11.15 Stakeholder

Stakeholder.UnsupportedLanguage:'language'

The provided language is not supported. The message might contain the currently supported values.

Stakeholder.BirthdayInFuture:'fieldValue'

The provided date cannot be a valid birthday because it is in the future.

Stakeholder.EmailAddressInvalid:'fieldValue'

The provided email address is invalid.

11.16 User

User.NotFound:'email address'

The user with the defined email address could not be found.

11.17 Audit Proofs

AuditProof.Disabled

Audit proof is disabled in the Configuration Index.

AuditProof.NotAvailableForPackageCorrelationId

No audit proof available for the given package correlationId.

AuditProof.NotAvailableForDocumentCorrelationId

No audit proof available for the given document correlationId.

AuditProof.InvalidStatus

Package has an invalid status.

AuditProof.NoAuditProofsForPackageAvailable

No audit proofs are available for this package.

AuditProof.NoAuditProofsForDocumentAvailable

No audit proofs are available for this document.