

# Table of Contents

## eSignatures 6.3

### Release Notes

1. Release Notes 6.3.0
2. Release Notes 6.3.1
3. Release Notes 6.3.2

### WebPortal User Documentation

1. Introduction
2. WebPortal users
3. Form fillers
4. Approvers
5. Signers

### WebPortal Branding Documentation

#### Preface

#### Revisions

1. Access the Theme settings
2. Create and customize themes
3. Brandable components
4. Apply themes to eSignatures

### Migration Guide to API v4

1. Introduction
2. Deprecated functionalities
3. Resource based calls
4. Elements
5. documentOptions object

### API v4 Technical Documentation

#### Revisions

1. Introduction
2. Concepts overview
3. Authentication
4. Error handling responses
5. Packages
6. Documents
7. Elements
8. Stakeholders

- 9. Actors
- 10. Process
- 11. Configuration
- 12. Audit proofs
- 13. Audit trails
- 14. SigningMethods
- 15. PDF Signing Locations
- 16. Error Code Descriptions

## API v3 Technical 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 Notes

This section contains the Release Notes of eSignatures 6.3.

- [eSignatures 6.3.0](#)
- [eSignatures 6.3.1](#)
- [eSignatures 6.3.2](#)

# 1. Release Notes – eSignatures 6.3.0

Release date: 2020-10-09

## 1.1 New features

### Form fields

In eSignatures 6.3, API users can now add form fields to the documents they're sending. To that end, two new element types have been introduced: **TextBoxField** element and **CheckBoxField** element.

TextBoxFields may have a prefilled, default value that can be edited afterwards, or may be empty to be completed afterwards. CheckBoxFields can be checked or unchecked by default. Both element types may be set as required or optional.

To support the Form fields feature, a new ActorType has also been introduced: **FormFiller**. FormFiller actors only need to fill in the TextBoxFields and/or CheckBoxFields on a document. They have access to the Signer Portal but they do not approve or sign documents.

Once the FormFiller actors have completed their tasks, the documents will be sent to the required approvers or signers, depending on the signing flow. Note that the form filling must not necessarily take place before the approval. In the current setup, it's entirely possible to approve documents of which the forms have not been filled out yet.

### Optional documents

In eSignatures 6.3, initiators can determine which documents within a package are optional.

When a document is marked as optional, signers won't be required to sign them. They can if they want to, but optional documents may also be left out of the signing flow.

API users can indicate that a document is optional by setting the new `IsOptional` parameter to true when doing a POST document call.

### Handwritten SigningMethod

In eSignatures 6.3, the **Handwritten** SigningMethod has been added to the Manual Signing Behavior.

With the **Handwritten** SigningMethod users can sign documents by typing their full name on their keyboard. The signature is then added to the document in a handwritten font.

When signing their documents, users can choose from a number of preconfigured fonts. Note that it's not supported to use any other fonts at the moment.

### Configurable visual presentation of signatures

A new parameter called **ImageTextPosition** has been added to the Config Index that configures the visual presentation of signatures.

eSignatures admins can now determine whether:

- The signature text will be placed on top of the signature image (**overlay**)
- The signature text will be placed underneath the signature image (**abovebelow**)
- The visual signature will be split up into 50% image and 50% text (**sidebyside**)

### Hotjar

Hotjar has been added to eSignatures 6.3 to analyze user behavior.

### Google Analytics

Google Analytics has been added to eSignatures 6.3 to track and report website traffic.

## 1.2 Improvements

### Improved face to face signing flow

The face to face signing flow has been improved in a number of ways to support the form fields and optional documents features:

- Initiators can now select the signer during the first step of the signing flow.
- Since the signer selection is now moved to the first step, you no longer need to select the signer when rejecting or reassigning a document.
- A **Next Signer** button has been added that allows initiators to go back to the signer selection screen when the signing is in progress for the current signer. This way, they can already select the next signer and start their signing process as well.

## 1.3 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7930	/	In API v3 thr Get Signing Locations call did not always return all signing locations.
CEP-8219	/	Reassign issue has been fixed.
CEP-8361	/	The API v4 validation issue of duplicate AdditionalProperties has been fixed.
CEP-8452	/	Inconsistent date formats of CompletedDates and CreationDates in API v4 have been fixed.
CEP-8519	/	Missing Package data issue in Signer Portal has been fixed.
CEP-8138	/	Inconsistent fonts in Audit Trail have been fixed.

## 1.4 Known issues

### eSignatures 6.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8551	/	Itsme signing of packages that contain optional documents currently doesn't reach the Finished status but stays in Pending.
CEP-8888	/	Itsme signing: When one signer of multiple signers has already signed, the subsequent signers (in a sequential or complex signing flow) receive 2 mail invitations to sign, instead of one. Signers and receivers also receive 2 mails to download the finished documents.
CEP-8406	/	Handwritten signature set to <b>Overlay</b> is currently not visible.
CEP-8432	/	It's currently not possible to select PDF_A_1A or PDF_A_2A as <b>TargetFormat</b> in API v4 when using the <b>FieldId</b> parameter to detect form fields.
CEP-8530	/	Worker log issue.
CEP-8547	/	Permissions issue to view package documents in logs.

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8549	/	It's currently not possible to download the Audit Trail of packages with optional/refused documents.
CEP-8541	/	Preview issue in Config Index Theme section.
CEP-8543	/	PDF preview of optional documents is off center on mobile devices.

## eSignatures 6.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8322	/	iDIN signing issues with long legal notice.
CEP-8371	/	SMS OTP throttling error
CEP-8077	/	Incorrect error message when mandated signing with BeID is set to 'nameandbirthdate' and the birthdate is missing for the members.
CEP-8288	/	Users are not automatically added to the default user group.

## eSignatures 6.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8207	/	User permissions issue when user logs in for the first time.
CEP-8299	33469	Swagger inconsistency

## eSignatures 6.0.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8056	/	When IsAutomaticMembershipEnabled is disabled and a user who has been invited to log in, clicks the invitation link while already being logged in with another account in the same browser (for instance as initiator who sent the invitation link), they will not be able to log in, and the login of the other account will no longer work either.
CEP-6409	30654	Error message when uploading a corrupt PDF should be improved.

## eSignatures 5.5.3

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7554	/	Internet Explorer: Wrong color for signing method when choosing it for a Contact Group.

## eSignatures 5.5.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7078	/	It's currently not possible to sign with FranceConnect using eIDAS3.

#### eSignatures 5.4.2

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6288	/	Branding issue: saving a change in one of the subtabs navigates to 1 tab above.

#### Signatures 5.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5944	/	Itsme scrolling is not as smooth as expected on Safari on iPhone.

#### 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.

#### eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
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
/	/	German installation of Chrome can generate errors during signing.

## 1.6 Known limitations

#### General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A single document must not contain more than 30 signing fields.
- 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.
- Adding multiple initiators within a single package is not supported.
- 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.

- 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.

## PDF files

- PDF portfolios are not supported. This is because a PDF portfolio may contain a wide range of file types that are not supported by eSignatures. A PDF portfolio may for instance contain e-mail messages, spreadsheets, CAD drawings, PowerPoint presentations, etc. As a result, a signer will only be able to view and sign the PDF cover sheet, and not the actual files the Portfolio contains, which renders the entire Portfolio invalid.
- Uploading PDF/A documents is only allowed if the format is PDF/A\_2A or PDF/A\_1A.
- Uploading PDF/A documents to which you add form fields in eSignatures API v4 breaks the "/A" part of the PDF. As a result, the documents end up a regular PDF.

As a workaround, you must set the **TargetFormat** parameter in the POST document call to PDF/A\_1A or PDF/A\_2A (depending on the source format). The breaking will still happen but after all form fields have been filled or saved, the document will be converted to the set target format, resulting in a PDF/A document again.

Note that the workaround mentioned does not work when using the **FieldId** parameter in a POST element call. When using **FieldId** to detect existing form fields (created in Adobe Acrobat Pro for instance) on the documents you're sending through the API, it's currently mandatory to select **pdf** as **TargetFormat**.

## Itsme signing

*(Note that these limitations do not apply when using itsme through OpenID Connect)*

- When using itsme as signing method, the target type of your documents must be **PDF/A\_1A** or **PDF/A\_2A**. 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, which means QuickSigning is not supported.
- When using itsme signing, the document and package names must only contain characters that are part of ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

## iDIN signing

- Avoid using long legal notices combined with iDIN signing, especially when sending packages through the API. If long legal notices are used, the MerchantUrl exceeds its maximum number of 512 characters.

## Mandated signing rules

- The mandated signing rules that have been configured in the Config Index cannot be overwritten using API v4 calls.

## Download notifications



It may occur that Download notifications are still handled like spam / read by URL protection software, even after whitelisting them.

As a workaround, mail recipients must add the URL to the Microsoft Office 365 Advanced Threat Protection (ATP) **Do not rewrite the following URLs** list.

## 2. Upgrade Information

---

If you have a version of eSignatures installed prior to version 6.3, consult the **Connective - eSignatures 6.3 - Installation Documentation** to learn how to upgrade it to version 6.3.

---

# 1. Release Notes - eSignatures 6.3.1

Release date: 2020-10-26

## 1.1 New features

eSignatures 6.3.1 is a hotfix version and doesn't contain new features.

## 1.2 Improvements

N/A.

## 1.3 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8550	/	Packages signed with itsme signing now correctly reach the Finished status.
CEP-8551	/	Itsme signing of packages that contain optional documents now correctly reach the Finished status.
CEP-8546	/	A new response parameter has been added to API v4: VerifiedName. This parameter contains the verified name returned from the signing certificate or signing service. The CompletedBy parameter is now also correctly filled in
CEP-8563	/	Custom legal notice issue has been fixed.
CEP-8583	/	Get Themes call issue when opening the WYSIWYS has been fixed.
CEP-8613 CEP-8553 CEP-8555 CEP-8614	/	Snippet issues in Config Index have been fixed.
CEP-8299	33469	Swagger inconsistency issue has been solved.
CEP-8432	/	It's now supported to select PDF_A_1A or PDF_A_2A as <b>TargetFormat</b> in API v4 when using the <b>FieldId</b> parameter to detect form fields.
CEP-8549	/	It's now supported to download the Audit Trail of packages with optional/refused documents.
CEP-8556	/	Audit Trail download issue of ended packages has been fixed.

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8557	/	Incorrect SigningTextFields issue for manual SigningMethods has been fixed.
CEP-8582	/	Back button issue with optional documents has been fixed.
CEP-8595	/	WYSIWYS language issue with optional documents has been fixed.

## 1.4 Known issues

### eSignatures 6.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8888	/	Itsme signing: When one signer of multiple signers has already signed, the subsequent signers (in a sequential or complex signing flow) receive 2 mail invitations to sign, instead of one. Signers and receivers also receive 2 mails to download the finished documents.
CEP-8406	/	Handwritten signature set to <b>Overlay</b> is currently not visible.
CEP-8530	/	Worker log issue.
CEP-8541	/	Preview issue in Config Index Theme section.
CEP-8543	/	PDF preview of optional documents is off center on mobile devices.

### eSignatures 6.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8322	/	iDIN signing issues with long legal notice.
CEP-8371	/	SMS OTP throttling error
CEP-8288	/	Users are not automatically added to the default user group.

### eSignatures 6.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8207	/	User permissions issue when user logs in for the first time.

### eSignatures 6.0.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8056	/	When IsAutomaticMembershipEnabled is disabled and a user who has been invited to log in, clicks the invitation link while already being logged in with another account in the same browser (for instance as initiator who sent the invitation link), they will not be able to log in, and the login of the other account will no longer work either.
CEP-6409	30654	Error message when uploading a corrupt PDF should be improved.

### eSignatures 5.5.3

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7554	/	Internet Explorer: Wrong color for signing method when choosing it for a Contact Group.

### eSignatures 5.5.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7078	/	It's currently not possible to sign with FranceConnect using eIDAS3.

### eSignatures 5.4.2

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6288	/	Branding issue: saving a change in one of the subtabs navigates to 1 tab above.

### Signatures 5.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5944	/	Itsme scrolling is not as smooth as expected on Safari on iPhone.

### 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.

### eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
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
/	/	German installation of Chrome can generate errors during signing.

## 1.6 Known limitations

### General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A single document must not contain more than 30 signing fields.
- 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.
- Adding multiple initiators within a single package is not supported.
- 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.
- 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.

### PDF files

- PDF portfolios are not supported. This is because a PDF portfolio may contain a wide range of file types that are not supported by eSignatures. A PDF portfolio may for instance contain e-mail messages, spreadsheets, CAD drawings, PowerPoint presentations, etc. As a result, a signer will only be able to view and sign the PDF cover sheet, and not the actual files the Portfolio contains, which renders the entire Portfolio invalid.
- Uploading PDF/A documents is only allowed if the format is PDF/A\_2A or PDF/A\_1A.
- Uploading PDF/A documents to which you add form fields in eSignatures API v4 breaks the "/A" part of the PDF. As a result, the documents end up a regular PDF.

As a workaround, you must set the **TargetFormat** parameter in the POST document call to PDF/A\_1A or PDF/A\_2A (depending on the source format). The breaking will still happen but after all form fields have been filled or saved, the document will be converted to the set target format, resulting in a PDF/A document again.

### Itsme signing

*(Note that these limitations do not apply when using itsme through OpenID Connect)*

- When using itsme as signing method, the target type of your documents must be **PDF/A\_1A** or **PDF/A\_2A**. 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, which means QuickSigning is not supported.
- When using itsme signing, the document and package names must only contain characters that are part of ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

#### iDIN signing

- Avoid using long legal notices combined with iDIN signing, especially when sending packages through the API. If long legal notices are used, the MerchantUrl exceeds its maximum number of 512 characters.

#### Mandated signing rules

- The mandated signing rules that have been configured in the Config Index cannot be overwritten using API v4 calls.

#### Download notifications

It may occur that Download notifications are still handled like spam / read by URL protection software, even after whitelisting them.

As a workaround, mail recipients must add the URL to the Microsoft Office 365 Advanced Threat Protection (ATP) **Do not rewrite the following URLs** list.

## 2. Upgrade Information

---

If you have a version of eSignatures installed prior to version 6.3, consult the **Connective - eSignatures 6.3 - Installation Documentation** to learn how to upgrade it to version 6.3.

---

# 1. Release Notes - eSignatures 6.3.2

Release date: 2020-11-17

## 1.1 New features

eSignatures 6.3.2 is a hotfix version and doesn't contain new features.

## 1.2 Improvements

### New Config Index setting: Automatic User Group Membership

Administrators can now configure in the Config Index to which user group newly created users must be added.

Note that since eSignatures 6.0, user creation is done in the Service Config Tool (SCT) and no longer in the eSignatures WebPortal. Once a user logs in to their account for the first time, or is invited by the admin, they are given the necessary permissions belonging to the group the admin has selected in the Config Index.

### New Config Index setting: Local Service Url in Plugin Settings

Depending on the SignID installer version that is used, the URL and/or port of the local SignID service differs. By adding the Local Service URL parameter, admins can switch easily between different versions, for testing purposes for instance.

## 1.3 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8711	34043	API v4: FormFiller process step issue has been fixed.
CEP-8709	4657	Incorrect validation rule issue on maximum length of external reference has been fixed.
CEP-8700	/	API v4: issue where finished packages did not return a download URL has been fixed.
CEP-8696	/	Issue where sequential signing order changed after saving has been fixed.
CEP-8685	/	F2F redirect URL issue has been fixed.
CEP-8664	/	Multipart specification issue when communicating with DSS has been fixed.
CEP-8682	/	Tenant Administrator rights issue has been fixed.
CEP-8778	/	Missing translations have been added.
CEP-8654	/	Configuration Administrators group has been removed.

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8650	/	Form fields scaling issue in WYSIWYS has been fixed.
CEP-8643	/	Upgrade issue with SMS settings has been fixed.
CEP-8621	/	Date formatting issue when doing a Get package by ID or Get stakeholder by ID call has been fixed.
CEP-8657	/	Parts of the SigningOptionsSettings in the Config Index were not correctly displayed.
CEP-8617	/	Connective logo has been removed from Maintenance screen.
CEP-8566	/	Issue where groups are incorrectly converted into single persons after signing has been fixed.
CEP-8558	/	Handwritten signature display issue has been fixed.
CEP-8288	/	Issue where users were not added to the default user group has been fixed.
CEP-8600	/	Config Index access issue when user belongs to a user group that has both the manage server and manage tenant permissions has been fixed.

## 1.4 Known issues

### eSignatures 6.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8888	/	Itsme signing: When one signer of multiple signers has already signed, the subsequent signers (in a sequential or complex signing flow) receive 2 mail invitations to sign, instead of one. Signers and receivers also receive 2 mails to download the finished documents.
CEP-8406	/	Handwritten signature set to <b>Overlay</b> is currently not visible.
CEP-8530	/	Worker log issue.
CEP-8541	/	Preview issue in Config Index Theme section.
CEP-8543	/	PDF preview of optional documents is off center on mobile devices.

### eSignatures 6.2.0



JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8371	/	SMS OTP throttling error

#### eSignatures 6.1.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8207	/	User permissions issue when user logs in for the first time.

#### eSignatures 6.0.1

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-8056	/	When IsAutomaticMembershipEnabled is disabled and a user who has been invited to log in, clicks the invitation link while already being logged in with another account in the same browser (for instance as initiator who sent the invitation link), they will not be able to log in, and the login of the other account will no longer work either.
CEP-6409	30654	Error message when uploading a corrupt PDF should be improved.

#### eSignatures 5.5.3

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7554	/	Internet Explorer: Wrong color for signing method when choosing it for a Contact Group.

#### eSignatures 5.5.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-7078	/	It's currently not possible to sign with FranceConnect using eIDAS3.

#### eSignatures 5.4.2

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-6288	/	Branding issue: saving a change in one of the subtabs navigates to 1 tab above.

#### Signatures 5.3.0

JIRA CODE	ISSUE CODE	DESCRIPTION
CEP-5944	/	Itsme scrolling is not as smooth as expected on Safari on iPhone.

#### 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.

#### eSignatures 5.2.0

JIRA CODE	ISSUE CODE	DESCRIPTION
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
/	/	German installation of Chrome can generate errors during signing.

## 1.6 Known limitations

### General

- A package must not exceed 150 MB.
- A single document inside a package must not exceed 30 MB.
- A single document must not contain more than 30 signing fields.
- 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.
- Adding multiple initiators within a single package is not supported.
- 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.
- 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.

### PDF files

- PDF portfolios are not supported. This is because a PDF portfolio may contain a wide range of file types that are not supported by eSignatures. A PDF portfolio may for instance contain e-mail messages, spreadsheets, CAD drawings, PowerPoint presentations, etc. As a result, a signer will only be able to view and sign the PDF cover sheet, and not the actual files the Portfolio contains, which renders the entire Portfolio invalid.
- Uploading PDF/A documents is only allowed if the format is PDF/A\_2A or PDF/A\_1A.
- Uploading PDF/A documents to which you add form fields in eSignatures API v4 breaks the "/A" part of the PDF. As a result, the documents end up a regular PDF.

As a workaround, you must set the **TargetFormat** parameter in the POST document call to PDF/A\_1A or PDF/A\_2A (depending on the source format). The breaking will still happen but after all form fields have been filled or saved, the document will be converted to the set target format, resulting in a PDF/A document again.

### Itsme signing

*(Note that these limitations do not apply when using itsme through OpenID Connect)*

- When using itsme as signing method, the target type of your documents must be **PDF/A\_1A** or **PDF/A\_2A**. 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, which means QuickSigning is not supported.
- When using itsme signing, the document and package names must only contain characters that are part of ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.
- Itsme signing is currently not supported on macOS Mojave v10.14 combined with Safari v12.0.

### iDIN signing

- Avoid using long legal notices combined with iDIN signing, especially when sending packages through the API. If long legal notices are used, the MerchantUrl exceeds its maximum number of 512 characters.

### Mandated signing rules

- The mandated signing rules that have been configured in the Config Index cannot be overwritten using API v4 calls.

### Download notifications

It may occur that Download notifications are still handled like spam / read by URL protection software, even after whitelisting them.

As a workaround, mail recipients must add the URL to the Microsoft Office 365 Advanced Threat Protection (ATP) **Do not rewrite the following URLs** list.

## 2. Upgrade Information

---

If you have a version of eSignatures installed prior to version 6.3, consult the **Connective - eSignatures 6.3 - Installation Documentation** to learn how to upgrade it to version 6.3.

---

# WebPortal User Documentation

This section contains the User Documentation of the eSignatures 6.3.x WebPortal.

# 1. Introduction

Welcome to the eSignatures 6.3.x user documentation.

This documentation consists of four parts, intended for

- **WebPortal users**
- **Form fillers**
- **Approvers**
- **Signers**

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

In the **Form fillers** part users will learn how to fill out forms in eSignatures.

In the **Approvers** part approvers will learn how to approve documents and packages.

In the **Signers** part signers will learn how to sign documents and packages.

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

---

**Important note:** eSignatures offers the same functionalities for documents and packages. 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
2020-01-13	DGI	Document creation
2020-01-20	DGI	Updated supported OS and Web browsers
2020-02-17	DGI	Update to version 5.5.2
2020-03-23	CJ	Update to version 5.5.3
2020-04-21	DGI	Updated FAQ on emails
2020-04-22	DGI	Added notes on package/document name, My Documents group and Themes in document groups
2020-07-14	DGI	Update to version 6.0.1
2020-09-10	DGI	Update to version 6.2.0
2020-10-09	DGI	Update to version 6.3.0
2020-11-16	DGI	Update to version 6.3.2

## 1.2 Copyright and legal notices

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.

© 2020 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 approvers, signers and receivers, configure signing methods, and sign documents in all possible ways. You'll also learn how to manage contacts and contact groups.



## 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 whether they should be approved before being sent to any signer(s), determine who should sign and which signing method they may choose from, 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 approvers and signers and define the order in which they should approve/sign. This way, your documents end up with the right person at the right time. The approvers and signers, on their end, can approve/sign their documents anywhere, anytime and on any device. They simply click the secured link in their email to open the documents, or log in to the Signer Portal using a web browser. A rebrandable Connective app for iOS and Android is also available.

Approvers and signers may also reassign the documents to someone else, if they feel they are not the right person to take action. (Note that this feature must have been enabled by the administrator to be able to reassign documents.)

### User roles

The eSignatures workflow contains 5 user roles:

**Initiator:** the eSignatures user who uploads the documents, and sends them for approval and signing.

**Form fillers:** the person who needs to fill out form fields on documents before they are signed by any signer(s).

The Form filler user role is new since eSignatures 6.3.

As of eSignatures 6.3, API users can create form fields, i.e. checkboxes and text fields, on the documents they're sending and specify which user must fill out the forms. The documents to be filled out can be accessed via email or in the Signer Portal, like any other documents to be approved or signed. Once the forms are filled out, the documents are sent to the required approvers or signers, depending on the configuration of the signing flow. Note that the form filling must not necessarily take place before the approval. In the current setup, it's entirely possible to approve documents of which the forms have not been filled out yet.

**Approver:** the person who needs to approve documents before they are sent to any signer(s).

**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 or through the eSignatures API. 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 6.3?

See the [eSignatures 6.3 - Release Notes - Public](#) for detailed information.

## 2.3 Accessing my account

2.3.1 How do I log in to my account?

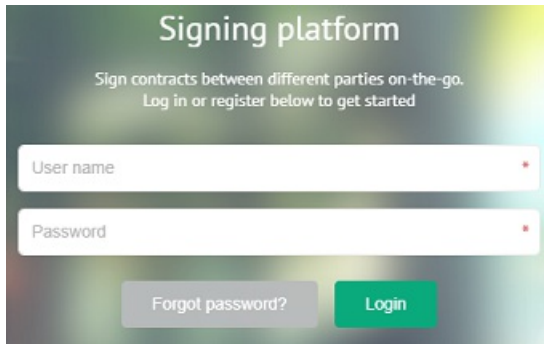
2.3.2 How do I log out?

2.3.3 How do I change my profile settings?

### 2.3.1 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 **User name** field.
- Enter your password in the **Password** field.

**Tip:** the password is the one that was assigned to you by your administrator.

The image shows a login interface for a 'Signing platform'. At the top, the title 'Signing platform' is displayed in a large, bold font. Below it, a subtitle reads 'Sign contracts between different parties on-the-go. Log in or register below to get started'. There are two input fields: 'User name' and 'Password', both with red asterisks indicating required fields. Below the 'Password' field, there is a link 'Forgot password?' and a green 'Login' button.

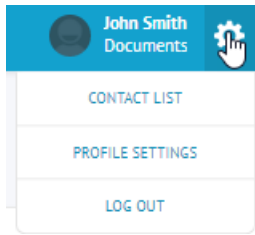
- Click **Login**.

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


**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.

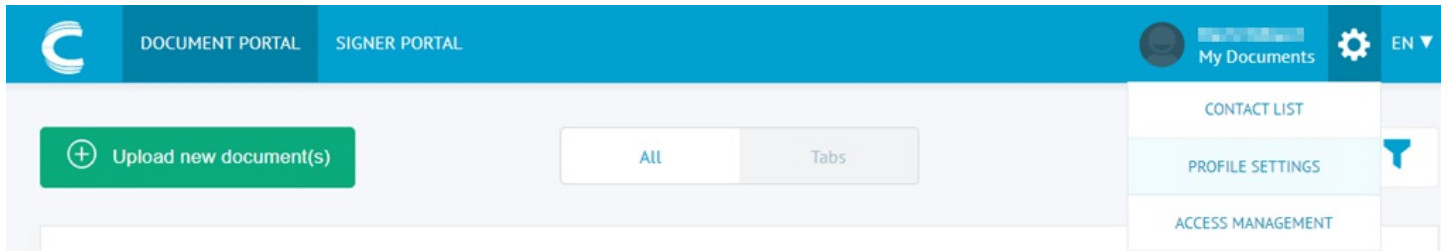
### 2.3.3 How do I log out?

- 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**.




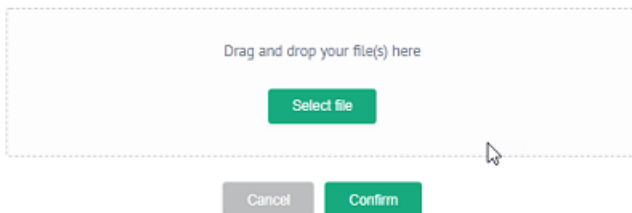
In profile settings you can do the following actions:

- Upload/change your profile picture
- Change your personal information
- Choose your language settings

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**.

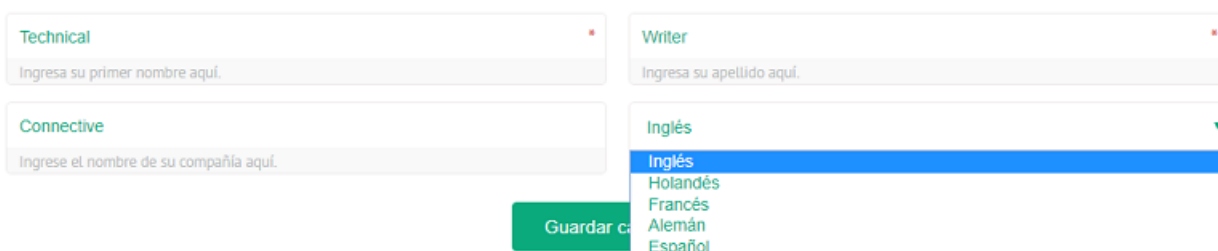
#### Change your personal information

- 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

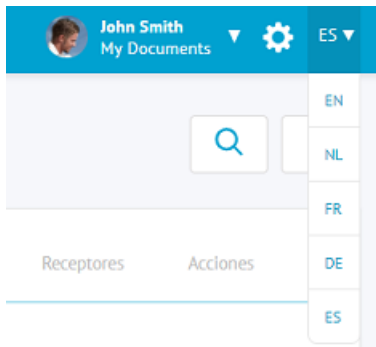
- Click the settings icon  in the top toolbar, and click **profile settings**.
- The preferred language was selected by the administrator when they created your account. If necessary, select another preferred language from the language drop-down list.

**Note:** the language you select here will also be the default language of each new contact you create, and the default language of each package you upload. Note however, that you can still override this default language on contact level and on package level.

The image shows a portion of the profile settings form. It includes three input fields: 'Technical' (with a red error icon and placeholder 'Ingresa su primer nombre aquí.'), 'Writer' (with a red error icon and placeholder 'Ingresa su apellido aquí.'), and 'Connective' (with placeholder 'Ingresa el nombre de su compañía aquí.'). To the right of these fields is a language drop-down menu currently set to 'Inglés'. The menu is open, showing a list of languages: 'Inglés' (highlighted in blue), 'Holandés', 'Francés', 'Alemán', and 'Español'. A green button labeled 'Guardar cambios' is partially 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.

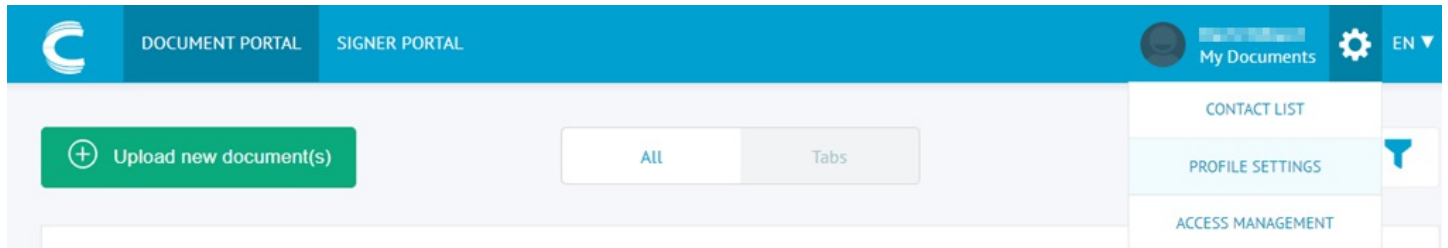


### 2.3.4 How do I manage my account?

If eSignatures is set up in an SSO environment, you can access your user account in Keycloak and manage its settings.

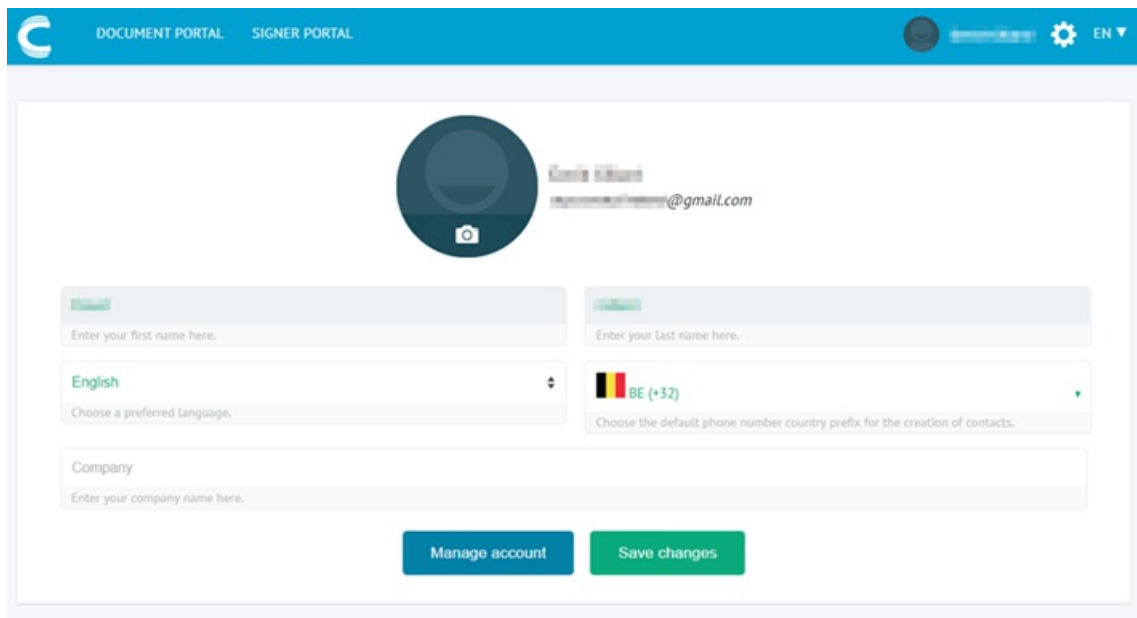
#### To manage your Keycloak account:

- Log in to eSignatures.
- Click the settings icon  in the top toolbar, and click **profile settings**.



- Click **Manage account**.

You are now redirected to your Keycloak account.



#### Edit your account data

On the **Account** tab you can modify your email, first name and last name.

- Once you've made the necessary changes, click **Sign out** in the top right corner.
- You can now sign in again to eSignatures using your new email address .

#### Important:

If documents had already been sent to your previous email address:

- They will no longer be visible in your Signer Portal when you log in using your updated email address.
- Any reminders the initiator sends to sign those documents are still sent to the previous email address.

#### Reset your password

On the **Password** tab you can reset your password.

- Enter your current password.
- Enter the new password.
- Then confirm the new password, and click **Save**.



- You now need to log in again, using your new password.

### Set up two time authentication

On the **Authenticator** tab you can set up two time authentication to log in to eSignatures and your Keycloak account.

Follow the on-screen instructions to do so.

## 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?](#)

[2.4.11 How do I end a document?](#)

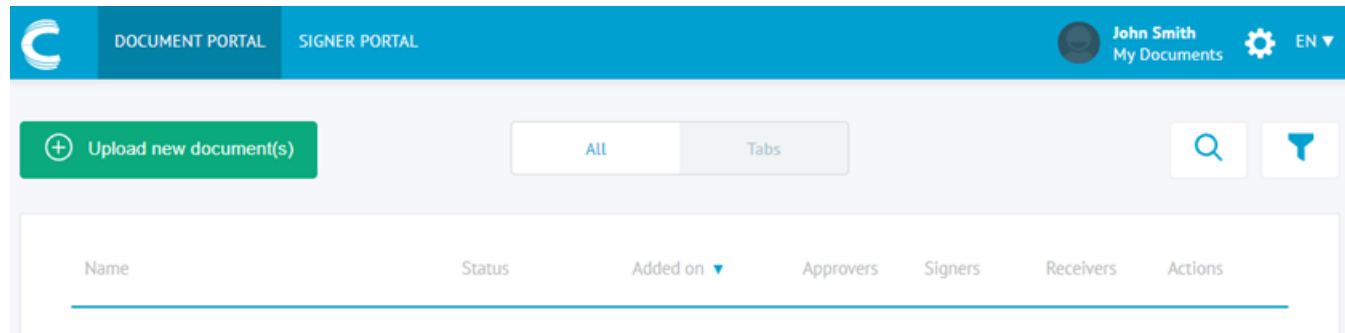
[2.4.12 How do I end an approval flow?](#)

[2.4.13 How do I download an Audit Trail PDF?](#)

### 2.4.1 Document Portal overview

The Document Portal is where you upload documents that need to be signed, determine who should approve them before being sent to any signer(s), determine who should sign them, which signing method they may choose from, 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 drop-down list next to your user name. **Tip:** when there's no down arrow, you only have access to one document group. 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 **Access 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**.

## 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.
- A document / package file name must contain between 1 and 150 characters.
- 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 that PDF portfolios are not supported. See [Why are PDF Portfolios not supported?](#) for more information. Also note that the administrator may not have enabled all file formats listed above in your eSignatures solution.
- Pay attention to the file name of your documents:
  - Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (\*), colon (:), pipe (|), quote ('), double quote ("), less than (<), greater than (>). Note however, that this list is not exhaustive.
  - Don't use characters that are HTML-sensitive such as ampersand (&) or apostrophe (').
- When using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.
- 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 and must not be used!

The document you have chosen contains markers

Continue using the available markers in the document



Confirm

- 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 **text fields** (created in Adobe Acrobat DC for instance). If the name of a such a text field corresponds to the **Text Field Marker 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 and keep their original text fields, make sure the name of the text field you create in your PDF Solution does not correspond to the **Text Field Marker 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 and must not be used!

### 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
- 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)
- 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 text 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 points high and 200 points wide will be placed. When adding a Text Marker, always put the Height value (minimum 70 points) in front of the Width value (minimum 112 points).

**Note:** it is recommended to use slightly higher values than the minimum ones. E.g. 75 points x 120 points. 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](#)

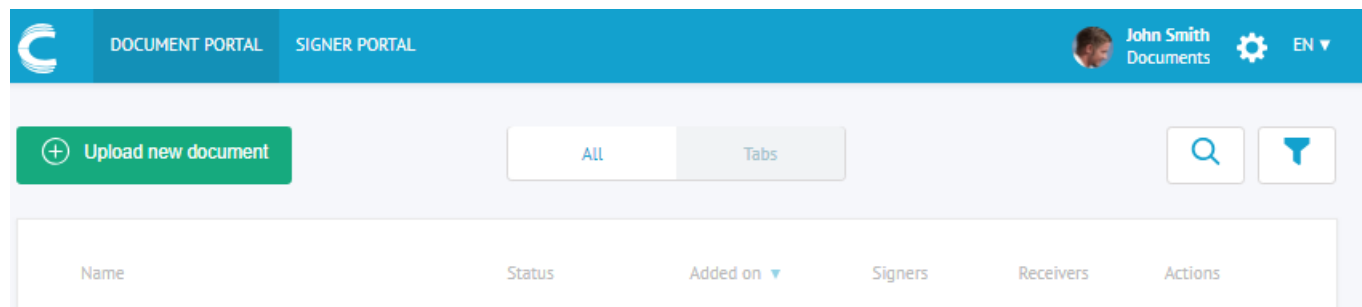
[Step 2 Define signing fields](#)

[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.

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.

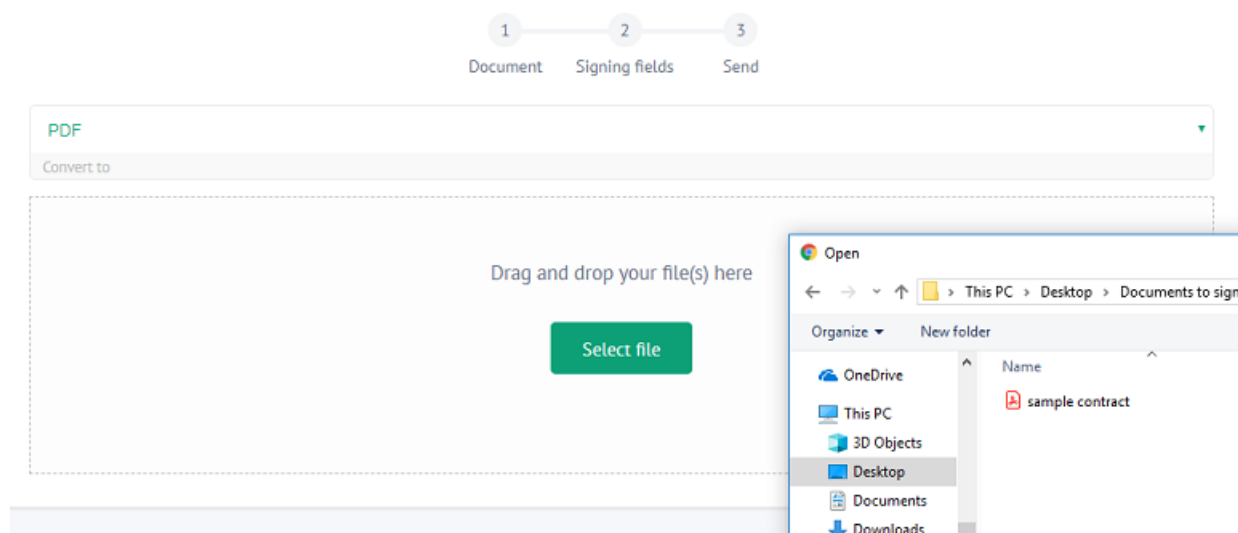
Pay attention to the file name of your documents:

- A document / package file name must contain between 1 and 150 characters.
- Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (\*), colon (:), pipe (|), quote ('), double quote ("), less than (<), greater than (>). Note however, that this list is not exhaustive.
- Don't use characters that are HTML-sensitive such as ampersand (&) or apostrophe (').
- When using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.

*Select files through the file explorer*

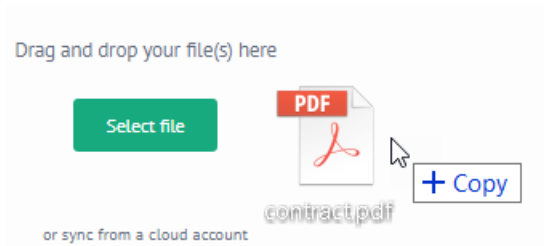
- Click **Select file**.
- In the window that opens, browse for the documents you want to upload.
- Ctrl-click or Shift-click the required files and click **Open**.

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



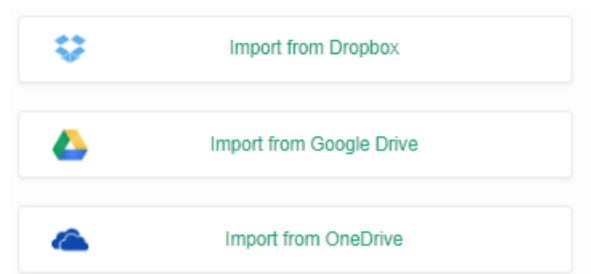
Select files by drag-and-drop

- 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 6.3.x - 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.

*This note does not apply when using itsme through OpenID Connect.*


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 text you can add to a document. They indicate where a signature field must be placed and which dimensions it must have.

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

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 features 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)

**Important:** Pay attention to the title you use:

- Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (\*), colon (:), pipe (|), quote ('), double quote ("), less than (<), greater than (>). Note however, that is list is not exhaustive.
- Don't use characters that are HTML-sensitive such as ampersand (&) or apostrophe (').

Select the document language

The document language is by default set to your preferred language (i.e. the language defined in your Profile Settings.)

If the document is drafted in another language, select the corresponding language from the list. When you choose to add legal notices, the default legal notices will be written in the language you select here.

Select the document group

Select the document group to which you want to upload the document. The default document group is **My Documents**.

Note that only you have access to the **My Documents** group. This group cannot be shared with other users.

**Important:** Never rename or delete the My Documents group! Doing so could lead to serious environment issues.

If multiple users should have access to the uploaded documents, make sure to upload them to a document group all required users have access to. See the [Access Management](#) section to learn how to configure the document groups permissions if necessary.

Select the theme

Select the theme in which the WYSISYS will be presented to the signer.

The themes that are available in this list, are the ones the administrator configured on the **Document group** tab of the [Access Management](#) section.

If no specific themes were configured, only the **System theme** is available.

1 Document 2 Signing fields 3 Send

SAMPLE\_CONTRACT.pdf  
Enter a package title (optional)

English  
Select the document language

My Documents  
Select a document group

System theme  
Select a theme

Do you want to add a legal notice?  
☐ 0

1 SAMPLE\_CONTRACT  
Click the document names to rename them

Add additional documents

## 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 multiple legal notices to a package, the content of each legal notice (per signer) must be identical for QuickSigning to work.

Itsme signing always requires a Signing Policy, which you normally can't combine with a legal notice. If you want to combine itsme signing with a legal notice, the setting **CombineLegalNoticeAndSigningPolicy** must be enabled in the Configuration Index.

- 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]

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.

## Determine which documents are optional

As of eSignatures 6.3, you can determine which documents within the package are optional.

- Enable the **Optional** button next to each document you want to mark as optional.
- When a document is marked as optional, signers won't be required to sign them. They can if they want to, but optional documents may also be left out of the signing flow.

123

DocumentSigning fieldsSend

package-Sketch System Requirements.docx

Enter a package title

English

Select the document language

My Documents

Select a document group

System theme

Select a theme

Do you want to add a legal notice?

☐ 0

Document nameOptional

Sketch System Requirements

☒

Productfiche (117361)

☐ 0

Add additional documents

→

## Add additional documents

- 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.

## Next step

- 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.

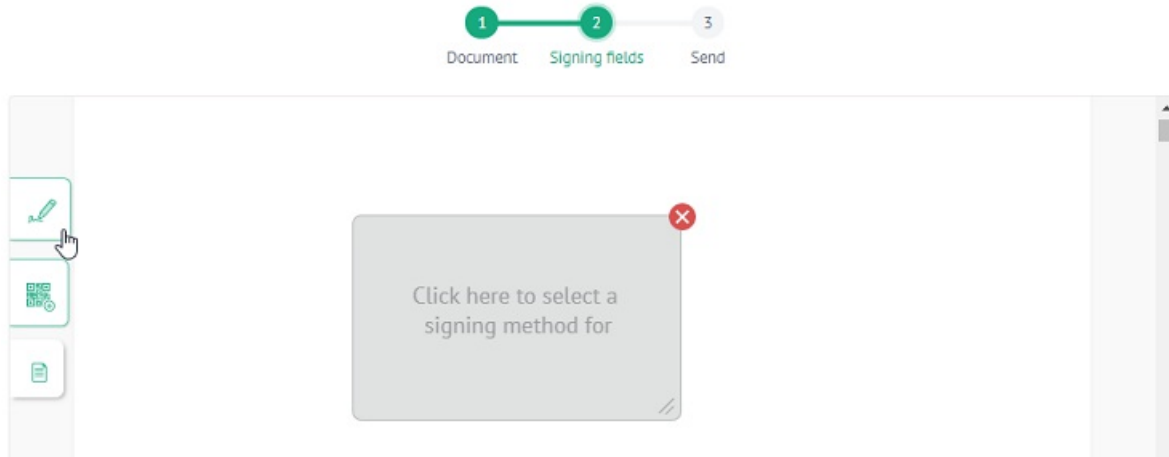
## Step 2 Define signing fields

### Add signing fields

- Scroll to the page where want to add the signing field.
- Click the signature icon to add a signing field.

**Important:** A single document must not contain more than 30 signing fields.

- 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.

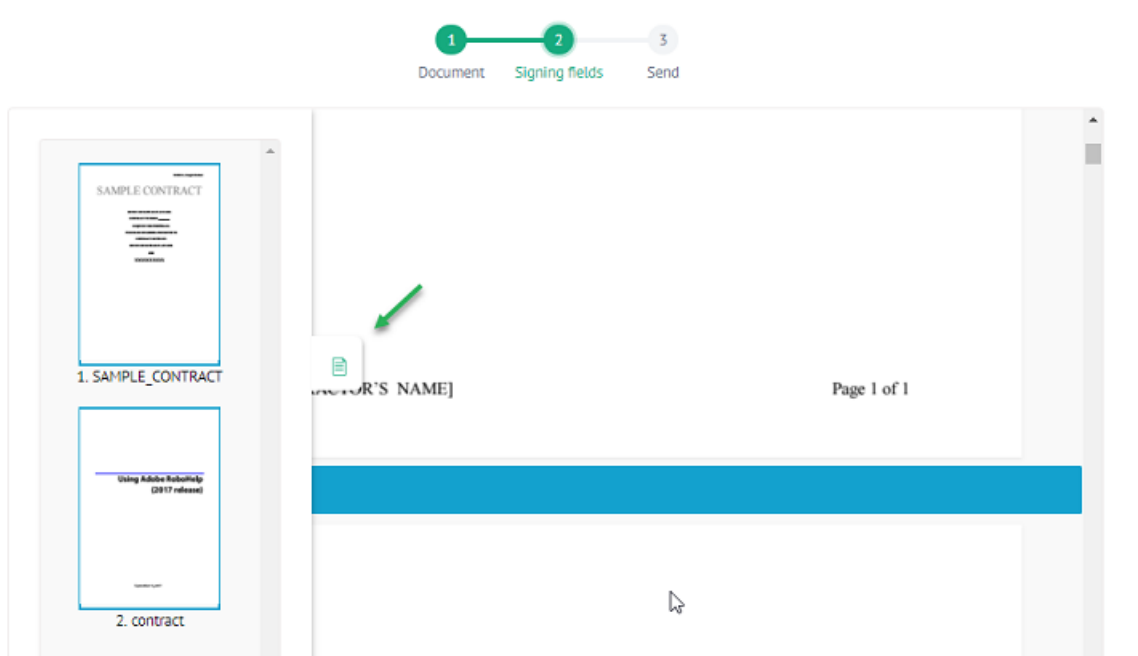
---

**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

You can either add a **contact** or **contact group** as signer.

When you add a *contact*, this specific contact is required to sign. When you add a *contact group*, any contact that is a member of the contact group is allowed to sign for the entire group. **Attention:** as soon as one member of the group has signed, the others no longer can.

- Click inside the signing field to add a signer.



- The available contacts and/or contact groups are displayed. Personal contacts/contact groups are marked by a fully colored icon. Shared contacts/contact groups are marked by a transparent icon. Cloud contacts are marked by a Google or Office 365 icon.
- Select the contact or contact group of your choice.



- To search for a specific signer, click inside the **Search name or add email address** field, and type in the name or email address of the signer you're searching.
- If no contacts are found, click the **Create new contact** button and enter all required fields (marked by an asterisk). Then click **Confirm**.

**Tip:** the text you entered in the **Search name or add email address** field is copied to the **Email** field when you click **Create new contact**.

**Create new contact**

Email <span style="float: right;">*</span> <small>Enter a valid email address.</small>	Personal title <small>Enter a personal title.</small>
First Name <span style="float: right;">*</span> <small>Enter first name here.</small>	Last Name <span style="float: right;">*</span> <small>Enter last name here.</small>
DD <span style="float: right;">-</span> <span style="float: right;">YYYY</span> <small>Day Month Year</small>	<div style="display: flex; align-items: center;"> <span>BE (+32) ▾</span> </div> <small>Enter a valid phone number to receive a one time SMS password.</small>
English ▾ <small>Choose a preferred language.</small>	
<div style="border: 1px solid #ccc; padding: 5px;"> <b>Additional Contact Fields</b>  <div style="text-align: center; margin-top: 10px;">No data to display</div> <div style="text-align: center; margin-top: 10px;"> <span style="background-color: #28a745; color: white; padding: 5px 10px; border-radius: 3px;">Add contact field</span> </div> </div>	
Make this a shared contact <input checked="" type="checkbox"/>	

Cancel
Confirm

**Important:** the email address must be unique. If another contact with the same email address already exist, you are prompted to merge it. See [How do I merge contacts?](#) for more information.

- Once a contact or contact group has been added, you can still edit the contact or members of the contact group, provided the administrator has given you the required permissions.

**Notes:**

- At this stage, you cannot add contacts to or remove them from the contact group you've selected. You can only edit the info of the contacts. Adding or removing contacts from the contact group must be done in the [Contact List](#) section.
- If no **Edit** button is displayed, this means you don't have the necessary permissions to edit the contact or contact group. Contact your administrator.

**Attention:** Be careful when changing a contact's email address. Documents that have already been sent to a contact's email address are not updated retroactively. So, they won't be sent automatically to the updated email address. Also, any reminders that are sent about existing documents will still arrive at the previous email address.

**Select Signer**

Mycontactgroup

Edit contact group members ✕

- 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.

Click the signing methods of your choice to select them.

**When a contact was added as signer:**

Any signing methods that have not been configured correctly, e.g. when a phone number, birthdate, national security number, etc. is missing, are greyed out cannot be selected. To solve this, edit the contact and add additional information (provided the **Edit**



button is available).

**Note:** which contact fields are mandatory depend on the mandated signing rules your system admin has applied.

### Select Signer

John Smith  
documentation.connective@gmail.com

Edit contact ×

### Select signing method for: John Smith

Manually	eID
One time SMS password	One time email password
Biometric	BeLawyer

Do you want to add a legal notice?

☐ 0

Signer must sign with the selected signing method

Cancel

Select

### When a contact group was added as signer:

A signing method that has not been configured correctly for any of the members, is marked is red when you try to select it.

### Select Signer

My contacts

Edit contact group members ×



None of the members in selected contact group can sign with the selected signing method(s)

### Select signing method for: My contacts

Manually	eID
Manually + eID	One time SMS password
One time email password	iDIN
Biometric	BeLawyer

A signing method that has not been configured correctly for *some* members is marked in orange and can still be selected. Note however, that in this case not all members will be able to sign the document.

To solve these issues, click the **Edit contact group members** button (if available).

### Technical Notes

- The birthdate is required when the system admin configured a mandated signing rule to check the birth date one of the following signing methods: eID, BeLawyer, iDIN or itsme, and you want the signers to use any of these signing types. In

these cases, the authenticity of the signer will be checked against their first name, last name and birth date during signing. So make sure the name of a contact is identical to the one on the signing certificate of their Belgian eID card, BeLawyer card, iDin 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, iDIN or itsme account.

- If you want the signer to be able to QuickSign their packages, the same choice of signing must be offered for each document of a package. It's not possible to QuickSign a package if you can choose from one pair of signing methods on one document, and choose from another pair on another document.

---

## Supported signing methods

Below you'll find an overview of the supported signing methods in eSignatures.

### Important:

- Your system admin may not have enabled all signing methods listed below in your eSignatures solution. If a desired signing method is not available, either the admin needs to enable and configure it in the Config Index, or the selected contact/contact group members don't contain the necessary info to use the signing method, as explained above.
- Since eSignatures 6.2, the **DisplayName** of the signing methods is configurable, so the name of the signing methods you see in your environment may not correspond to the ones listed below, but their behavior will be the same.

### **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.

### **BelD**

Select BelD 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.

**Important:** when the system admin applied a mandated signing rule to check the national security number, the contact's info must contain the national security number for the signing method to be available. If the contact's info doesn't contain all required data, the signing method will not be selectable.

### **Manually + BelD**

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.

**Important:** when the system admin applied a mandated signing rule to check the national security number, the contact's info must contain the national security number for the signing method to be available. If the contact's info doesn't contain all required data, the signing method will not be selectable.

### **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.

**Important:** the phone number of the signers must be known in order to use this signing method. If the contact's info doesn't contain a valid phone number, this signing method cannot be selected.

**Important:** if phone number confirmation is disabled in the Configuration Index, signers won't need to complete their phone number during signing. 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 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, 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 Finline 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.

**Important:** when the system admin applied a mandated signing rule to check the lawyerID, the contact's info must contain the lawyerID for the signing method to be available. If the contact's info doesn't contain all required data, the signing method will not be selectable.

## ***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. *This note doesn't apply when using itsme through OpenID Connect.*
- When using itsme signing in packages, each document within the package must be signed individually, which means QuickSigning is not supported.
- Itsme signing requires a Signing Policy, which you normally can't combine with a legal notice. If you do want to combine itsme signing with a legal notice, the setting **CombineLegalNoticeAndSigningPolicy** must be enabled in the Configuration Index. *This note doesn't apply when using itsme through OpenID Connect.*

### **Limitation:**

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

## **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.

---

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

Select signing method for: john smith

Manually	eID
Manually + eID	SMS OTP
Email OTP	iDIN
Biometric	BeLawyer
Itsme	PINCODE

Do you want to add a legal notice?

☒

Read and approved

Select the preferred legal notice.

Read and approved

Edit the legal notice.

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.

**Tip:** the legal notice settings you apply on signer level, are retained in any subsequent signing field you add for the same signer in the package. When you add a new signing field for a different signer, the legal notice settings configured on Package level will be applied, and still need to be customized (if necessary).

**Important:** if you do want to combine itsme signing with a legal notice, the administrator must enable the setting **CombineLegalNoticeAndSigningPolicy** in the Configuration Index. *This last note doesn't apply when using itsme through OpenID Connect.*

#### To configure a legal notice on signer level:

- 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.

Read and approved

Read and approved

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

Guarantor (able 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.

×

1 2 3

Legal notice

Signing field 1 of Addendum

Fill out the legal notice below. (Note: the field is case sensitive.)

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

Next

#### 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 multiple legal notices to a package, the content of each legal notice (per signer) must be identical for QuickSigning to work.

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

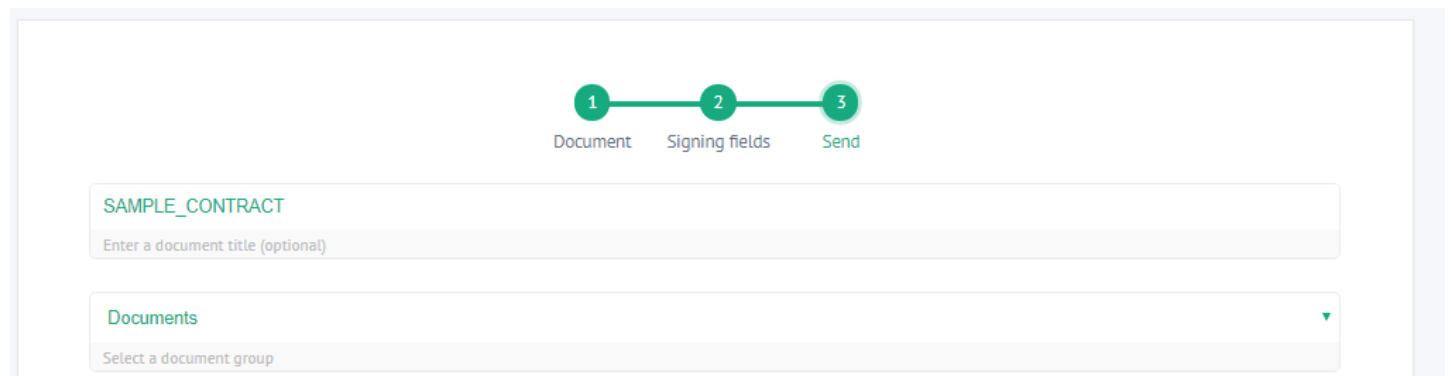
Sending the documents is the final step in the upload process. All the settings you can configure in this screen are optional.

#### Enter a title

If necessary, you can still modify the title here.

**Important:** Pay attention to the title you use:

- Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (\*), colon (:), pipe (|), quote ('), double quote ("), less than (<), greater than (>). Note however, that this list is not exhaustive.
- Don't use characters that are HTML-sensitive such as ampersand (&) or apostrophe (').
- When using iSigner signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.



The screenshot shows a three-step process flow at the top: 1. Document, 2. Signing fields, and 3. Send. Below this, there are two input fields. The first field is labeled 'SAMPLE\_CONTRACT' and has a placeholder text 'Enter a document title (optional)'. The second field is labeled 'Documents' and has a placeholder text 'Select a document group'. Both fields have a dropdown arrow on the right side.

#### Select the document group

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

**Note:** whether you have access to different groups depends on your user permissions.

#### Select the theme

Select the theme in which the WYSISYS will be presented to the signer.

The themes that are available in this list, are the ones the administrator configured on the **Document group** tab of the **Access Management** section.

If no specific themes were configured, only the **System theme** is available.

#### Add approvers and arrange their order

If the document or package must be approved before sending it to any signer(s), you can add one or more approvers.

The order in which approvers must approve can be configured in the same way as signers. Approvers may approve in parallel, sequentially or in a complex approval flow. See **Arranging the signers** below for more information.

#### To add approvers:

- Click inside the **Search name or add email address** field.
- The available contacts and contact groups are displayed. Personal contacts/contact groups are marked by a fully colored icon. Shared contacts/contact groups are marked by a transparent icon. Cloud contacts are marked by a Google or Office 365 icon.

## Approvers

+ Create new contact

Jane Jefferson

John Smith

- To search for a specific approver, type in the name or email address of the approver you're searching.
- If no contacts are found, click the **Create new contact** button and enter all required fields (marked by an asterisk). Then click **Confirm**.

**Tip:** the text you entered in the **Search name or add email address** field is copied to the **Email** field when you click **Create new contact**.

**Important:** the email address must be unique. If another contact with the same email address already exist, you are prompted to merge it. See [How do I merge contacts?](#) for more information.

- Repeat the steps above to add additional approvers.

**Note:** when you select a **contact group** as approver, all members of the group will receive a link to approve the document. As soon as one approver has approved however, the others no longer can't.

When you assigned multiple approvers, you can decide whether they can **approve in parallel**, **approve sequentially** or design a **complex** approval flow. This functions in the same way as arranging signers (explained below).

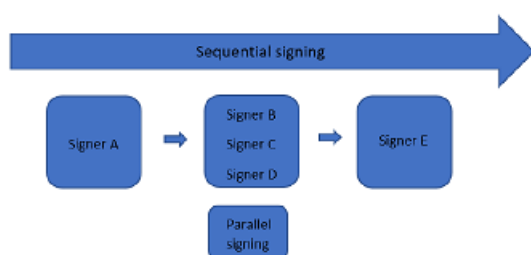
### Arrange the signers

When you 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 their 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 : Signer A must sign the package first. Once signer A has signed, signers B, C and D must sign the package. The order in which they sign doesn't matter, as long as they all sign. When all three signers have signed, signer E needs to sign to give their final approval. In this example, signers 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

Sign in parallel

Sign sequentially

Sign complex

1	John Smith documentation.connection@gmail.com		✕	≡
2	Jane Jefferson documentation.connective2@gmail.com		✕	
	marker	👤	✕	
	marker	👤	✕	

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

Sign in parallel

Sign sequentially

Sign complex

▼ Step 1

Technical Writer  
documentation.connective1@gmail.com

markers

▼ Step 2

John Smith  
documentation.connective@gmail.com

markers

▼ Step 3

Jane Jefferson  
documentation.connective3@gmail.com

markers

+ Create a new step by dropping here

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

### Signers

Sign in parallel

Sign sequentially

Sign complex

▼ Step 1

Technical Writer  
documentation.connective2@gmail.com

markers

▼ Step 2

John Smith  
documentation.connective@gmail.com

markers

Jane Jefferson  
documentation.connective3@gmail.com

markers

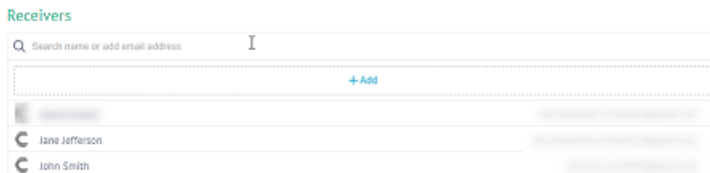
+ Create a new step by dropping here

**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 name or add email address** field. The available contacts and/or contact groups are displayed.
- The available contacts and contact groups are displayed. Personal contacts/contact groups are marked by a fully colored icon. Shared contacts/contact groups are marked by a transparent icon. Cloud contacts are marked by a Google or Office 365 icon.
- Select the contact or contact group of your choice.



- To search for a specific receiver, type in the name or email address of the receiver you're searching.
- If no contacts are found, click the **Create new contact** button and enter all required fields (marked by an asterisk). Then click **Confirm**.

**Tip:** the text you entered in the **Search name or add email address field** is copied to the **Email** field when you click **Create new contact**.

**Important:** the email address must be unique. If another contact with the same email address already exist, you are prompted to merge it. See [How do I merge contacts?](#) for more information.

- Repeat the steps above to add additional receivers.

**Note:** when you select a **contact group** as receiver, all members of the group will receive a link to the fully signed document.

## 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 approver/signer again, the initiator will have to [extend the expiry date](#) in the Document Portal.

**Note:** the link sent to the approvers and signers to approve/sign their document, and the link to download a signed document are by default [one-time URLs](#) (unless disabled by the administrator). This means the links can only be used once, even when you don't set an expiry date. If you try to use a link a second time, the system will prompt you to request a new email containing a new link.

## Enable or disable the download of the unsigned documents

Depending on the configuration of your environment, you may be able to enable or disable the download of unsigned documents in the WYSIWYS.

Enable or disable the download of the unsigned documents.



- Set this setting to **enabled** if you want approvers/signers to be able to download documents before signing and read them in a PDF Viewer or even on paper for instance.
- Set this setting to **disabled** to hide the download icon in the WYSIWYS.

**Note:** this button is only available if the setting **IsDownloadUnsignedFilesVisibleInFrontEnd** is set to enabled in the Configuration Index.

### Enable or disable the reassign feature

Depending on the configuration of your environment, you may be able to enable or disable the reassign feature. When enabled, approvers and signers may reassign a document to someone else when they feel they're not the right person to take action.

Allow signers and approvers to reassign the package



- Set this setting to enabled if you want approvers/signers to be able to reassign documents.
- Set this setting to disabled to hide the **Reassign** button in the WYSIWYS.

**Note:** this button is only available if the setting **IsReassignVisibleInFrontEnd** is set to enabled in the Configuration Index.

### Enable or disable the expiration of action URLs for this package

The action URLs users receive from eSignatures, to sign their package face-to-face, to approve or sign a package or to download a signed package are by default one-time URLs, which means they can only be used once. When the URLs are not used however, they remain valid indefinitely or until the package expires.

As an additional security measure, you can now set an expiration period for these URLs, provided the admin enabled this feature in the configuration. This way, you can have a package that is valid for 30 for instance, while its action URLs must be renewed every other day.

To set an expiration date, enable this setting and enter the number of days after which the URLs must expire.

**Note:** if you don't configure this setting, or if the setting isn't visible, the expiration behavior that has been configured on environment level in the Config Index applies.

### 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 persons who need to act on the document (i.e. approve or sign) will receive a download link to the document.

**Important:** as initiator you can also have signers **sign immediately** (provided no approvers have been assigned). 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.

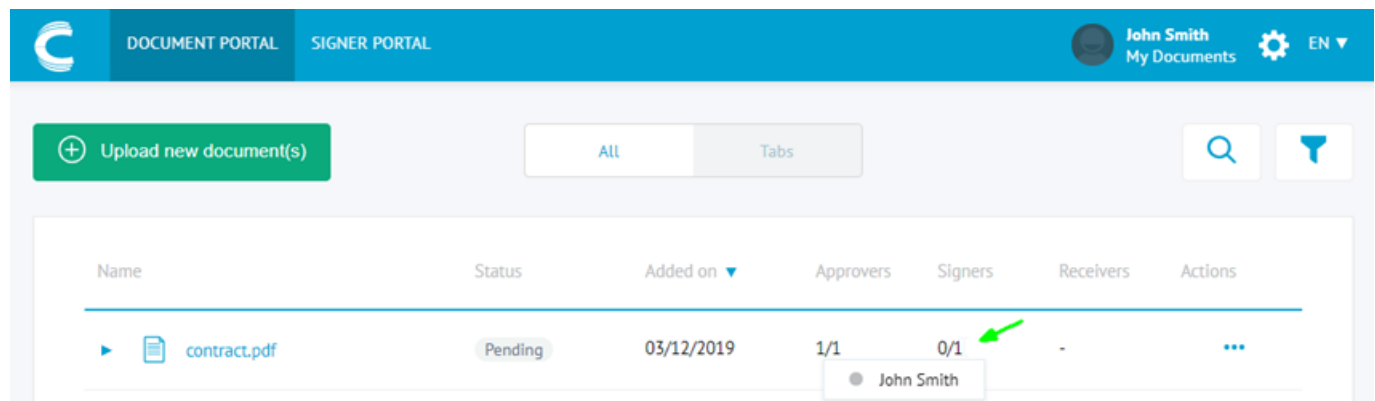
- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.

##### 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, how many approvers have approved, how many signers have signed, how many receivers they've been sent to.

**Tip:** hover over the number of approvers/signers/receivers info to see their names. The persons who completed their action are marked by a green bullet. The ones who didn't are marked by a grey bullet.

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



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 signing method that was used is marked by a green icon. The signing methods that were offered, but not used are marked in grey.

The different approvers, signers and receivers (if any) are displayed in a logical order:

- Receivers are always sorted alphabetically by last name, then by first name.
- Approvers/signers are sorted based on the approval/signing flow defined by the initiator, either in the WebPortal or in the API:
  - **Parallel:** approvers/signers are always sorted alphabetically by last name, then by first name.
  - **Sequential:** approvers/signers are sorted in the defined order.
  - **Complex:** approvers/signers are first sorted in the defined order. If multiple approvers/signers may approve/sign during the same step of the flow, they are sorted alphabetically by last name, then by first name.

DOCUMENT PORTAL

SIGNER PORTAL

John Smith  
My Documents

EN

Upload new document(s)

All

Tags

Name	Status	Added on	Approvers	Signers	Receivers	Actions
<div><div></div><div>contract.pdf</div></div>	Pending	03/12/2019	1/1	1/2	-	<div></div>
Initiator	<div></div>		Document added		03/12/2019 12:10	
Approvers	<div><div></div>Jane Jefferson</div> <div>Approved date 03/12/2019 12:11</div>		<div></div>			
Signers	<div><div></div>John Smith</div> <div>Signed date 03/12/2019 12:12</div> <div><div></div>Mycontacts</div>		<div><div></div><div></div><div></div></div>			
Receivers	None					

The following details are available for the different states:

### Draft

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers (if already defined)
- Signing methods (if already defined)
- Receivers (if applicable)

### Pending

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Approvers (if applicable)
- Date when the document was approved or rejected (if applicable)
- Reason for rejection (if applicable)
- Signers
- Signing methods the signers may choose from, or have to use
- Date when the document will expire (if applicable)
- Receivers (if applicable)

### Revoked

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing methods signers could choose from
- Date when the document was revoked
- Receivers

### Rejected

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing methods signers could choose from
- Receivers
- Signer who rejected the document
- Date when the document was rejected
- Reason for rejection

### Expired

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing methods signers could choose from
- Date on which the document expired
- Receivers

### Finished

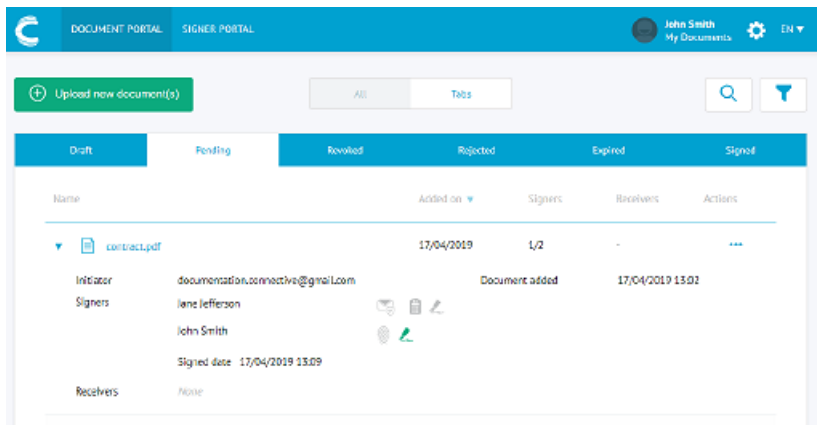
- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing methods signers could choose from
- Signing method the signers used
- Date when the document was signed (if applicable)
- Date when the document was refused (if applicable)
- Receivers

### Failed

- Whether the document is optional
- Initiator (who sent the document)
- Date when the document was added
- Signers
- Signing methods signers could choose from
- Signing method signers used
- Receivers

### Tabs view

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



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 approval/signing.

**Revoked:** the document had been sent but was canceled by the initiator. The document can no longer be approved or signed.

**Rejected:** the document has been sent but an approver or one or more signers rejected it. In this case the document is no longer available.


**Expired:** the document has been sent but was not approved or signed before the expiration date.

**Finished:** the document has been sent and all users have taken the necessary action.

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


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.

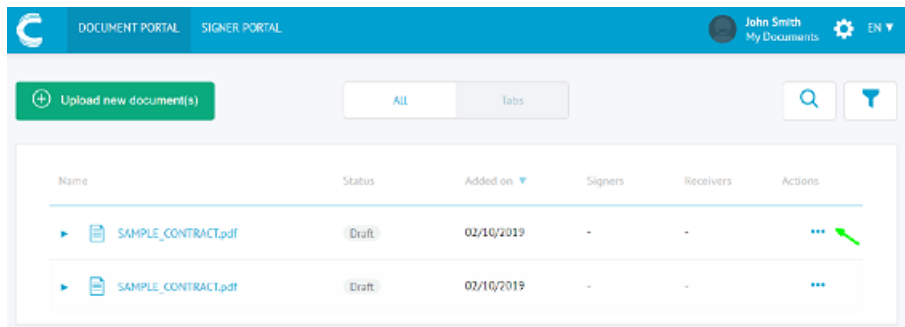
## 2.4.5 How do I edit a draft document?


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 their 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.

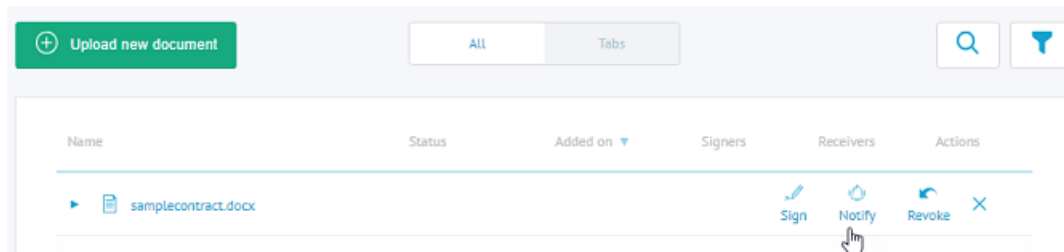


## 2.4.6 How do I send a reminder to an approver or signer?

As initiator you can send a reminder to approvers or signers, reminding them to take action on the document.

### To send a reminder:

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



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

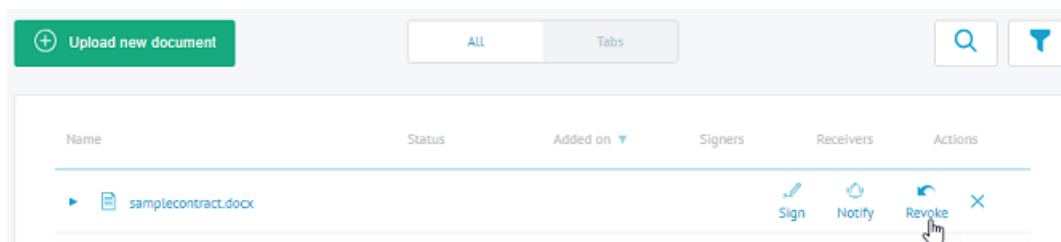
## 2.4.7 How do I revoke a document?

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

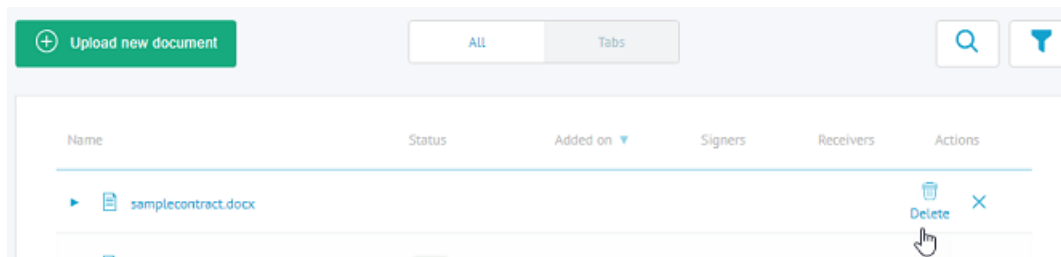
## 2.4.8 How do I delete a document?

A document can be deleted when it's in draft, signed, rejected or revoked status.

Note that you can only delete a document of which you are the initiator.

### 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

CancelConfirm

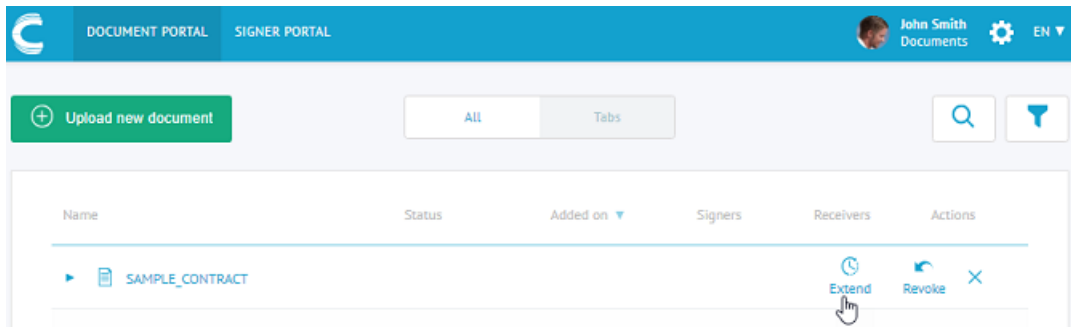
**Note:** documents or packages are never deleted automatically. They are stored indefinitely if they are not deleted manually.

## 2.4.9 How do I extend the expiration date?

When a document has expired - meaning when the approver did not approve it or 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**.

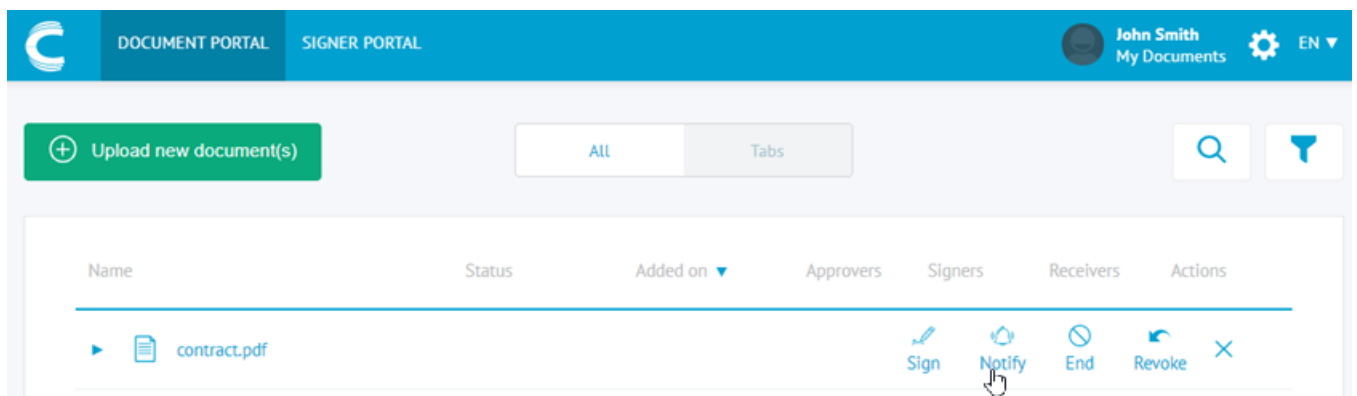
Extend expiry period

Pick a date.

Hour

Minutes

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

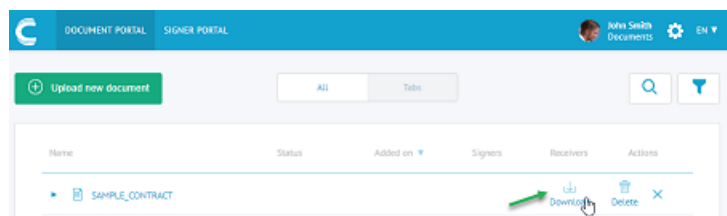


## 2.4.10 How do I download a document?

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.

### Notes:

- A package is always downloaded as .zip file. You'll need to unzip it to view its contents.
- Any unsupported characters your package may contain are escaped during download. Therefore, they might have a slightly different file name than the one chosen by the initiator.

## 2.4.11 How do I end a document?

As initiator you can now choose to end a document or package as soon as it has been signed by at least one party.

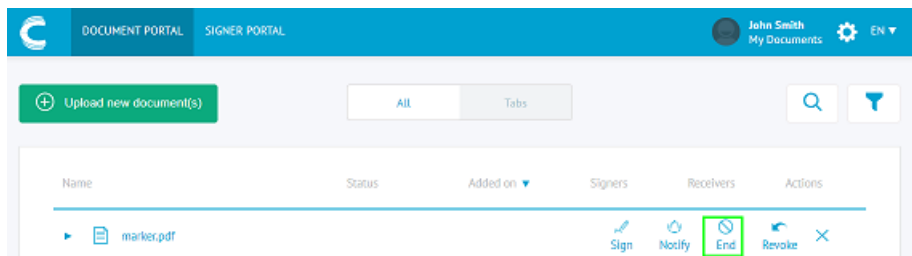
This way, the document becomes available for download and it is no longer necessary to wait for all parties to sign before anyone is able to download the document.

The parties that will be able to download the document are: the initiator, the signer(s) who actually signed and the receivers.

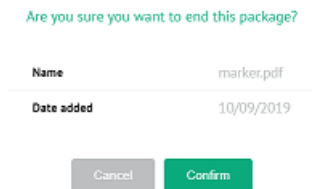
Any signers that were asked to sign but haven't signed yet, will receive an email that the document has been ended by the initiator. They can no longer do anything with the document.

### To end a document:

- [Log in](#) to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document you want to end, and click the menu button **...** in the **Actions** column.
- Click **End**. Note that the **End** button is only available if the document must be signed by multiple signers, and at least one of them has signed.



- Click **Confirm** in the pop-up window.



The status of the document now changes to **Signed** in the Document Portal and the document is available for download to the signer and to the initiator.

**Note:** if your eSignatures environment is using Audit proofs, and an initiator ends a document in the Document Portal, it is added to the Audit proofs which initiator ended the document.

## 2.4.12 How do I end an approval flow?

As initiator you can choose to end an approval flow as soon as a document has been approved by at least one of multiple approvers.

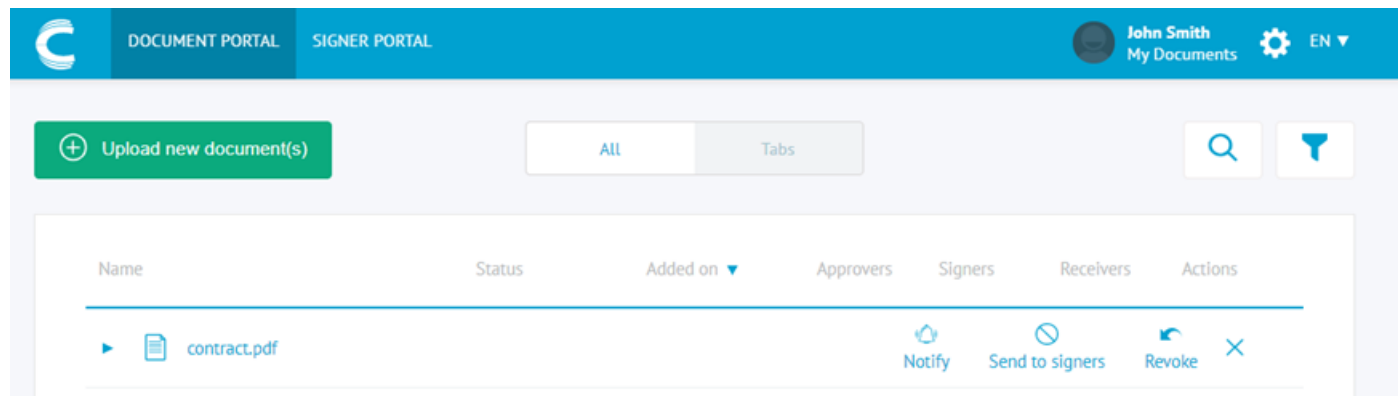
This way, the document becomes available for signing and it is not necessary to wait until all approvers have approved.

Any approvers that were asked to approve but haven't done so yet, will receive an email that the document has been ended by the initiator. They can no longer do anything with the document.

### To end an approval flow:

- Log in to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document whose approval flow you want to end and click the menu button in the **Actions** column.
- Click **Send to signers**.

Note that the **Send to signers** button is only available if the document must be approved by multiple approvers, and at least one of them has approved.



- Click **Confirm** in the pop-up window.

The status of the document now changes to **Pending** in the Document Portal and the document is available for signing.

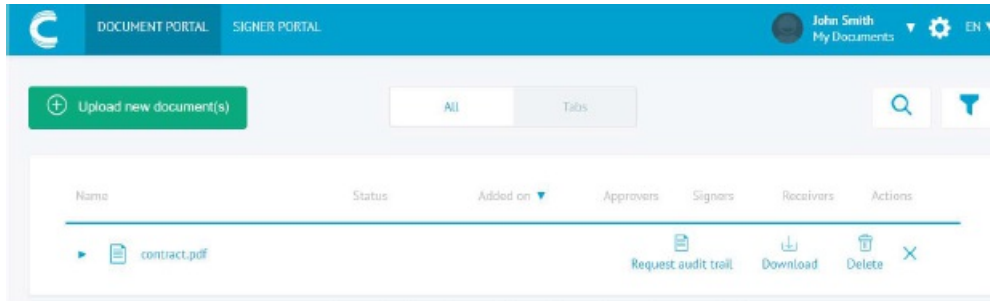
**Note:** if your eSignatures environment is using Audit proofs, it is added to the proofs which initiator ended the approval flow

## 2.4.13 How do I download an Audit Trail PDF?

As initiator you can download the audit trail of packages that have a finished state in the document portal .

### To download an audit trail:

- Log in to your eSignatures account.
- Click the **Document Portal** tab.
- Go to the document you want the audit proof from, and click the menu button in the Actions column.
- Click the **Request audit trail** 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:

- The functionality must be active for your environment
- The 'Download Audit Trail' is a user permission that needs to be granted to your document group by your system administrator

Below an example of the Audit Trail



PACKAGE ID	ef1049f2-cf93-455a-8633-9a8e50fc1912		
PACKAGE NAME	Package		
CREATED ON	Friday, July 31, 2020		
INITIATOR	[Redacted]		
PACKAGE STATUS	FullySigned		

SENT FOR APPROVAL 	Friday, July 31, 2020 8:25 AM	
	Sent for approval to	Signer Test2
		Signer Test2
VIEWED 	Friday, July 31, 2020 8:25 AM	
	Viewed by	Signer Test2
APPROVED 	Friday, July 31, 2020 8:25 AM	
	Approved by	Signer Test2



## 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 sign face to face?](#)

2.5.4 [How do I reject a document?](#)

2.5.5 [How do I reassign a document?](#)

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

2.5.7 [How does mandated signing work?](#)

### 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.

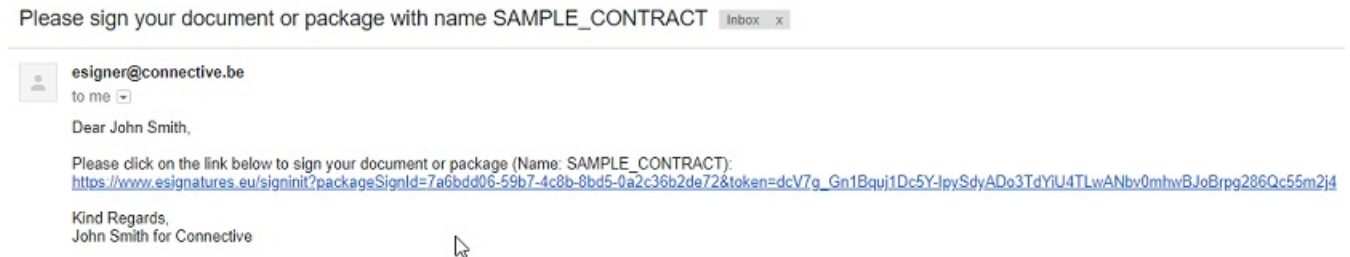
**Important:** the standard Connective app available in the App store and Google Play only works with the **standard demo environment** of eSignatures.

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](mailto:esigner@connective.be).
- Click the secure link inside the email. The document will open in a new tab in your default web browser.

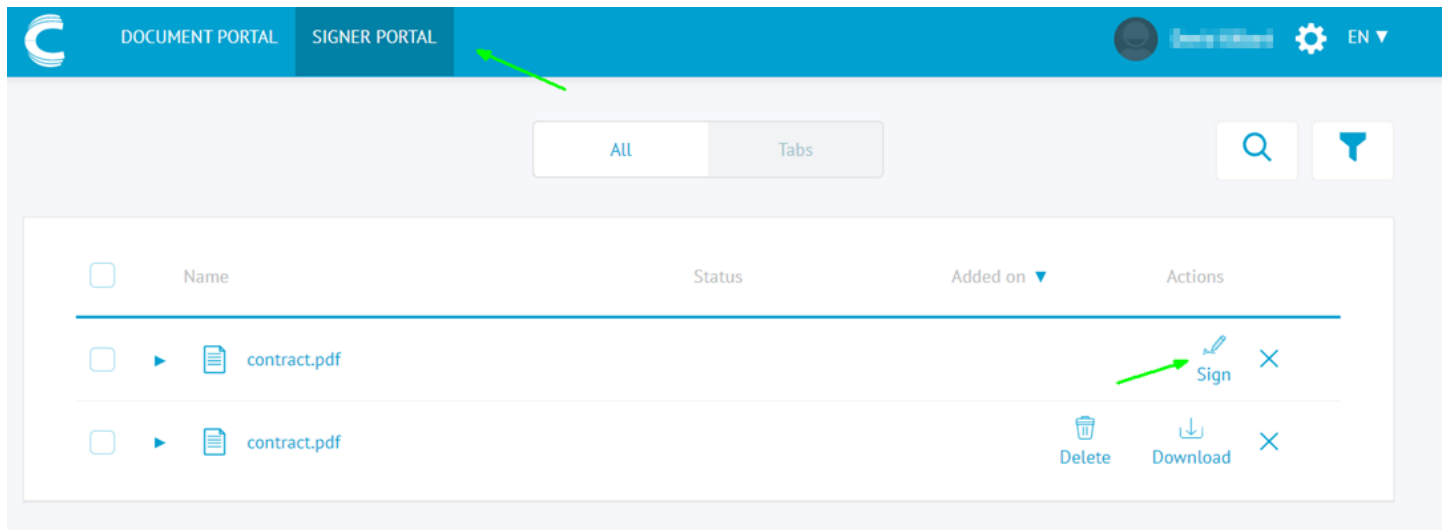
**Important:** this link will only work once. Once you've clicked it, you need to sign or reject the document. If the link expired, click **Request a new email** to receive a new link.



- 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.



- 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

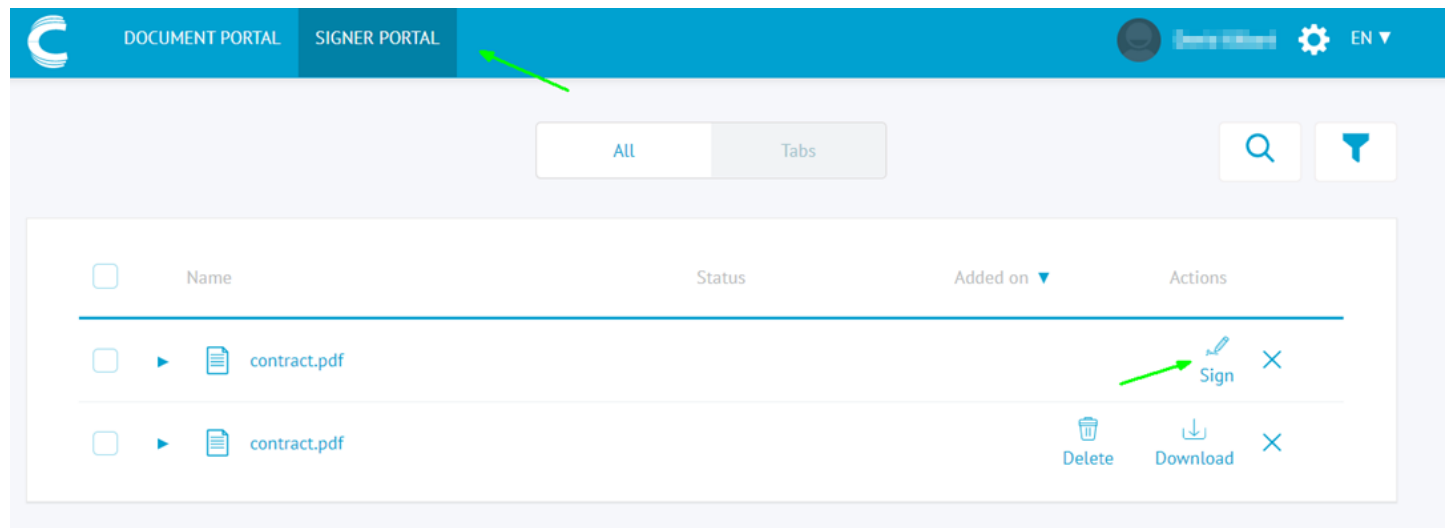
## Deleting a document from the Signer Portal

Signers can now delete documents from their Signer Portal, provided the documents have been signed by all signers or rejected by at least one signer.

Note that deleting a document from a Signer Portal does not delete it from the initiator's Document Portal.

### To delete a document from the Signer Portal:

- Log in to your eSignatures account.
- Click the **Signer Portal** tab.
- Go to the document you want to delete, and click the menu button **...** in the **Actions** column.
- Click **Delete**.



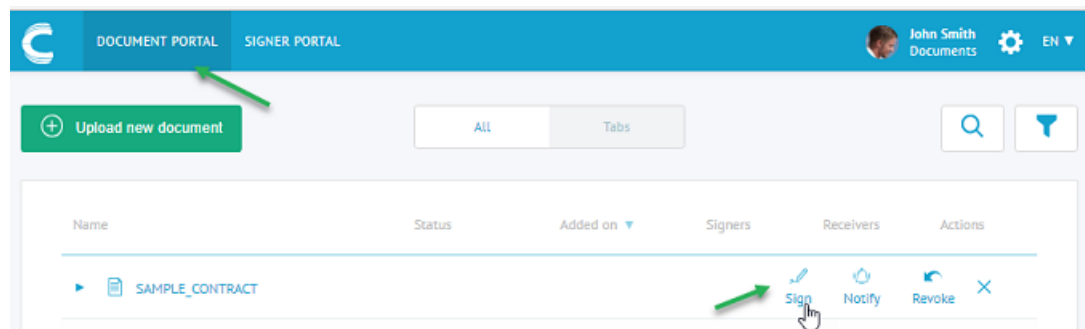
- Then click **Confirm** in the pop-up window.

## 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.
- The different signers are displayed alphabetically by last name in the **Start signing** window.

**Note:** if multiple signers have the same last name, they are sorted by first name.

- Click **Sign all** if all signing parties are present.
- Click **Sign** next to the required signer, if only that signer is present.
- Click **Select signer** next to a contact group, if only one member of that contact group is present. Then, click **Sign** next to the signer that is present. **Important:** only 1 member of the contact group can sign. If the document must be signed by multiple signers, you should add them separately and not in a document group.

**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.

## Start signing

Select a signer below

Sign immediately as John Smith

Sign

Sign immediately as member of group Mycontactgroup

Select signer



Done

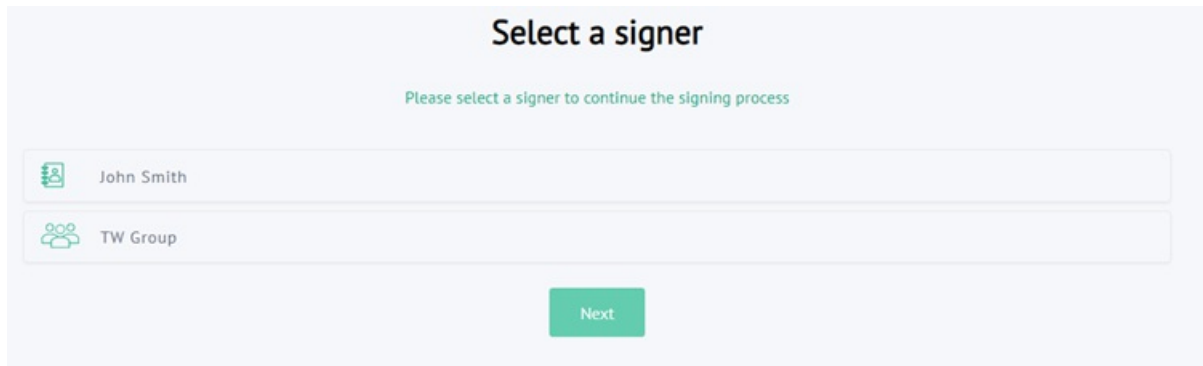
Sign all

## Using Sign all

In this section we cover the **Sign all** scenario. When you select a single signer, the signing steps are identical to a regular signing flow.

- Click **Sign all** to start a face to face signing flow for all signers.
- The required signers are listed as shown below.

- Contact groups are marked by a group icon .
- Single contacts are marked by a notebook icon .



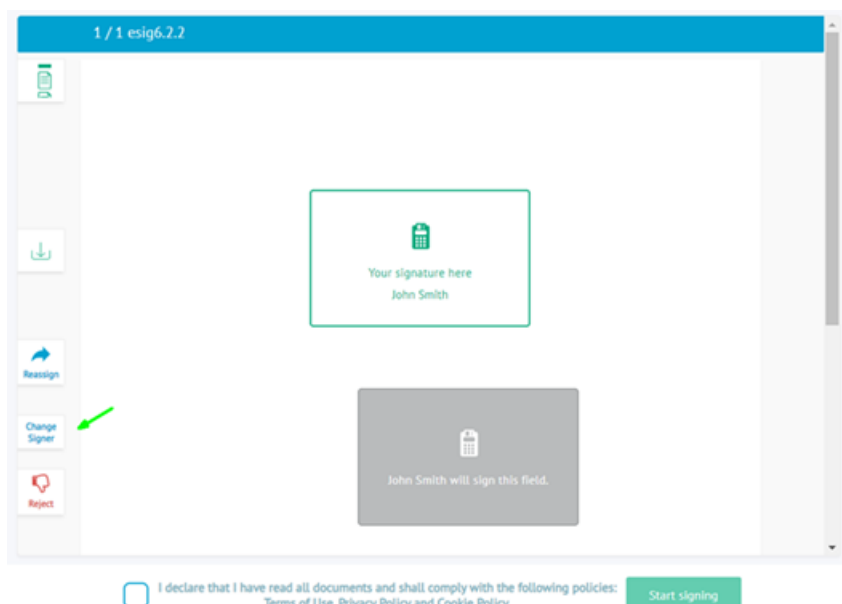
- Click the signer you want to start with and click **Next**.

#### To sign the document:

- 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**.
- You can find the different signing steps in [Signing a document step-by-step](#).
- When the signing is in progress for the current signer, you can click **Next Signer** to go to the next one (provided you're using a signing method that supports asynchronous signing).

#### To change the signer:

- If you selected the wrong signer by mistake and want to start with another signer, click the **Change Signer** button.
- You are now redirected to the face to face signing screen where you can select a different signer.



#### To reassign the document:

If the signer thinks they are not the right person to approve or sign a document, you can reassign it to someone else.

**Attention:** whether you are able to reassign documents depends on the configuration of your eSignatures solution. If the administrator has not enabled this feature, you won't be able to reassign.

- Click the **Reassign** button on the left-hand side of the document.
- Fill in the new signer's personal info and click **Reassign**.

For detailed info, see [How do I reassign a document?](#)

- Once the document has been reassigned, you can click the **Change Signer** button to continue with another signer
- When you're redirected back to the face to face signing screen, you'll see the signer to whom the document has been reassigned as available signer.

#### **To reject a package:**

- Click the **Reject** button on the left-hand side of the document.
- Enter a reason for rejection and click **Reject**.
- The signing flow ends here. The other signers can no longer sign the document.



## 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 [Signers](#) 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 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.

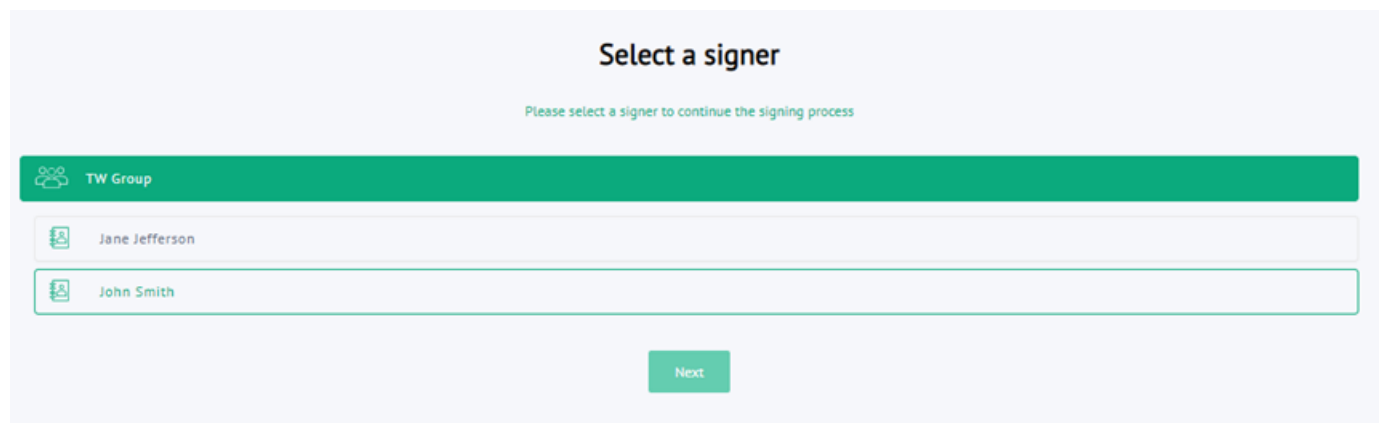
### Signing step-by-step

- [Access the documents](#) that require your signature.

*Select the signer*

**Note:** this step only applies in two situations:

- When multiple signers need to sign the document, and you clicked the **Sign all** button in the Document Portal.
- When a contact group needs to sign the document, and you clicked the **Sign all** button in the Document Portal.
- Select the signer you want to sign first.



The screenshot shows a web interface titled "Select a signer". Below the title is a green bar with a group icon and the text "TW Group". Below this bar are two selection options, each with a person icon and a name: "Jane Jefferson" and "John Smith". The "John Smith" option is currently selected, indicated by a green border. At the bottom center is a green button labeled "Next".

- Click **Next**.

*Optional documents*

- If your package contains any optional documents, an overview is provided of the **mandatory** and the **optional** documents.

### Select optional documents

This package contains optional documents. Please review and select the documents you would like to retain.

#### Mandatory documents

In this session you will be processing these documents. You can click on the thumbnails to preview them.



Polis - Nieuwe verzekering (117360)



Productfiche (117361)

#### Optional documents

Please click on the thumbnails to preview the document and select the ones you choose to process.

Select all



Algemene voorwaarden (117362)

Select



IPID fiche (117364)

Select

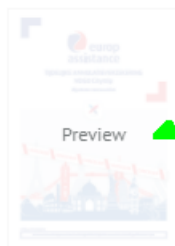
Next

- If you want to sign optional documents, click the **Select** button next to each optional document you want to sign.  
Or to sign all optional documents, click **Select all** beneath **Optional documents**.
- If you don't want to sign any of the optional documents, click **Next** directly.
- **Tip:** Click the thumbnail of an optional document to open it in preview. That way you can read through the document and decide whether you want to sign it. When you're done, click the **Back** button.

### Optional documents

Please click on the thumbnails to preview the document and select the ones you choose to process.

Select all



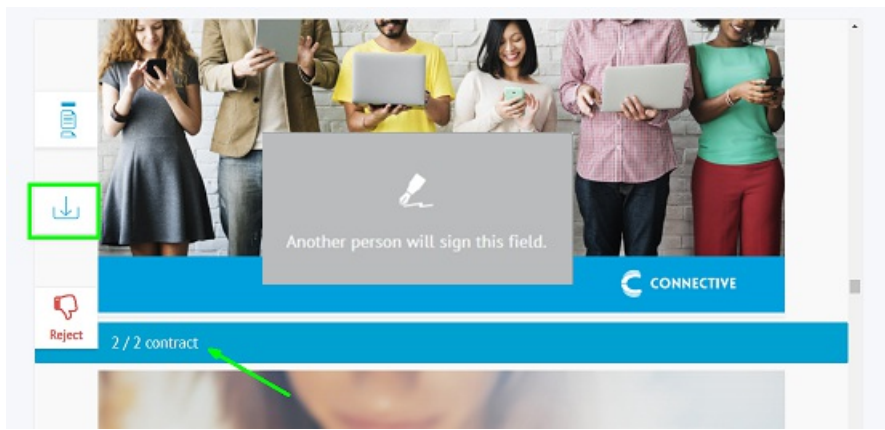
Algemene voorwaarden (117362)

Unselect

Next

- After you've clicked **Next**, read the entire document (if still necessary) 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.



**Tip:** you may now also be able to download the document before signing and print it, to read it on paper for instance. To do so, click the download icon.

**Note:** whether the download icon is available depends on whether the initiator enabled it. Also note that the download icon becomes unavailable as soon as you click **Start signing**.

Also note that if your document contains form fields, only the original values of the fields will be displayed when downloading the files, not the values that have been filled in or modified by the form filler. This is intended behavior in the current version.

Also note that the download icon becomes unavailable as soon as you click **Start signing**.

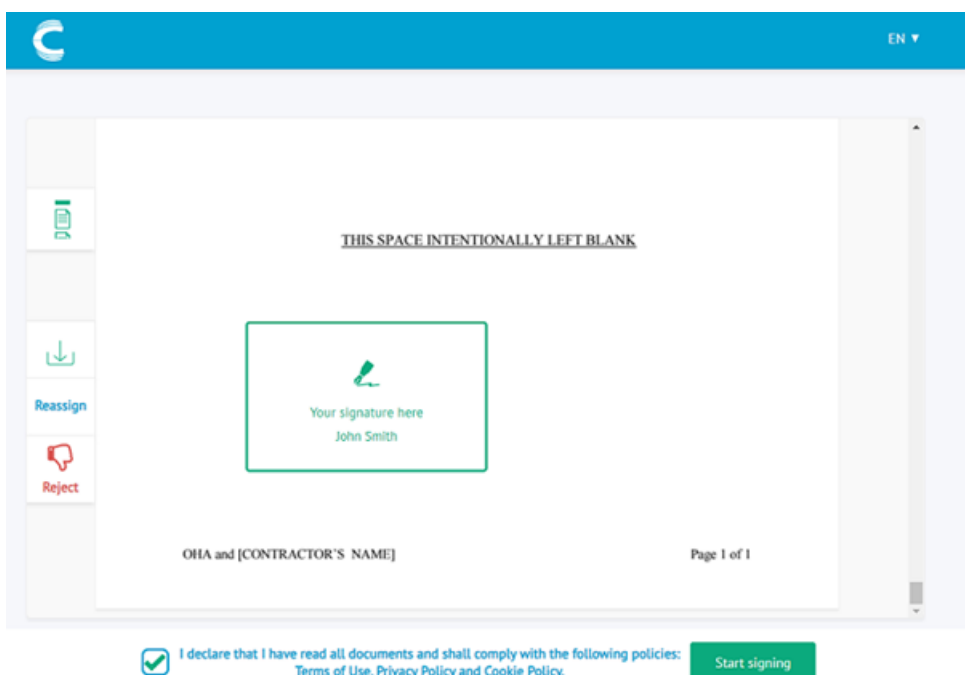
### Signing

- 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**.

**Note:** this checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.

- If you are not the right person to sign the document, click the **Reassign** button and assign it to someone else. **Note:** whether the **Reassign** button is available depends on the configuration by the administrator. See [How do I reassign a document?](#) for more information.

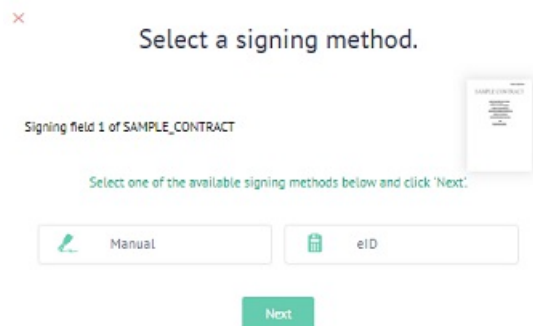


**Tip:** click the links to consult the Terms of Use, Privacy Policy and Cookie Policy respectively.

### 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?](#)

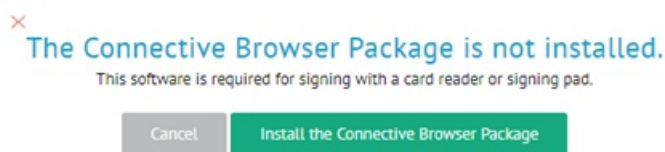


### Install the Connective browser package

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.



After the browser package has been installed, you'll restart at the first step of the signing process.

**Note:** 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.

### Signature policy

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

**Technical note:** a signature policy is a document that contains a set of rules that detail 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 v3 calls. See **Appendix IV** of the **Connective - eSignatures 6.3.x - Configuration Documentation** to learn how to use signature policies. Note that signature policies can only be entered in version 3 of the eSignatures API, not in the latest v4 version or 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.

✕

1 2 3 4

### Signature Policy:

[Accept the signature policy to continue](#)

Signature policy url:	<a href="https://view.publitas.com/49592/459794/odfs/6b54bc746f3d18b310581b17b2adep90b986f4f0.pdf">https://view.publitas.com/49592/459794/odfs/6b54bc746f3d18b310581b17b2adep90b986f4f0.pdf</a>
Commitment Type:	1.2.840.113549.1.9.16.63
User capacity:	on behalf of
Qualification:	Lawyer
Notice:	usenotice

Accept Reject

#### 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**.

✕

1 2 3

### 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 - and all QuickSign conditions are met - 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 assigned to you.



## Sign manually

Signing field 1 of marker



Sign in the field below.

Retry Next

### Conditions for QuickSigning

- When multiple signing methods have been selected for you to choose from, the same signing methods must be offered to the signer on document of the package. It's not possible to QuickSign a package if the signer can choose from one pair of signing methods on one document, and choose from another pair on another document.
- The following signing methods cannot be combined with QuickSigning: biometric and itsme. If one of these signing methods has been selected, 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 the signer is using a PIN pad eID reader they will be prompted to sign each field individually.

- When the package contains multiple legal notices, their content must be identical (per signer) for QuickSigning to work.

### Signing methods

One of the following signing methods may have been assigned to you.

#### Important:

- Your system admin may not have enabled all signing methods listed below in your eSignatures solution.
- Since eSignatures 6.2, the **DisplayName** of the signing methods is configurable, so the name of the signing methods in your environment may not correspond to the ones listed below, but their behavior will be the same.

### 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.



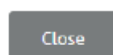
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



### Adding signatures to documents ...



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



### Sign Handwritten

- Type in your full name in the required field.



- Select one of the fonts. Note that these fonts are preconfigured and cannot be modified.

×

1 2

## Sign handwritten

Signing field 1



Sign with your full name in the field below and choose a handwritten font.

John Smith

☒

☐

☐

☐

☐

Cancel

Next

- 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.

### 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** (for Windows and macOS) been properly installed. Otherwise eID signing will not work.

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.

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** (for Windows and macOS) been properly installed. Otherwise manual + eID signing will not work.

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.

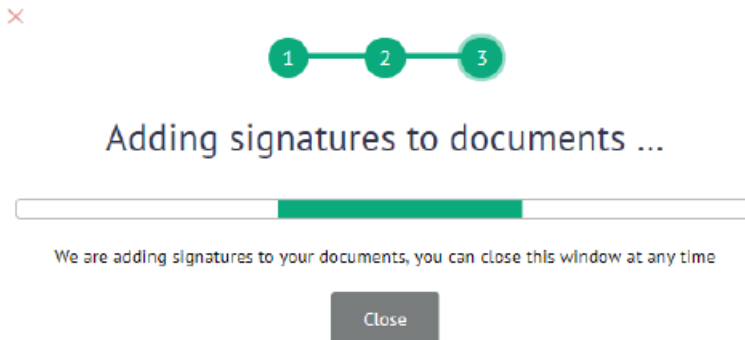
- Enter the code you received and click **Next**.

**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.

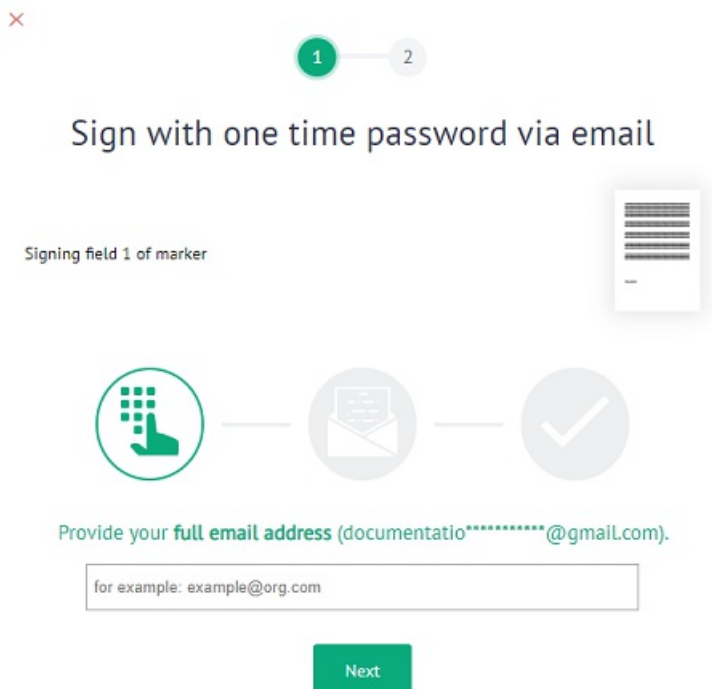
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



**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.



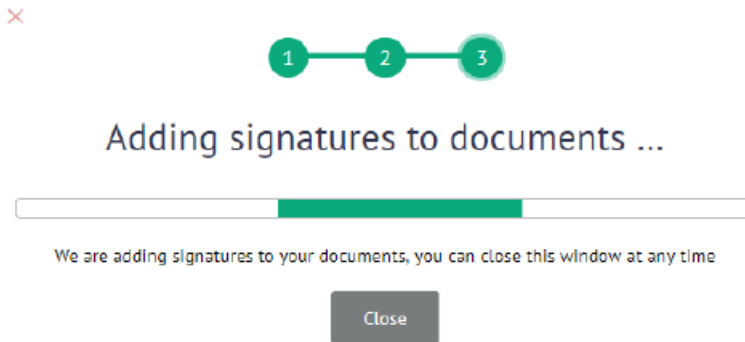
- Click **Next**. If you entered the correct email address, an email is sent to the email address you entered.
- Enter the code you received in the confirmation field.

**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.



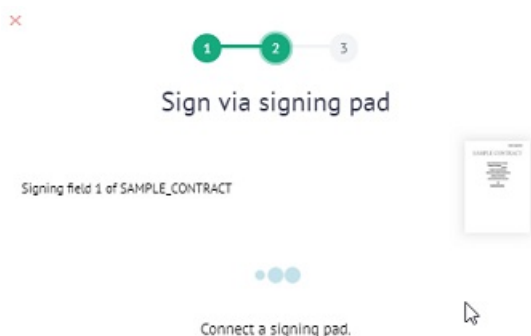
### ***Sign via signing pad (biometric)***

- 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, which means QuickSigning is not supported.



- 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.



## Adding signatures to documents ...



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

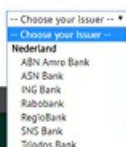
Close

### 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.



### Quicksign via iDIN



### Sign with BeLawyer card

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

Make sure the **Connective Browser Package** (for Windows and macOS) been properly installed. Otherwise BeLawyer signing will not work.

- 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.
- Itsme cannot be combined with QuickSigning: each document within a package must be signed individually.

#### 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. *Note that the certificate creation steps don't apply when using itsme through OpenID Connect.*
- Click **Create my certificate**. A message is now sent to your itsme app.



en ▼

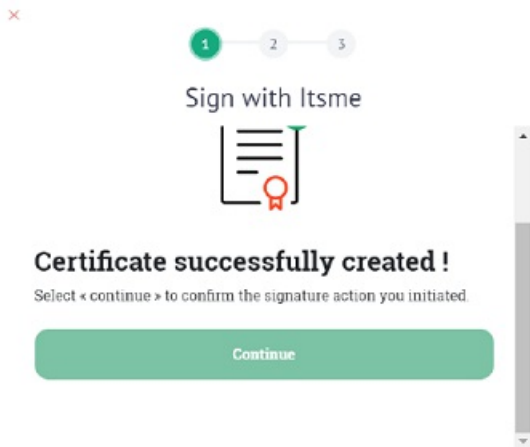
### Create your certificate

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

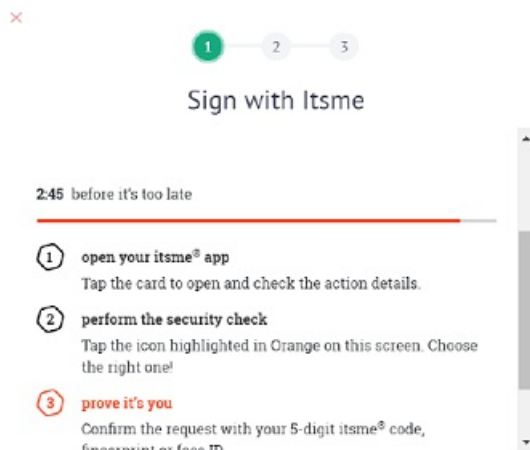
By clicking on « Create my certificate », you agree to Quo Vadis [terms and conditions](#).

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.



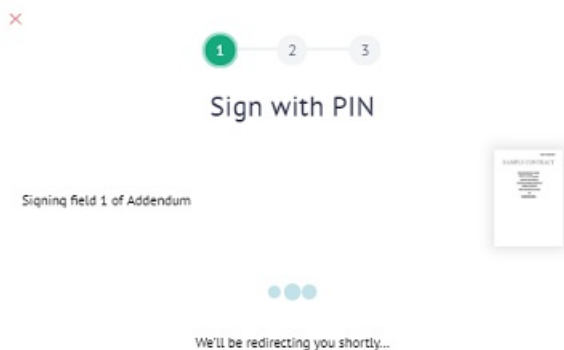
- Your document will now be signed with itsme.

### ***Sign with custom signing type***

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 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 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.

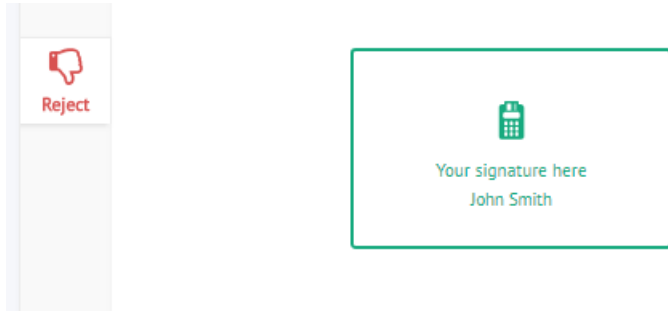
See [Signing in the Document Portal](#) for more info.

#### 2.5.4 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.

✕ **Reject to sign the package.**

Reason for rejection:

|

\* The reject reason is required

Close Reject

- 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.

DOCUMENT PORTAL

SIGNER PORTAL

John Smith

EN ▼

All

Tabs

<input checked="" type="checkbox"/>	Name	Status	Added on ▼	Actions
<input type="checkbox"/>	<div><div></div><div>▶</div><div><div></div>Addendum</div></div>	Pending	26/07/2018	<div><div></div>Sign</div>
<input checked="" type="checkbox"/>	<div><div>▼</div><div><div></div>SAMPLE_CONTRACT</div></div>	Rejected	26/07/2018	
	Initiator		Rejected on	26/07/2018 12:45
			Reason for rejection	Amount is not correct

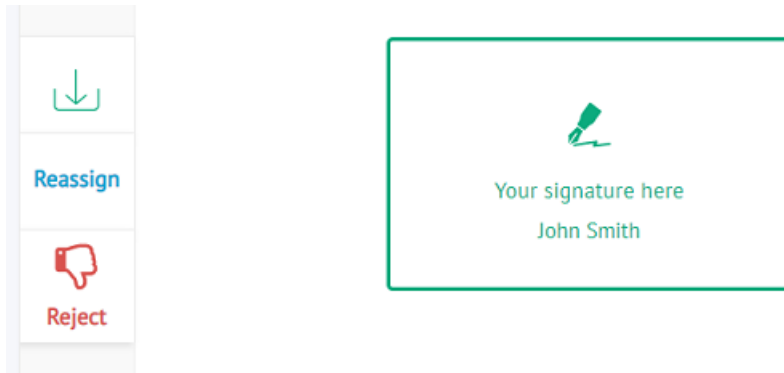
### 2.5.5 How do I reassign a document?

If you feel you are not the right person to approve or sign a document, you can reassign it to someone else.

**Attention:** whether you are able to reassign documents depends on the configuration of your eSignatures solution. If the administrator has not enabled this feature, you won't be able to reassign.

#### To reassign a document:

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



The fields you are required to enter vary, depending on whether you're reassigning the document to another approver or to another signer, and based on the signing methods the previous signer was able to choose from. The fields marked with an asterisk in the interface are mandatory.

×

### Reassign Package

\*

▼

\*

\*

Contact fields

\*

\*

Close

Reassign

- Email

**Note:** if the email address corresponds to a known user in eSignatures, not only do they receive an email notification to approve/sign their document, but the document will also be available for approval/signing in their Signer Portal.

- Language

The language is by default set to the preferred language of the previous approver/signer. Select a different preferred

language if necessary.

The emails the approver/signer receives will be drafted in the language you select here.

- First Name
- Last Name

**Important:** when mandated signing rules have been applied to iDIN signing, the last name must be entered following a specific format. See **Naming conventions for mandated signing with iDIN**. When mandated signing rules have been applied, the authenticity of the signer will be checked. So, make sure the data entered for the contact are identical to that on the signing certificate of their Belgian eID card, BeLawyer card, iDin or itsme account. For instance, 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, iDIN or itsme account. **Note:** iDIN does not store the first name of its users.

- Birthdate
- Phone number

The phone number is required if you want the signer to be able to use SMS signing. The country code prefix is by default set to the one configured by the administrator in the Configuration Index. If the administrator did not select a country code prefix, BE (+32) will be selected as default.

- Contact fields

**Attention:** additional contact fields for eID and BeLawyer signing are only displayed if the signing type(s) of the previous signer were set to BeID, manual+BeID and/or BeLawyer and mandated signing rules are applied.

- Enter the new signer's national security number for eID signing.
- Enter the new signer's lawyerID for BeLawyer signing.

- Reason for reassignment
- When you're done, click **Reassign**.

All parties involved receive an email notification informing them about the change:

- The initiator who sent the document is notified that one or more approvers/signers have changed.
- The new approvers/signers are notified a document requires their attention

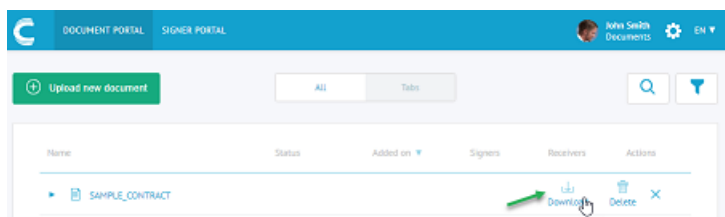
## 2.5.6 How do I download a signed document?

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.

**Note:** Any unsupported characters your package or document may contain are escaped during download. Therefore, they might have a slightly different file name than the one chosen by the initiator.

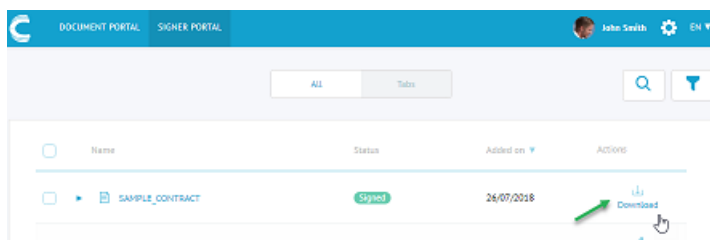


- Click the document to open it.

### 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.

**Note:** Any unsupported characters your package or document may contain are escaped during download. Therefore, they might have a slightly different file name than the one chosen by the initiator.



## 2.5.7 How does mandated signer validation work? (eSignatures 6.2 and higher)

---

**Important:** this section describes how mandated signer validation works as of eSignatures 6.2.

If you're using an eSignatures version prior to 6.2, or API v3 combined with eSignatures 6.2 or higher, see the deprecated [How does mandated signer validation work?](#) topic for more information.

---

### Introduction

Mandated signing is an extra check eSignatures can do to verify if a signer is mandated to sign during a particular signing session.

Now how does eSignatures check whether a signer is mandated to sign? eSignatures compares the data that are retrieved from the signing certificate (in case of BeID, BeLawyer and Itsme signing) or from the signing service (in case of iDIN) to the signer's data stored in the contact (when using the WebPortal) or passed in a Create Stakeholder call (when using the API).

The data stored in the signing certificate or returned by the signing service is directly tied to the signer. They may contain their first and last name, but also their national security number or lawyerID.

If the data matches based on the rules you've configured, the signer is mandated to sign. If it doesn't, the signer receives a message they are not mandated.

### Mandated signing validation rules

In previous versions of eSignatures, you only had two types of mandated signing: based on **nameandbirthdate** or based on **matchid**.

When using **nameandbirthdate**, the first name, last name and birthdate had to be known to eSignatures and had to match the external data.

When using **matchid**, the national security number (in case of eID signing) or lawyerID (in case of BeLawyer signing) had to be known and match the external data.

As of eSignatures 6.2, you have a series of preconfigured mandated signing rules that are each tied to one specific contact field, and that you can apply to the SigningMethods of your choice. Also new is that you can apply multiple signing rules to a single SigningMethod. For BeID signing for instance, you could choose to do a check on the national security number, first name, last name and birth date. In that case you would be combining the two 'old' mandated signing rules based on **matchid** and **nameandbirthdate** for instance, which was not supported in previous versions. Or for iDIN signing, you can choose to apply a signing rule that only checks the last name and birthdate, and not the first name, since iDIN does not store first name info. You can also create custom mandated signing rules and configure a custom expressions that must be used to extract the data.

Note however that not every SigningMethod supports mandated signing. Only the SigningMethods that use a signing certificate or signing service are supported. This currently comes down to:

- BeID
- BeLawyer
- Itsme
- iDIN
- OpenID Connect

### How to use mandated signer validation?

1. [Configure mandated signing rules](#)
2. [Mandated signing in the WebPortal](#)
3. [Mandated signing in API v4](#)
4. [Which data must match?](#)



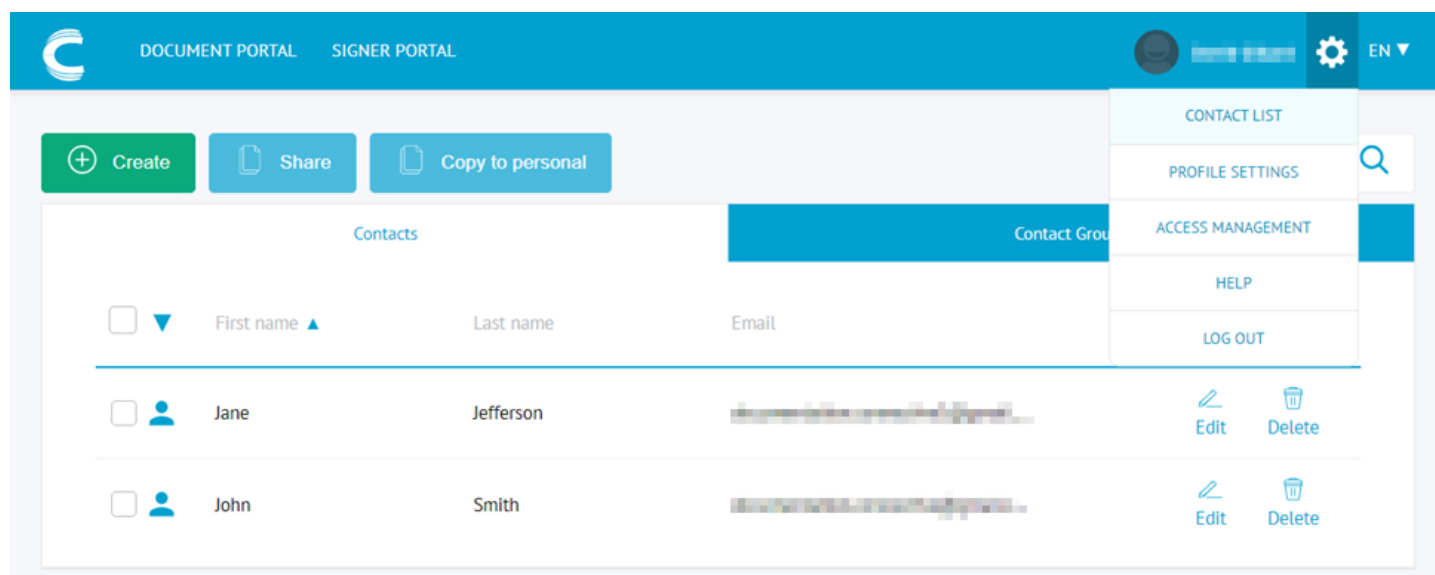
## 2.6 Managing contacts and contact groups

In the **Contact list** section you can manage the **contacts** and **contact groups** in eSignatures.

A *contact* in eSignatures is a person who can **approve** and or **sign** documents and **receive** signed documents. To use a contact as signer, you need to create a signing field on the document, and add the contact to it.

Contacts can be grouped into *contact groups*, which in turn can be added to a signing field, just like a singular contact. The difference is that any contact who belongs to the contact group is able to sign the signing field. It doesn't matter which contact signs it, as long as it's a member of the group. Note however that as soon as one member has signed, the others no longer can't. Once a document has been signed by a member of a contact group, all members will be notified.


Contacts and contact groups can now also be shared with all users on the eSignatures environment. This way, it's no longer necessary to create the same list of contacts for each user, as was the case in previous versions.



### Contacts vs Users

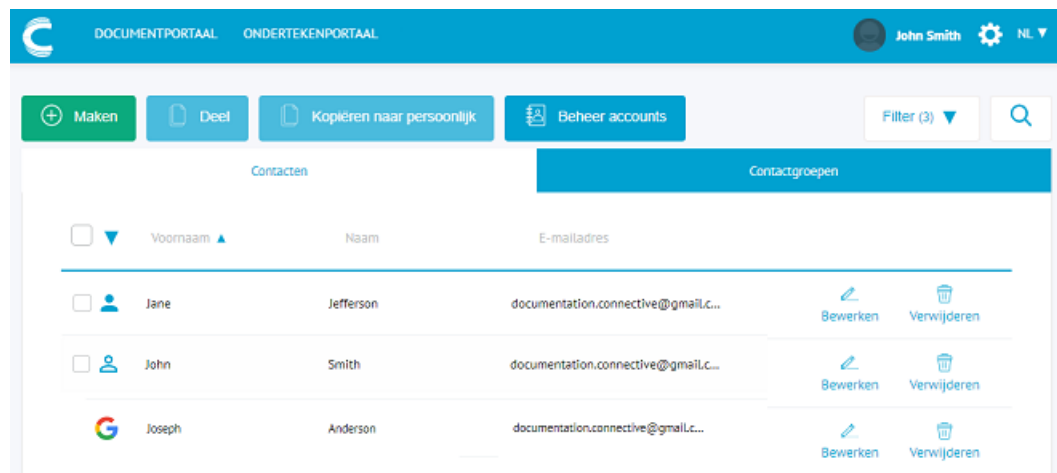
Don't confuse *contacts* with *users*. These are two different roles within eSignatures. A *contact* is a person who can sign and receive document, explained earlier. A *user* is a person who has access to the eSignatures WebPortal.

### To access the Contact list section:

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

## 2.6.1 Contacts

On the **Contacts** tab you have an overview of all your contacts. These may be the contacts you've created yourself, you've linked from a Cloud account or that were shared with you by other users (depending on the system configuration).



### Personal contacts

Personal contacts are the contacts you've created yourself in eSignatures.

They are marked by a fully colored icon.

You are the only user who can access them.

### Shared contacts

Shared contacts are the contacts you've shared with other users, or that other users have shared with you.

They are marked by a transparent icon.

Shared contacts can be used by all eSignatures users with the necessary permissions.

### Cloud contacts

Cloud contacts are the contacts you've linked from your Google and/or Office 365 account.

They are marked by the Google icon or Office 365 icon.

You are the only user who can access them.

### Actions


On the **Contacts** tab you can do the actions listed below. Whether you're able to do all actions depends on the user permissions the administrator has assigned to you.

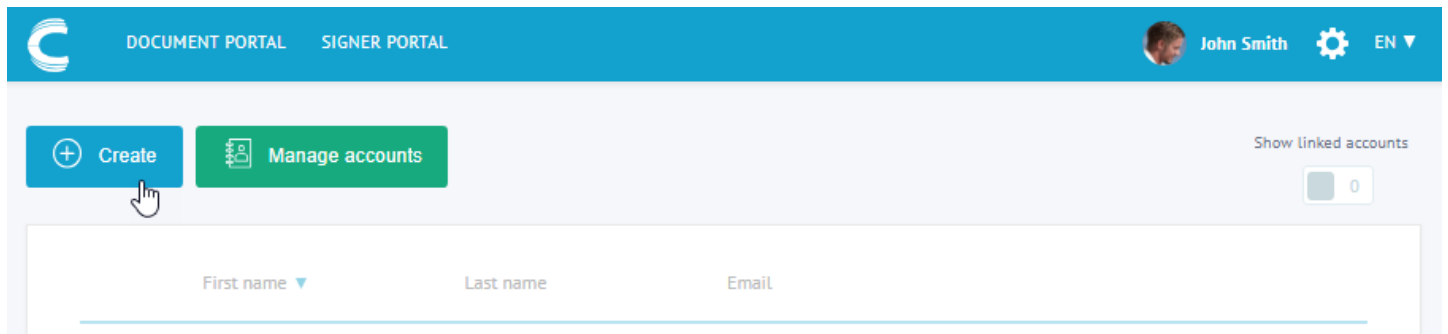
- [Create contacts](#)
- [Share contacts](#)
- [Copy contacts](#)
- [Manage contacts from Cloud accounts](#)
- [Filter and search for contacts](#)
- [Edit contacts](#)
- [Delete contacts](#)

- [Merge contacts](#)

**Note:** there is no "Merge" button. You are only prompted to merge contacts when creating (or editing) a contact whose email address already exists in eSignatures.

## How do I create 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\*

**Important:** the email address must be unique. When creating a new personal contact and another personal contact with the same email address already exists, you'll be prompted to merge it. The same goes for shared contacts. However, if you're creating a *personal* contact whose email address is already used in a *shared* contact, or vice versa, you won't be prompted to merge.

- Personal title
- First Name\*
- Last Name

**Important:** when mandated signing rules have been applied to iDIN signing, the last name must be entered following a specific format. See Naming conventions for mandated signing with iDIN. When mandated signing rules have been applied, the authenticity of the signer will be checked. So make sure the data entered for the contact are identical to that on the signing certificate of their Belgian eID card, BeLawyer card, iDin or itsme account. For instance, 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, iDIN or itsme account. **Note:** iDIN does not store the first name of its users.

- Birthdate
- Phone number The phone number is required if you want the contact to be able to use SMS signing. The country code prefix is by default set to the one configured by the administrator in the Configuration Index. If the administrator did not select a country code prefix, BE (+32) will be selected as default.
- Language

The language is by default set to the preferred language of the user who is creating the contact (i.e. the language that is configured in the user's Profile settings.) Select a different preferred language if necessary. The emails the contact receives will be drafted in the language you select here.

Create new contact

Email <span style="float: right;">*</span> <small>Enter a valid email address.</small>		Personal title <small>Enter a personal title.</small>	
First Name <span style="float: right;">*</span> <small>Enter first name here.</small>		Last Name <span style="float: right;">*</span> <small>Enter last name here.</small>	
DD <small>Day</small>	- <small>Month</small>	YYYY <small>Year</small>	<div style="display: flex; align-items: center;"> <span>BE (+32) ▾</span> </div> <small>Enter a valid phone number to receive a one time SMS password.</small>
English ▾ <small>Choose a preferred language.</small>			
<div style="border: 1px solid #ccc; padding: 10px; min-height: 100px;"> <p style="color: #008000; margin: 0;">Additional Contact Fields</p> <p style="text-align: center; color: #ccc; margin: 10px 0;">No data to display</p> <div style="background-color: #008000; color: white; text-align: center; padding: 5px; margin: 0 auto; width: 100px;">Add contact field</div> </div>			
Make this a shared contact <div style="display: flex; align-items: center;"> <input checked="" type="checkbox"/> </div>			

Cancel
Confirm

- To share this contact with other eSignatures users, select **Share contact**.

All other users with the necessary permissions on your eSignatures environment will be able to view and use this contact.

**Note:** whether this option is available also depends on your permissions.

- When you're done, click **Confirm** to create the contact. The new contact is now added to the contacts list.

#### Contact fields

**Important:** Contact fields must *only* be used if you want the signer to use mandated signing based on the national security number for BeID signing or based on the lawyerID for BeLawyer signing.

#### To create an additional contact field:

- Click **Add contact field**.
- Select the **contact field type**: eID or BeLawyer.
- Enter the contact's national security number or lawyerID, depending on the selected type.
- Click **Confirm** to save the contact's settings.

### Create new contact

Email \*  
Enter a valid email address.

Personal title  
Enter a personal title.


First Name \*  
Enter first name here.

Last Name \*  
Enter last name here.

DD  
Day

-  
Month

YYYY  
Year

 BE (+32) ▼  
Enter a valid phone number to receive a one time SMS password.

English ▼  
Choose a preferred language.

Additional Contact Fields

-- Select contact field type -- ▼

-- Select contact field type --

eID

BeLawyer

✕

Make this a shared contact

☒ 0

Cancel

Confirm

#### Supported countries for SMS OTP signing

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**.

When using mandated signing based on **name and birth date** with iDIN, and the system admin set the **MatchLevel** to 100 (%), the name of the contact must meet the following format:

- It must be the legal last name of the iDIN consumer, without prefixes.
- It may contain a maximum number of 200 characters without numbers.
- It must not include the prefix of the *first* last name. Prefixes of later sections of the last name must be included.
- If the last name consists of multiple sections that are separated by a -, the – must not be lead and followed by a space.
- Leading and trailing spaces are not allowed.

#### *Examples*

- For the name "Luana van Oranje-Nassau van Amsberg" the value should be "Oranje-Nassau van Amsberg".
- For the name "Jacques d'Ancona" the value should be "d'Ancona".
- For the name "Jacques d' Ancona" the value should be "Ancona".

**Tip:** in case problems should occur with special names and mandated signing with iDIN, contact the eSignatures system admin to slightly reduce the **MatchLevel**.

## How do I share contacts?

Since eSignatures 5.4, you can choose to share contacts with all other users who have access to the eSignatures WebPortal. This way, each user no longer needs to create contacts manually.

All other WebPortal users who have the necessary permissions, will be able to view and use the contacts you shared.

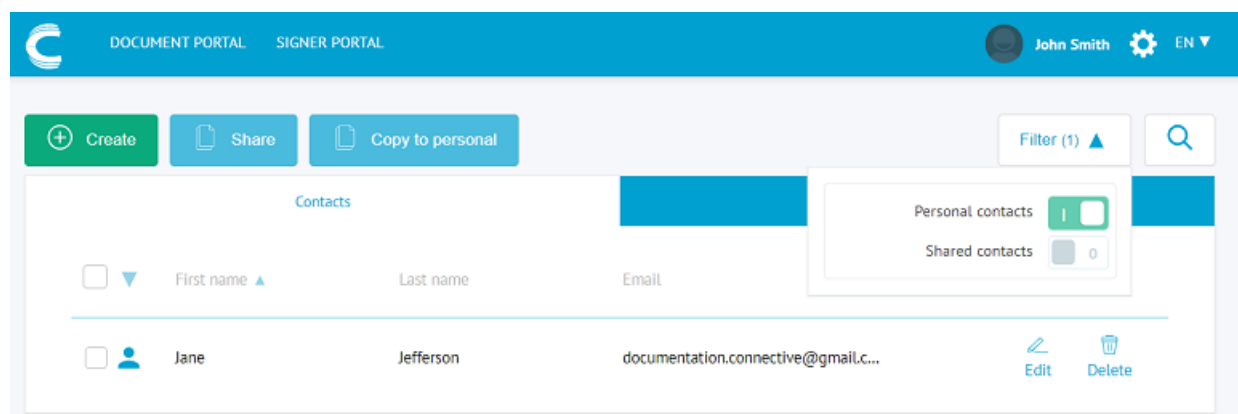
**Attention:** whether you're able to share contacts, depends on whether the administrator provided you with the necessary sharing permissions.

### To share contacts:

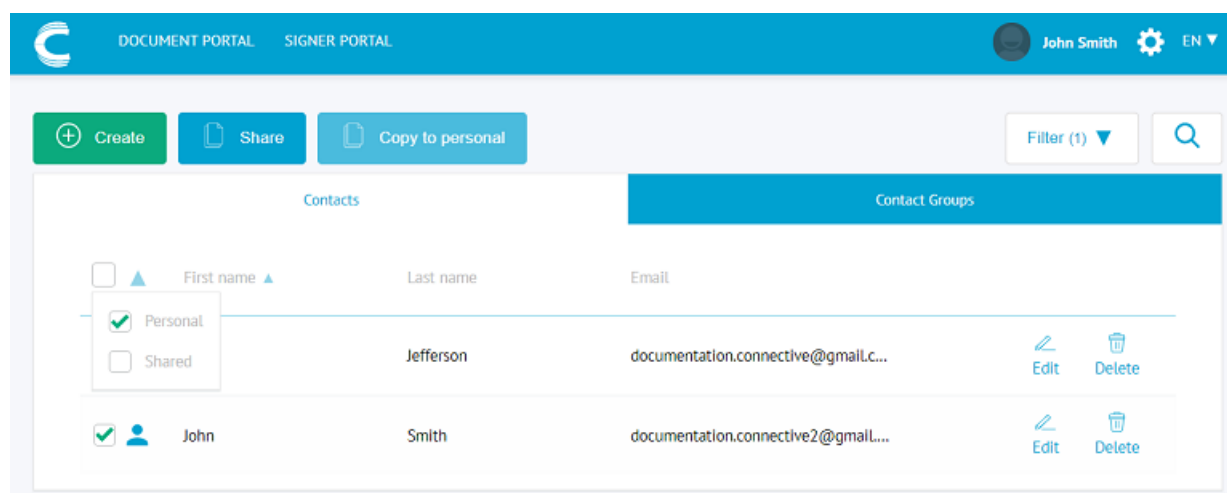
- Click the settings icon in the main menu.
- Click **Contact list**.
- Select the personal contacts you want to share.

#### Tip:

To view only only your personal contacts, click the down arrow next to **Filter**, and only select **Personal contacts**. Make sure **Shared contacts** and **Cloud contacts** are unselected.



- To select all personal contacts, click the checkbox in front of **First Name**, and then select **Personal**.



- Click **Share**.
- If a shared contact with the same email address should already exist, you are prompted whether to merge the contact. See [How do I merge contacts?](#) for more information.



## How do I copy shared contacts to personal contacts?

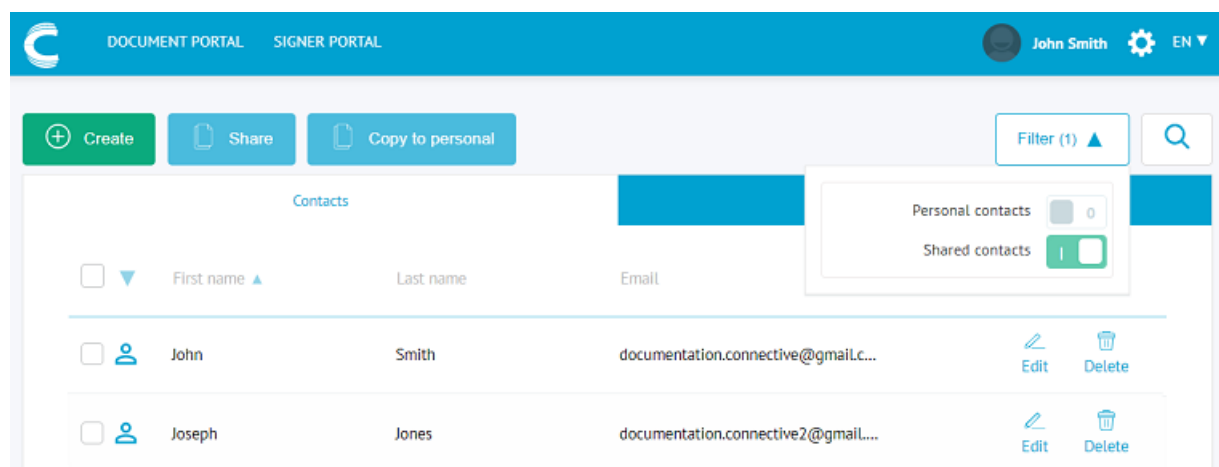
You can copy the contacts that have been shared with you to your personal contacts.

**Attention:** whether you're able to copy shared contacts, depends on whether the administrator provided you with the necessary sharing permissions.

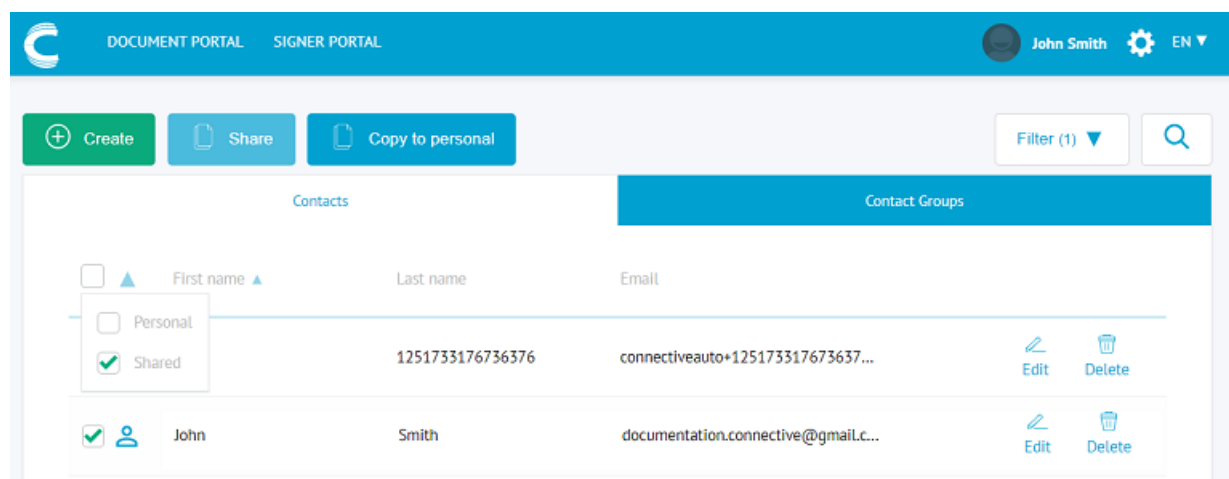
### To copy shared contacts to your personal contacts:

- Click the settings icon in the main menu.
- Click **Contact list**.
- Select the shared contacts you want to copy.

**Tip:** To view only your shared contacts, click the down arrow next to **Filter**, and only select **Shared contacts**. Make sure **Personal contacts** and **Cloud contacts** are unselected.



- To select all shared contacts, click the checkbox in front of **First Name**, and then select **Shared**.



- Click **Copy to personal**.
- If a personal contact with the same email address should already exist, you are prompted whether to merge the contact. See [How do I merge contacts?](#) for more information.

## How do I manage Cloud contacts?

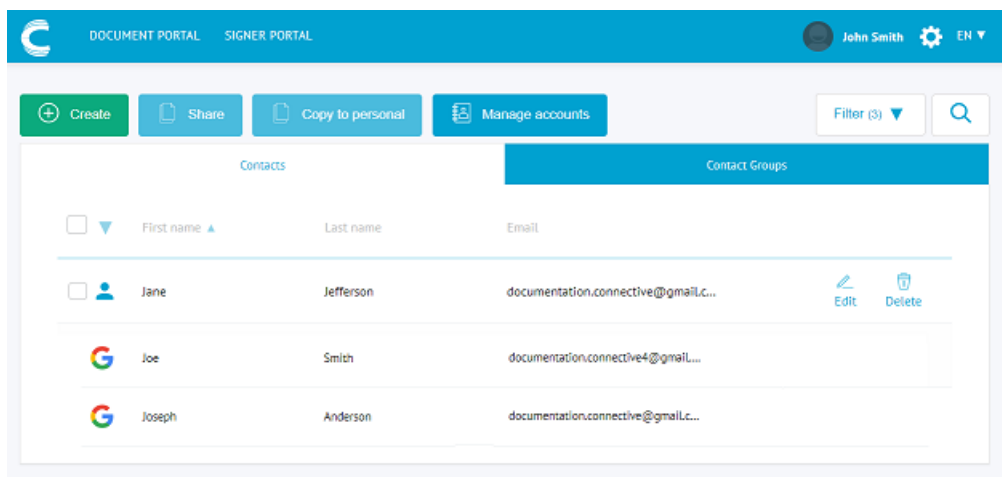
Besides creating contacts manually, you can also link the contacts you have in Google Contacts and Office 365 to eSignatures. When these accounts have been linked, your Cloud contacts are displayed in the **Contact List** section, under **Contacts**, and you can use them as signers and receivers.

**Important:** for this feature to work, the **Cloud settings** must be enabled and properly configured in the Configuration Index. To learn how to do so, see **Connective - eSignatures 6.3.x - Configuration Documentation**.

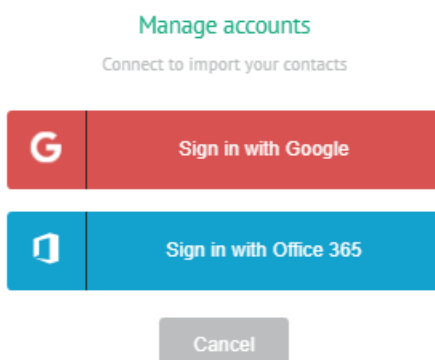
### To manage Cloud contacts:

- 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 enabled in the Config Index. Contact your administrator to enable them.



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



### Sign 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.

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

When you no longer want to use Cloud contacts, you simply unlink the account you no longer wish to use.

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

or

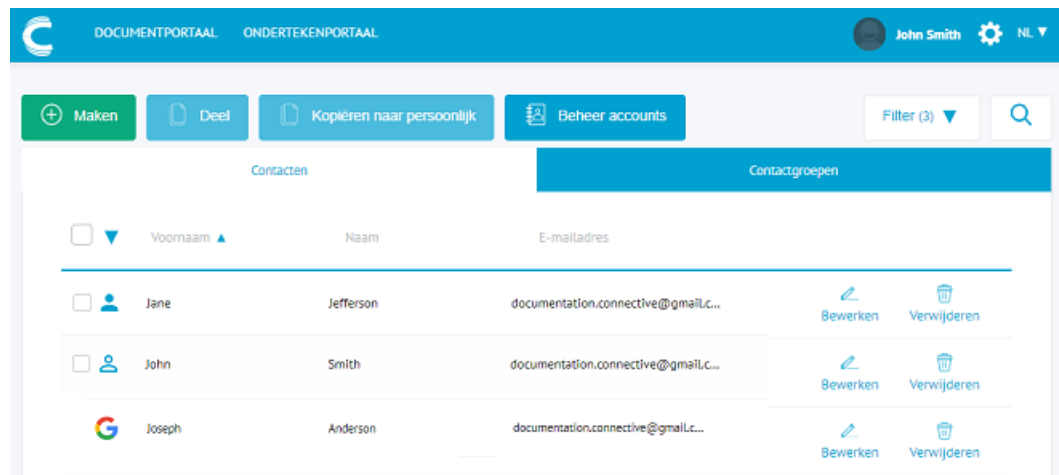
- Click **Unlink from Office 365**.

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

## How do I filter and search for contacts?

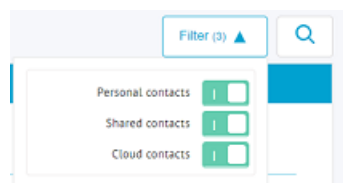
By default, all types of contacts are displayed on the **Contacts** tab:

- Personal contacts
- Shared contacts
- Cloud contacts



### To filter the contact types:


- Click the down arrow next to **Filter**.
- Select the contact types you want to display.



### To search for contacts:

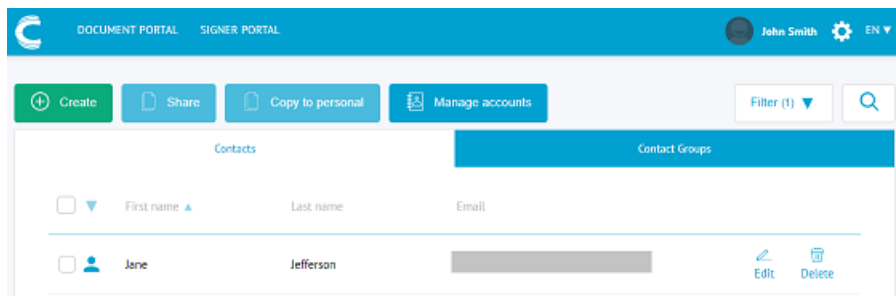
- Click the search icon.
- Enter the name of the contact you want to search.
- Click **Search**. Note that only the contact types you've selected in the filter will be searched.

## How do I edit contacts?

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

**Tip:** if the **Edit** button is unavailable, this means the contact is a shared contact and you don't have the necessary permissions to edit it. You can still view the contact's details by clicking the **View** button.

To obtain editing permissions of shared contacts, contact your administrator.



- Edit the contact's information. Note that in the Edit screen, the **Share contact** option is not available. To share a contact, you need to click the **Share** button.


**Attention:** Be careful when changing a contact's email address. Documents that have already been sent are not updated retroactively. So, they won't be sent automatically to the updated email address. Also, any reminders that are sent about existing documents will still arrive at the previous email address.

**Edit this contact**

<input type="text" value="Enter a valid email address."/>	<input type="text" value="Personal title"/>
<input type="text" value="John"/>	<input type="text" value="Smith"/>
<input type="text" value="Enter first name here."/>	<input type="text" value="Enter last name here."/>
<input type="text" value="Day"/> Day	<input type="text" value="Month"/> Month
<input type="text" value="1999"/> Year	<input type="text" value="BE (+32)"/> Enter a valid phone number to receive a one time SMS password.
<input type="text" value="English"/> Choose a preferred language.	
<b>Additional Contact Fields</b> No data to display <input type="button" value="Add contact field"/>	

- Click **Confirm** to save the changes.

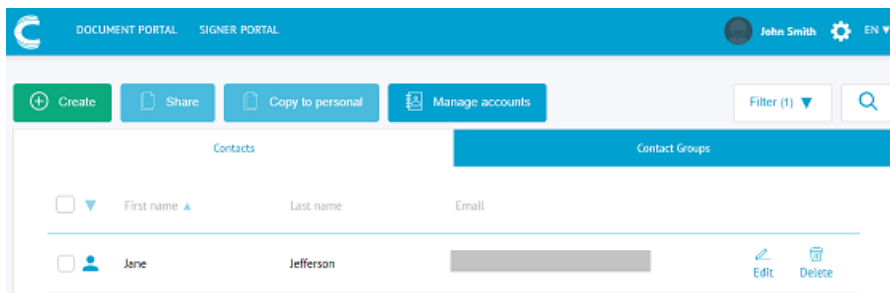
## How do I delete contacts?

- Click the settings icon  in the top toolbar.
- Click **Contact List**.
- Click the **Delete** button next to the contact you want to delete. Note that it's currently not supported to select multiple contacts and delete them simultaneously.

**Attention:** when you delete a shared contact, it will no longer be available to the other users of your eSignatures environment.

**Tip:** if the **Delete** button is unavailable, this means you don't have the necessary permissions to delete the shared contact. Contact your system administrator.

- Click **Confirm** to delete.



## How do I merge contacts?

Since the email address of a contact must be unique in eSignatures, you are prompted to merge contacts whenever you're creating, editing, copying or sharing a contact whose email address already exists.





The only situation in which an email address may occur twice in eSignatures, is when you have a personal contact that uses a certain email address, and a shared contact that uses the same email address. So, if you're creating/editing/sharing/copying a *personal* contact whose email address is already used in a *shared* contact, or vice versa, you won't be prompted to merge.

### To merge contacts:

- When copying or sharing multiple contacts, you are asked which contacts to merge.

Select contact to merge

These shared contacts already exists. Please select the contacts you would like to merge.

First name	Last name	Email	
Jane	Jefferson	documentation.connective@ig...	 Merge  Skip
John	Smith	documentation.connective2@...	 Merge  Skip

[Skip all](#)






- If you don't want to merge any contacts, click **Skip all**.
- If you want to skip a specific contact, click the **Skip** button next to it.

Contacts you skip will not be merged, and remain shared or personal contacts, depending on their initial state.

- Select the contact you want to merge and click **Merge**.
- Any info that doesn't match between the contacts is displayed.

Merge contact

This shared contact already exists. Please review and select the contact details you would like to merge

Your contact		Shared contact
documentation.connective2@gmail.com		documentation.connective2@gmail.com
Jane		Janine
Enter first name here.		Enter first name here.
19 May 1982		DD - YYYY
Day Month Year		Day Month Year
 BE (+32)		 BE (+32) 472236766
Enter a valid phone number to receive a one time SMS password.		Enter a valid phone number to receive a one time SMS password.
<a href="#">Cancel</a>		<a href="#">Confirm</a>

- Select the info you want to add to merge and click the corresponding arrow. Note that there is no **Undo** button. Once you've clicked an arrow, the only way to edit the merged content is manually retype it or click **Cancel** and restart the merge process. Also note that the left column is not editable.
- If you don't want to merge data fields, and keep the original data, click **Confirm** without doing any modifications.

## Deleting contacts

This step only applies when sharing or copying one or more contacts: When your contacts have been successfully merged, you're prompted whether to delete the original copy of the contact.

Click the option of your choice.

Bent u zeker dat u dit contact wilt **verwijderen**?

**Naam** Jane Jefferson

**E-mailadres** 

Annuleren

Bevestigen

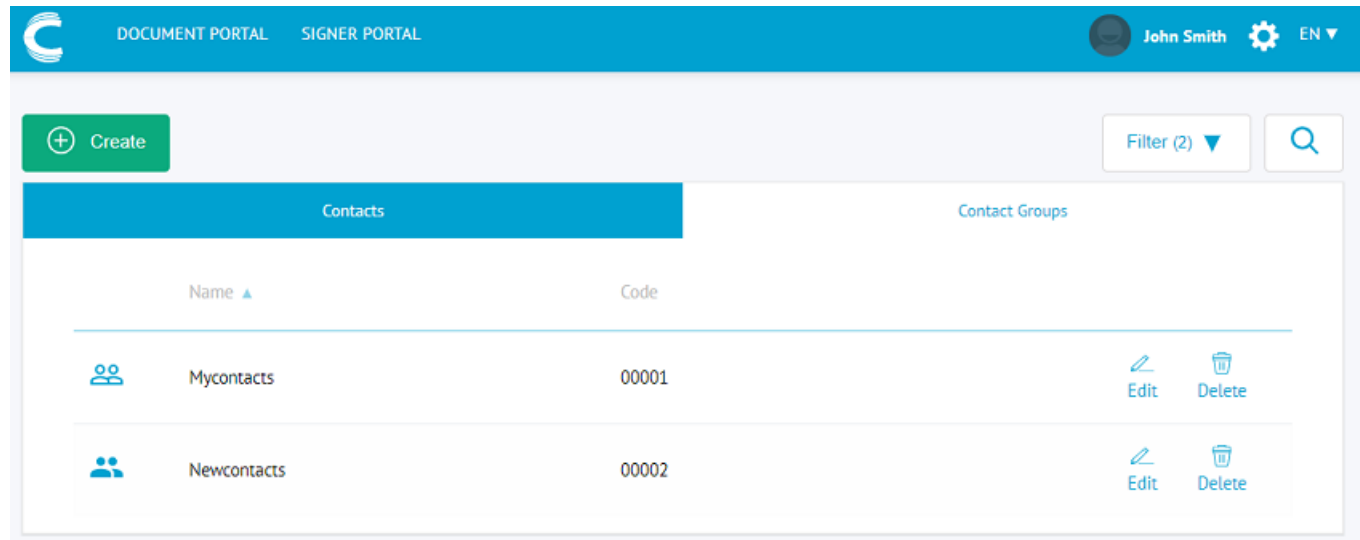




### 2.6.3 Contact groups


On the **Contact groups** tab, you have an overview of all your contact groups. These may be the contact groups you've created yourself or that were shared with you by other users (depending on the system configuration.)

A contact group can be added as approver, signer and receiver in the same manner as a single contact. The difference is, any member of the contact group can approve/sign for the entire group. Note however, that once one member has approved/signed, the others no longer can't.



#### Personal contact groups

Personal contact groups are the ones you've created yourself in eSignatures.

They are marked by a fully colored icon. 

You are the only user who can access them.

#### Shared contact groups

Shared contact groups are the ones you've shared with other users, and that other users have shared with you.

They are marked by a transparent icon. 

Shared contact groups can be used by all eSignatures users with the necessary permissions.

#### Actions

- [Create contact groups](#)
- [Share contact groups](#)

Note there is no "Share" button. If you want to share a contact group, you must select the **Share contact group** option when creating the contact group. It's currently not supported to first create a contact group, and share it afterwards like you can do with single contacts.


- [Filter and search for contact groups](#)
- [Edit contact groups](#)
- [Delete contact groups](#)

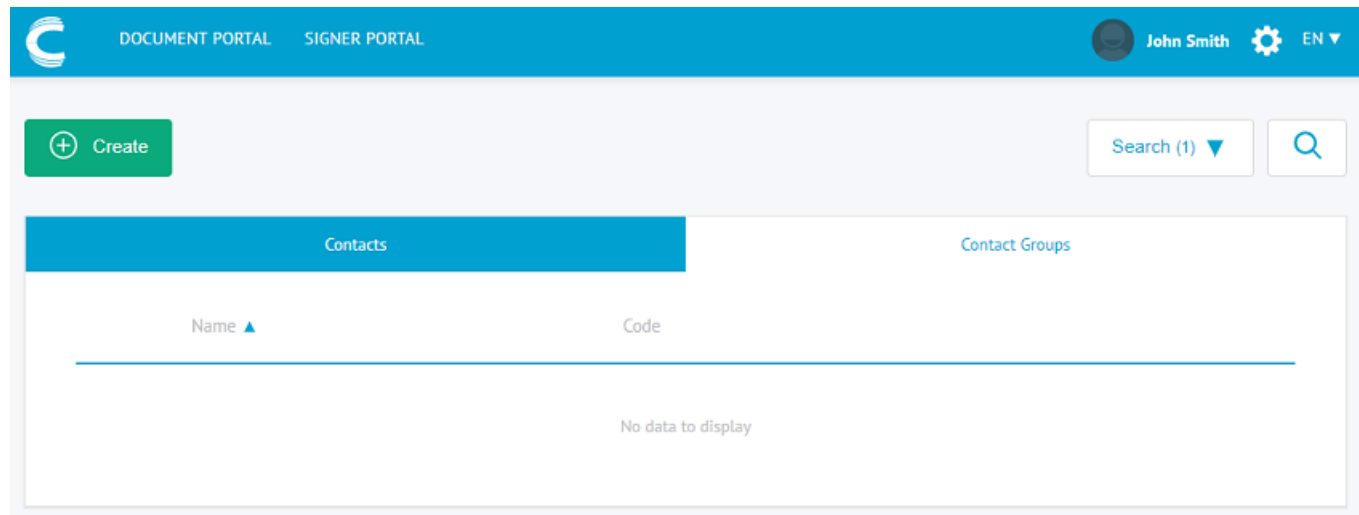
## How do I create contact groups?

Besides creating contacts, you can now also create contact groups. Any contact that is a member of the group will be allowed to sign for the entire group.

**Attention:** as soon as one member of the contact group has signed, the others no longer can't.

### To create a contact group:

- Click the settings icon  in the top toolbar.
- Click **Contact List**.
- Click the **Contact Groups** tab.
- Click **Create**.



- Enter a name for the contact group.

**Important:** the name must be unique.

- If you want to share this contact group with other eSignatures users, select **Share contact group**. All other users with the necessary permissions on your eSignatures environment will be able to view and use this contact group.

**Important:** It's currently not supported to first create the contact group, and then share it afterwards. The decision whether to share the contact group must be made during the creation of it.

**Note:** whether the share option is available also depends on your permissions.

- Click inside the field **Search name or add email address** to view the available contacts.

**Note:** when you've selected the share option, only shared contacts are displayed.

- Select a contact from the list to add it to the contact group.

**Tip:** if no contacts are available, you can create them by clicking the **Create new contact** button.


## Create contact group

My contact group


Contact group name


Code: XXX


Select a contact to add to the contact group

 Search name or add email address

[+ Create new contact](#)

 Jane Jefferson

 Joe Smith

 John Smith

- The contacts you've selected are now listed under **Included in contact group**.
- Click **Confirm**.

How do I share contact groups?

Sharing a contact group must be done during the creation of it.

See [Create contact groups](#) for more information.

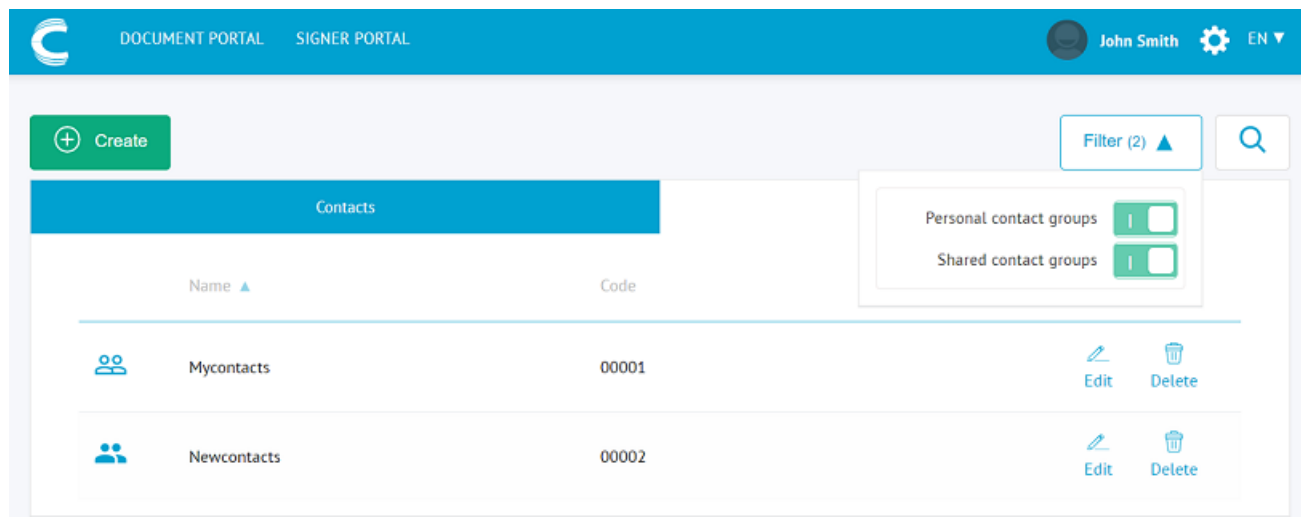
## How do I filter or search for contact groups?

By default, the two types of contact groups are displayed on the **Contacts Groups** tab:

- Personal contact groups
- Shared contact groups

### To filter the contact group types:

- Click the down arrow next to **Filter**.
- Select the contact group types you want to display. **Note:** if the **Shared contact groups** option is not displayed, it means you don't have the permissions to use shared contact groups. Contact your administrator.




The screenshot shows the 'Contacts Groups' interface. At the top, there's a blue header with a logo, 'DOCUMENT PORTAL', 'SIGNER PORTAL', a user profile 'John Smith', and a language dropdown 'EN'. Below the header, there's a green 'Create' button with a plus icon. To the right of the 'Create' button is a 'Filter (2)' button with an upward arrow and a search icon. A dropdown menu is open, showing two options: 'Personal contact groups' and 'Shared contact groups', each with a toggle switch. The 'Personal contact groups' toggle is currently turned on. Below the filter menu, there's a table with two columns: 'Name' and 'Code'. The table lists two contact groups: 'Mycontacts' with code '00001' and 'Newcontacts' with code '00002'. Each row has 'Edit' and 'Delete' icons.

Name	Code
Mycontacts	00001
Newcontacts	00002

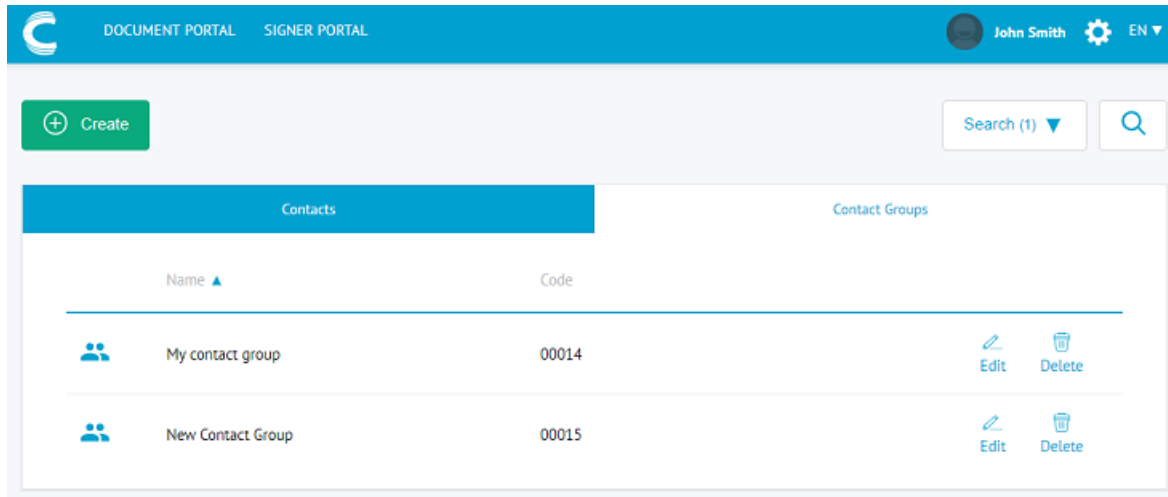
### To search for contact groups:

- Click the search icon.
- Enter the name of the contact you want to search.
- Click **Search**. Note that only the contact group types you've selected in the filter will be searched.

## How do I edit contact groups?

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

**Tip:** if the **Edit** button is unavailable, this means the contact group is a shared group and you don't have the necessary permissions to edit it.



- To add additional contacts to the group, click in the field **Search name or add email address** and select the contacts you want to add. If no contacts are displayed, this means all your contacts have been added to the group. Click **Create new contact** to create new ones. Note that you cannot edit the contacts themselves at this point. This must be done on the **Contact** tab.
- To remove a contact from the contact list, click the **Delete** button next to the required contact. Note that it's currently not supported to select multiple contacts and delete them simultaneously.




**Edit contact group**

My contact group Code: 00014

Contact group name


Select a contact to add to the contact group

Included in contact group

Jane	Jefferson	<input type="text"/>	 Delete
John	Smith	<input type="text"/>	 Delete
Joe	Smith	<input type="text"/>	 Delete

- Click **Confirm** to save the changes.

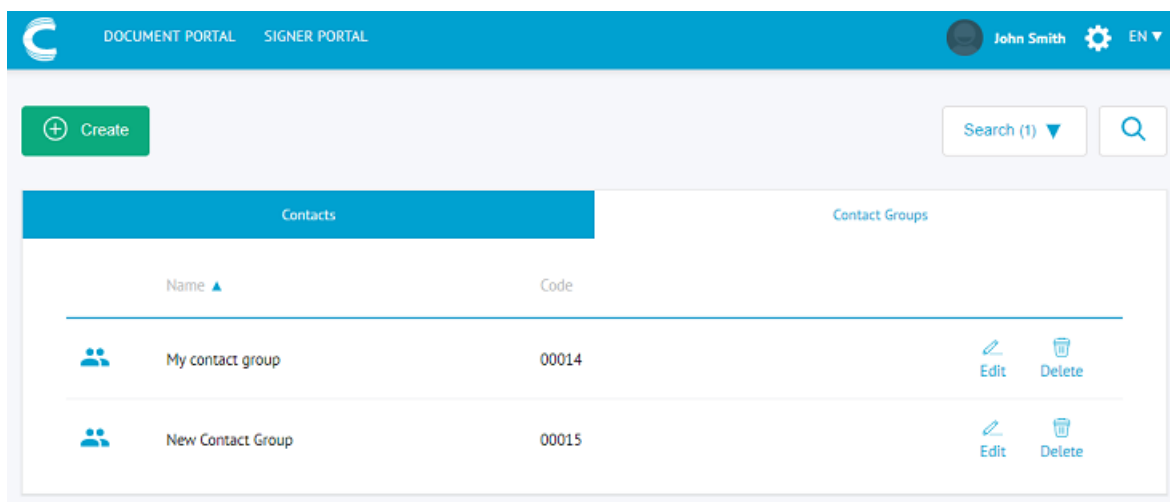
## How do I delete contact groups?

- Click the settings icon  in the top toolbar.
- Click **Contact List**.
- Click the **Contact Groups** tab.
- Click the **Delete** button next to the contact group you want to delete. Note that it's currently not supported to select multiple contact groups and delete them simultaneously.







**Attention:** when you delete a shared contact group, it will no longer be available to the other users of your eSignatures environment.

**Tip:** if the **Delete** button is unavailable, this means you don't have the necessary permissions to delete the shared group. Contact your system administrator.

- Click **Confirm** to delete.



The screenshot shows the 'Contact Groups' management interface. At the top, there is a blue header bar with the company logo, 'DOCUMENT PORTAL', 'SIGNER PORTAL', and a user profile for 'John Smith' with a settings gear icon and a language dropdown set to 'EN'. Below the header, there is a green '+ Create' button on the left and a search bar with 'Search (1)' and a magnifying glass icon on the right. The main content area has two tabs: 'Contacts' (active) and 'Contact Groups'. Under the 'Contacts' tab, there is a table with two columns: 'Name' and 'Code'. The table lists two contact groups: 'My contact group' with code '00014' and 'New Contact Group' with code '00015'. Each row has an 'Edit' button (pencil icon) and a 'Delete' button (trash can icon) to its right.

Name ▲	Code	
 My contact group	00014	 Edit  Delete
 New Contact Group	00015	 Edit  Delete

## 2.7 Access management

In the **Access management** section you have an overview of all the users who have been created in the eSignatures environment and can manage their access to the different components of the solution.


You can also create user groups and define the permissions users of that group have within eSignatures. Each eSignatures user must be part of a user group, since user permissions are defined on **user group** level. This way, you don't need to determine the permissions for each user individually, but only on group level, which saves a lot of time.

In the **Access Management** section you also determine to which **document groups** a user group must have access. Creating documents to which multiple user groups have access allows users to collaborate, since they can all access the documents uploaded to the document group.

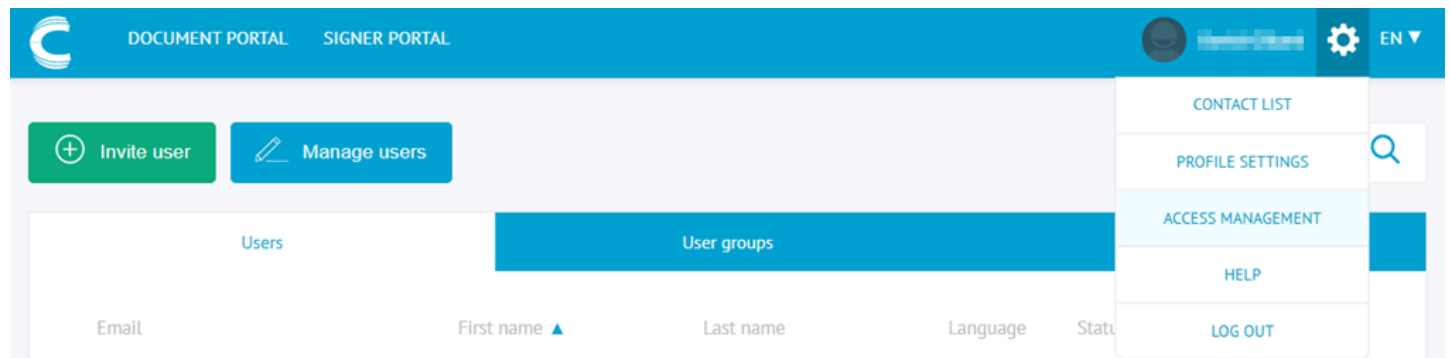
### Users vs contacts

Don't confuse *users* with *contacts*. These are two different roles within eSignatures. A *user* is a person who has access to the eSignatures environment. A *contact* is a person you add as approver, signer or receiver of a document.

### To access the Access management section:

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

**Important:** If the **Access management** button is not displayed, this means you don't have the required permissions to access it. You must be part of a user group which has the **Access to the Access management portal** permission enabled. See [Manage user groups](#) for more info.






### 2.7.1 Manage users

In the **Access management** section you have an overview of all the users that have been created in the eSignatures environment.

Note that as of eSignatures 6.0, the actual creation of the users is done in a new tool called **Service Configuration Tool (SCT)**. This tool is part of Connective's single sign-on (SSO) setup, which will allow users to access multiple Connective solutions using only one set of credentials. The SCT can be reached via the **Manage users** button in the **Access Management** section.

#### To manage users:

- Click the settings icon  the top toolbar.
- Click **Access management**.
- Click **Manage users**.

You are now redirected to the SCT.

See the [corresponding documentation section](#) for more info.

**Important:** for the **Manage users** button to work the eSignatures Config Index setting **UserManagementUrl** (under **Environment Settings**) must be filled in correctly by the admin.


## Invite users

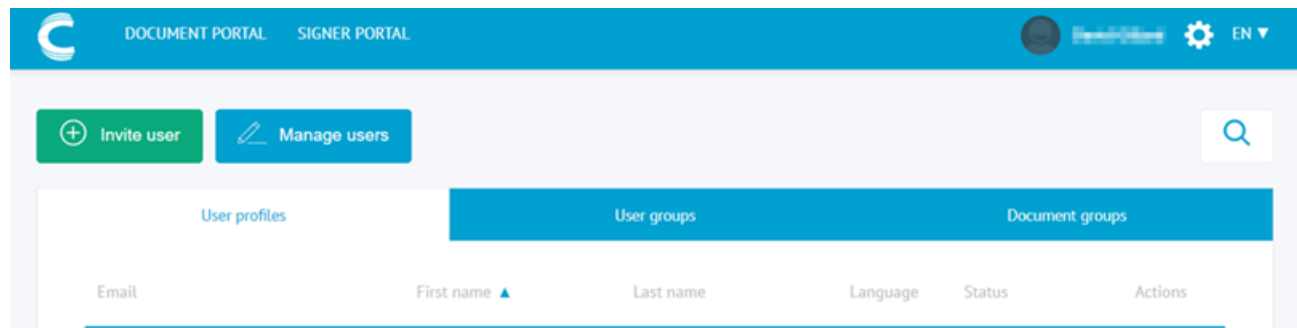
When a user has been created in the SCT, they receive an email to log in to their eSignatures account. As soon as they log in, their user profile will be created in eSignatures and they are added to the eSignatures user group that the admin has selected in the eSignatures Config Index.

If a user hasn't logged in yet, their user profile is not visible to the administrator. To solve this, the admin can invite the user to log in. As soon as the invitation has been sent, the user's basic user profile becomes visible and the user is also added to the selected user group.

**Note:** the above-mentioned scenario applies when the eSignatures Config Index default setting **IsAutomaticMembershipEnabled** is activated. If this setting is disabled, the eSignatures user profile will not be created automatically, and the admin is required to use the Invite user button

### To invite a user to log in to their eSignatures account:

- Click the settings icon  the top toolbar.
- Click **Access management**.
- Click **Invite user**.



- Enter the user's email address.

**Important:** the email address must correspond to the one used in the SCT. If you don't know the exact email address, click **Manage users** to check it in the SCT.

- Click **Confirm**.

The user's user profile will now become visible and the user is added to the selected user group as explained above.

## 2.7.2 Manage user groups

On the **Users groups** tab you have an overview of the available user groups in your eSignatures environment. If you have the required permissions, you can create, edit and delete user groups.

As mentioned in the [Access management](#) introduction, permissions are configured on user group level. This means every user belonging to the same group has the same permissions. User do not have individual permissions in eSignatures.

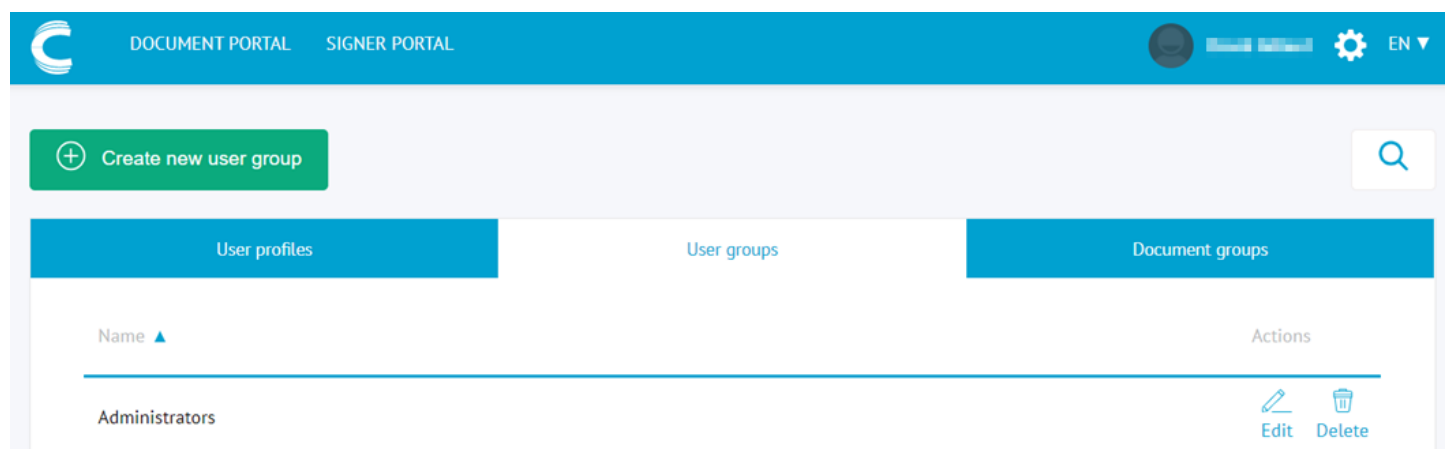
### Important:

- Never change the names of the default user groups **Administrators**, **Tenant Administrators** and **Default User Group**.
- Be careful when changing the permissions of the default user groups.

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

### Create new user group

- In the **Access 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.

**Important:** The newly created user group cannot be used as such as all its permissions are disabled. To configure a newly created user group, follow the instructions in **Edit a user group** below.

### 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?
  - **User profiles:** which users belong to this user group?
  - **Document groups:** to which document groups do the users of this user group have access?
  - **Alternate group names:** which can be used to map the user group to an externally managed group, such as in Azure AD.

#### General

On the **General** tab you determine which permissions the users of the current user group must have.

- Enable the required permissions. When a permission is enabled, its button is set to 1 instead of 0.
- When you're done, click **Save**.

Update

Default User Group

General

User profiles

Document groups

Alternate group names

#### GENERAL APPLICATION SETTINGS

##### Access to the Document portal

Create new shared contact group(s)

Create new shared contact(s)

Delete existing shared contact group(s)

Delete existing shared contact(s)

Edit existing shared contact group(s)

Edit existing shared contact(s)

View shared contact group(s)

View shared contact(s)

##### Access to the Signer portal

Signer portal - Bulk sign documents

Cancel

Save

**Note:** before users are able to view, create, edit or delete shared contacts and contact groups, the eSignatures admin must provide them with the necessary permissions.

Like all other permissions, the contact sharing permissions are managed on User Group level. The instructions below refer to contacts but can be used for contact groups as well.

#### To assign sharing permissions:

- Make sure the permission **Access to the Document Portal** is enabled.
- All sharing permissions depend on this permission.
- Now determine which sharing rights the user group should have:
  - If the user group should only be able to view shared contacts, but not edit them or delete any new ones, select **View shared contact(s)**.
  - If the user group should be able to create new contacts, select **Create new shared contact(s)**. The corresponding viewing, editing and deletion permissions are enabled automatically, but each permission can be disabled individually if necessary.
  - Click **Save** to save the settings.
- Now make sure the user group contains all users that should have sharing permissions (explained below).

#### User profiles

On the **User profiles** tab you determine which users must be part of the user group.

- 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
User profiles
Document groups
Alternate group names

Search for a name

USER PROFILES NOT IN THE USER GROUP

My Documents

mydocuments@company.com

+

My Documents

mydocuments@company.com

+

My Documents

mydocuments@company.com

+

My Documents

mydocuments@company.com

+

My Documents

mydocuments@company.com

+

USER PROFILES IN THE USER GROUP

My Documents

mydocuments@company.com

×

Cancel
Save

Document groups

On the **Document groups** tab you determine which actions the users of the current user group can do in which document group. By default, users of a new user group don't have any permissions at all.

The available document groups are listed in the **Document groups** column on the left.

- Select the required document group.
- Select the required permissions on the right:
  - Upload document(s)
  - Sign document(s)
  - Download document(s)
  - Delete existing document(s)
  - View
- Click **Save**.

**Attention:** when you don't select any permissions for a particular document group, the users of the current group won't be able to use the document group.

Update
Default User Group

General
User profiles
Document groups
Alternate group names

DOCUMENT GROUPS

My Documents

My Documents

Test Documents

PERMISSIONS

Upload document(s)

Sign document(s)

Download document(s)

Delete existing document(s)

View

Request audit trail

Cancel
Save

The settings you configure here are automatically stored in the [Document groups](#) settings (on the corresponding tab) as well.

Note you cannot add or remove a Document group on this tab. This must be done one level higher. See [2.7.3 Manage document groups](#) for more information.

#### Alternate group names

On the **Alternate group names** tab you can add an external group to which the current user group will be mapped. You can for instance map user groups to Azure AD or any other tool that manages user groups externally. This way, the external user groups inherit all permissions you've configured for the current user group.

To add an alternate group name:

- Click the **Alternate group names** tab.
- All available alternate group names to which this user group can be mapped are listed.
- To add a new one, enter the name of the alternate group in the text field, and click **Add**.
- Click the **Save** button at the bottom to save all changes made.

The screenshot shows a web interface for managing user groups. At the top, there's a header bar with a green 'Update' button and a dropdown menu currently showing 'Administrators'. Below this is a tabbed interface with four tabs: 'General', 'User profiles', 'Document groups', and 'Alternate group names'. The 'Alternate group names' tab is selected and highlighted with a blue underline. Under this tab, there is a text input field containing 'Azure AD alternate group' and a green 'Add' button to its right. Below this is a section titled 'ALTERNATE GROUP NAMES' which contains a table-like structure. It has one row with a text input field containing 'Azure AD alternate group' and a red 'Delete' button to its right. At the bottom of the interface, there are two buttons: a grey 'Cancel' button and a green 'Save' button.


#### Delete a user group

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

## How to create a *Signers only* user group?

In this topic you'll learn how to configure a Signers only user group. Users who experience issues accessing their documents to sign via email, or who can't download their signed documents via email, can be added to this user group. This way, they can access the Signer Portal and sign/download their documents directly in the portal.

### Step 1: Log in to eSignatures as admin

- Log in to eSignatures as admin.
- Click the settings icon  in the top toolbar.
- Click **Access management**.

### Step 2: Create the *Signers only* user group

- Click the **User groups** tab.
- Click **Create new user group**.
- Name the user group, and click **Confirm**.

E.g. **Signers only**.

- By default, only one permission has been enabled for a new user group: **Access to the Signer Portal**.

Make sure no other permissions are enabled. To do so:

- Click the **Edit** button next to the user group you just created.
- On the **General** tab, verify that only **Access to the Signer Portal** is enabled.
- If nothing needs to be modified, click **Cancel**.

### Step 3: Create the users that require access

- Still within **Access management**, click the **Manage Users** button.

You are now redirected to the Service Config Tool (SCT) where you can create new users.

- In the SCT, click **Users** and then click **Create**.
- Enter the user's personal information: First Name, Last Name and Email.

**Important:** their email address **must** be the same as the one that was used during the signing flow. If you use a different email address here, users will not have access to the documents that have already been sent to their other email address.

- Enter a **Username** and **Password** for that user, and click **Create**.
- Then click **Save** to save the settings.
- Follow the steps above to add additional users.
- The new users now receive an invitation via email to log in to their eSignatures account.

### Step 4: Invite the new users

- You can now wait for the new users to log in to their eSignatures, for their user profile to become visible in your eSignatures **Access management** section.
- Or you can click the **Invite user** button in **Access management**.
- Enter the user's exact same email address as in the SCT, and as was used to start the signing flow in the first place.
- Click **Confirm**.
- The user profile is now added to **User profiles** list.
- You can now move the user profile to the **Signers only** user group.

### Step 5: Add the newly added users to the *Signers only* group

- Still within eSignatures **Access management**, click the **User groups** tab again.
- Click **Edit** next to the **Signers only** user group.
- Click the **User profiles** tab.
- In the **User profiles not in this user group** column, click the + icon next to the users you want to add to this group.
- Changes are saved automatically. Click **Cancel** to exit the screen.



### 2.7.3 Manage document groups

On the **Document groups** tab, you have an overview of all the available document groups in your eSignatures environment.

A document group is a container of documents. By means of document groups, documents can be shared amongst different users. When multiple users have access to the same document group (other than MyDocuments) they can view and/or edit (depending on the permissions) the documents other users have uploaded.

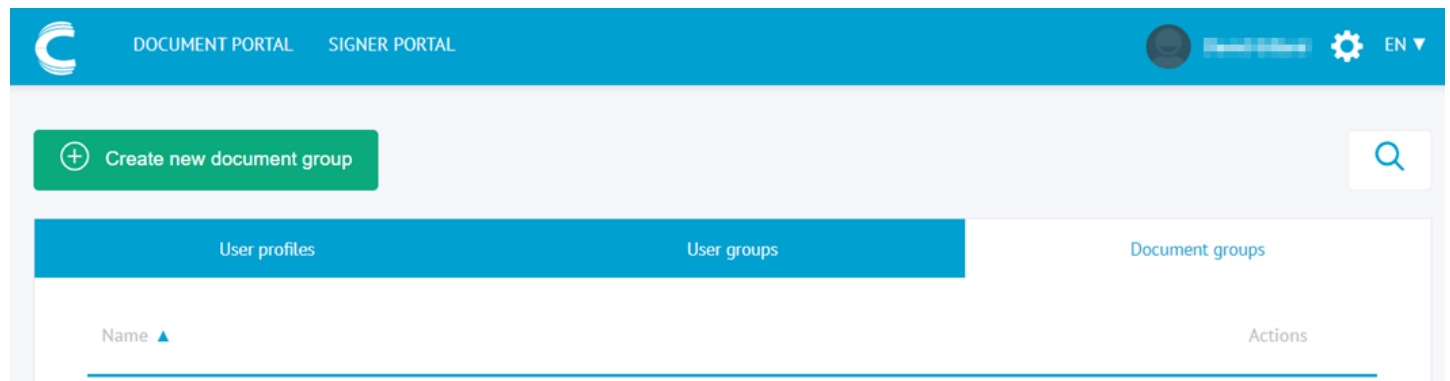
Per document group you can now also select multiple **Themes** to choose from while uploading a package. A theme is a collection of branding settings that determine the look and feel of eSignatures. This way, packages uploaded under a certain theme can have a different look.

#### Important:

- Never rename or delete the default document group **My Documents**! Doing so could lead to serious environment issues.
- If you want to apply special permissions to a document group, you are recommended to create a new document group.

#### Create a new document group

- In the **Access 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.

**Important:** the newly created document group cannot be used as such. You must first assign the appropriate permissions to the user groups that need access to the document group. To configure a newly created document group, follow the instructions in **Edit a document group** below.

#### Edit a document group

- Click the **Edit** button next to the document group you want to edit.

#### Select a User group

Now select the user groups that must have access to the document group and configure which permissions the user groups should have.

- The different user groups that are created in the system are listed under the User groups tab.

## Update document group

CBER - Document Group

Provide a document group name.

User groups

Themes

USER GROUPS	PERMISSIONS
Administrators	Upload document(s) <input checked="" type="checkbox"/> 0
CBER - contact management	Sign document(s) <input checked="" type="checkbox"/> 0
CBER - administrators	Download document(s) <input checked="" type="checkbox"/> 0
CBER - Document Portal	Delete existing document(s) <input checked="" type="checkbox"/> 0
CBER - Signer Portal - Remove this text after linking	View <input checked="" type="checkbox"/> 0

1

2

>

Cancel

Save

- Select the required user group in the **User Groups** column.
- Enable the permissions you want the user group to have in the document group. All permissions are by default disabled for a newly created 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.

### Select a theme

Per document group you can now also select multiple Themes to choose from while uploading a package. A theme is a collection of branding settings that determine the look and feel of eSignatures. This way, packages uploaded under a certain theme can have a different look.

- Click the **Themes** tab.
- By default, the **System theme** is selected.
- The available themes the admin has created for your environment are listed under **Themes not in the document group**.
- Click the plus sign next to every theme you want to make available to the document group.

The themes you've selected are now listed under **Themes in the document group** and can be selected as default theme in the **Default theme** drop-down list.

## Update document group

CBER - Document Group

Provide a document group name.

User groups

Themes

System theme

Default theme

Search for a theme

THEMES NOT IN THE DOCUMENT GROUP

Connective theme

+

THEMES IN THE DOCUMENT GROUP

System theme

×

Cancel

Save

- In the **Default theme** drop-down list, select which theme must be used as default when uploading a package to this document group.

Note that initiators may still choose a different theme during the upload (provided it's one of the themes you've added to the document group).

- When you're done, click **Save**.

### 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.

## 2.9 WebPortal FAQ

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

## 2.9.1 Uploading documents

[Which file types can I upload?](#)

[What is the size limit for documents I upload?](#)

[Why are PDF Portfolios not supported?](#)

[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.

## **Why are PDF Portfolios not supported?**

Even though it's technically possible to upload a PDF Portfolio, it is not supported to legally sign it.

This is because a PDF portfolio may contain a wide range of file types that are not supported by eSignatures. A PDF portfolio may for instance contain e-mail messages, spreadsheets, CAD drawings, PowerPoint presentations, etc. As a result, a signer will only be able to view and sign the PDF cover sheet, and not the actual files the Portfolio contains, which renders the entire Portfolio invalid.



## 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. *This note does not apply when using itsme through OpenID Connect.*

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

## 2.9.2 Signing documents

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

[Why can't I select a particular signing method?](#)

[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 are the different signing methods?

The following signing methods are supported in eSignatures.

### Important:

- Your system admin may not have enabled all signing methods listed below in your eSignatures solution. If a desired signing method is not available, either the admin needs to enable and configure it in the Config Index, or the selected contact/contact group members don't contain the necessary info to use the signing method, as explained above.
- Since eSignatures 6.2, the **DisplayName** of the signing methods is configurable, so the name of the signing methods you see in your environment may not correspond to the ones listed below, but their behavior will be the same.

### Tips:

- If a desired signing method is not available, contact your administrator to enable it in the Configuration Index.
- If a desired signing method is not selectable, some elements are missing in the contact info and/or in the configuration. See [Why can't I select a particular signing method?](#)

### 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.

#### Handwritten

Select **Handwritten** if you want signers to type in their name in a handwritten font using their keyboard.

Signers will be able to choose from a number of preconfigured handwritten fonts in the WYSIWYS.

#### 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, make sure the **Connective SignID software** or **Connective Browser Package** (both for Windows and macOS) has been properly installed. Otherwise eID signing will not work.

To learn how to install the Connective SignID software, see the [SignID section](#) on the documentation website. For the Connective Browser Package, click [here](#).

#### 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 manual + eID signing method for the first time you, make sure the **Connective SignID software** or **Connective Browser Package** (both for Windows and macOS) has been properly installed. Otherwise eID signing will not work.

To learn how to install the Connective SignID software, see the [SignID section](#) on the documentation website. For the Connective Browser Package, click [here](#).

#### 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 Finition Stylus (only on iPad)

Signature pads and the Bamboo Finition 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 Finition 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, make sure the **Connective SignID software** or **Connective Browser Package** (both for Windows and macOS) has been properly installed. Otherwise eID signing will not work.

To learn how to install the Connective SignID software, see the [SignID section](#) on the documentation website. For the Connective Browser Package, click [here](#).

## **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 their 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. *This note doesn't apply when using itsme*

*through OpenID Connect.*

- When using itsme signing in packages, each document within the package must be signed individually, which means QuickSigning is not supported.
- Itsme signing always requires a Signing Policy. The default Signing Policy is drafted by Belgian Mobile ID and configured by Connective in the Configuration Index. If you want to combine itsme signing with a legal notice, the setting **CombineLegalNoticeAndSigningPolicy** must be enabled in the Configuration Index. *This note doesn't apply when using itsme through OpenID Connect.*

### **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.

## Why can't I select a particular signing method?

It may occur that one of the signing methods you want to select is greyed out. This means some elements are missing in the configuration.

SIGNING METHOD	CAUSE	SOLUTION
SMS OPT is greyed out	The contact's info doesn't include their phone number.	<a href="#">Edit the contact's info</a> and add their phone number in a valid format.
BeID, Manually+BeID or BeLawyer is greyed out.	Mandated signing rules are applied, and the contact's info doesn't contain all required data.	Contact your system admin to know which mandated signing are applied, and which contact data is required. Then edit the contact's info accordingly.
iDIN is greyed out	The iDIN settings are not configured correctly or are incomplete in the Config Index.	<a href="#">Edit the contact's info</a> and add their BIN number. Contact your system administrator or consult the Configuration Documentation.
itsme is greyed out.	Mandated signing rules are applied, and the contact's info doesn't contain all required data.	Contact your system admin to know which mandated signing are applied, and which contact data is required. Then edit the contact's info accordingly.
	The Itsme settings are not configured correctly, or are incomplete in the Config Index.	Contact your system administrator or consult the Configuration Documentation.

### 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 SignID software, see the [SignID section](#) on the documentation website.

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 SignID software, see the [SignID section](#) on the documentation website.

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.

## 2. Which operating systems are supported?

The following operating systems are supported:

### **Microsoft Windows**

- Microsoft Windows 10 (64-bit)
- Microsoft Windows 8.1 (64-bit)

### **Mac OS**

- macOS Big Sur
- macOS Catalina

### **Android**

- Android 11
- Android 10

### **iOS**

- iOS 14
- iOS 13

Other operating systems are officially *not* supported.

## Which web browsers are supported?

The following web browsers are supported:

### Windows

- Microsoft Edge n-2
- Internet Explorer 11

Earlier versions of Internet Explorer are not supported.

- 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.

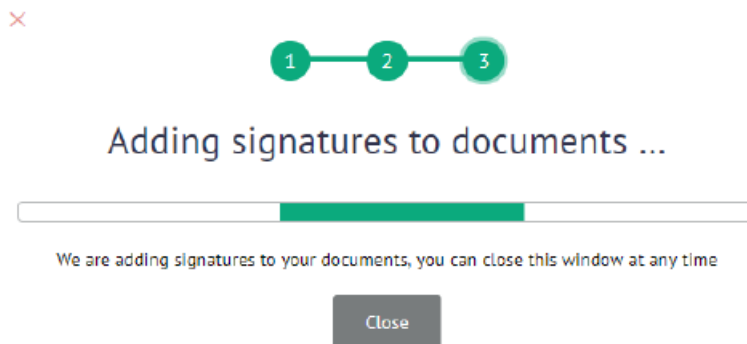
---

## How does asynchronous signing work?

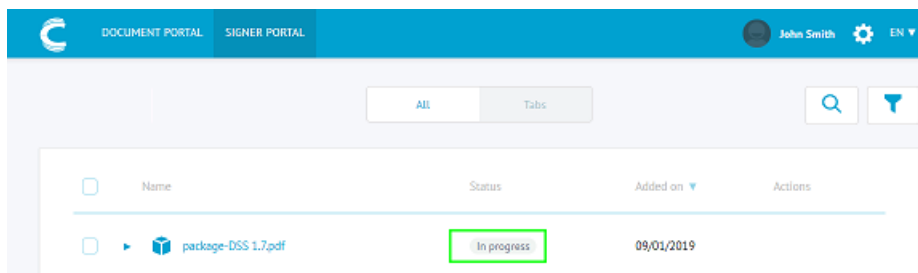
Asynchronous signing allows you to close the signing session and let the signing run in the background. This way, you can work on other things while your documents are being signed. This comes in handy especially when signing large documents that take quite some time to be signed.

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**.  
**Note:** this checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.
- 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.



- 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.

#### 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.

## My account

[I can't access my account](#)

[I get a "Bad request" error when trying to log in](#)

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 <b>Forgot Password?</b> 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 administrator.

## I get a "Bad request" error when trying to log in to my account

POSSIBLE CAUSE	SOLUTION
You're using a bookmark from the login page that includes "auth-" in the URL	Navigate to your environment using the main page
Caching issue	Clear your browser's cache
If the two scenarios above don't apply	Try resetting your password by clicking the <b>Forgot password?</b> button

If none of the suggested solutions solves the problem, please contact [service@connective.eu](mailto:service@connective.eu).

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 start over.
Another user may have rejected the document you were both required to sign.	You can no longer take action on the document.
The document you're trying to access may have already been signed.	You no longer need to take action on the document.

## My contacts and contact groups

[I'm unable to edit a contact or contact group](#)

[I'm unable to delete a contact or contact group](#)

[Contacts don't receive the documents I send for signing/approval](#)

[Contacts don't receive reminders about documents to sign](#)



I'm unable to edit a contact or contact group

POSSIBLE CAUSE	SOLUTION
The contact/contact group is shared, and you don't have the necessary permissions to edit shared contacts.	If you have access rights to Access Management, modify the permissions of the User Group you belong to. If you don't have access, contact your administrator.

I'm unable to delete a contact or contact group

POSSIBLE CAUSE	SOLUTION
The contact/contact group is shared, and you don't have the necessary permissions to edit shared contacts.	If you have access rights to Access Management, modify the permissions of the User Group you belong to. If you don't have access, contact your administrator.

Contacts don't receive the documents I send for signing/approval

POSSIBLE CAUSE	SOLUTION
You've modified the contact's email address after documents have already been sent to their previous email address.	Reassign the document to the modified contact's email address.  Or if that's not possible, Send the documents again using the correct, updated email address.

Contacts don't receive reminders about documents to sign

POSSIBLE CAUSE	SOLUTION
You've modified the contact's email address after documents have already been sent to their previous email address.	Reassign the document to the modified contact's email address. Or if that's not possible, Send the documents again using the correct, updated email address.

2.10.4 eSignatures is running slow

POSSIBLE CAUSE	SOLUTION
A large number of documents are loaded into the system.	Delete old documents you no longer need.

## 2.10.5 Signing documents

[I'm not mandated to 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](#)

[The Connective app doesn't open](#)

I'm not mandated to sign

When trying to sign with eID, BeLawyer, iDIN 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, or returned by iDIN.	<p>Make sure the info in your contact matches the certificate info on the eID card, BeLawyer card, iDIN or itsme account.</p> <p>When the system admin applied mandated signing rules in the Config Index to check the names and birthdate, you need to enter the given names and birthdate in the contact info.</p> <p>If the info still doesn't match, contact your administrator to reduce the <b>MatchLevel</b> in the Config Index.</p>

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) <b>Important:</b> the Wacom Bamboo Fineline can only be used on iPad.



I'm unable to sign with itsme

POSSIBLE CAUSE	SOLUTION
You're using an unsupported Operating System: macOS Mojave v10.14.	Use a <a href="#">supported Operating System</a> .
You're using an unsupported browser: Safari v12.0.	Use a <a href="#">supported browser</a> .

I'm not able to sign a PDF/A document

In the rare case this happens, contact your 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.

## 3. Form fillers

The **Form fillers** section is intended for end users who exclusively fill out forms on eSignatures documents.

Note that it's currently not supported to create forms in the eSignatures WebPortal. In the current version, only API users can create form fields on the documents they're sending.

The documents to be filled out can be accessed via email or in the Signer Portal, like any other documents to be approved or signed.

Once the forms have been filled out, the documents are sent to the required approver(s) or signer(s), depending on the setup of the signing flow. Note that the form filling must not necessarily take place before the approval. In the current setup, it's entirely possible to approve documents of which the forms have not been filled out yet.

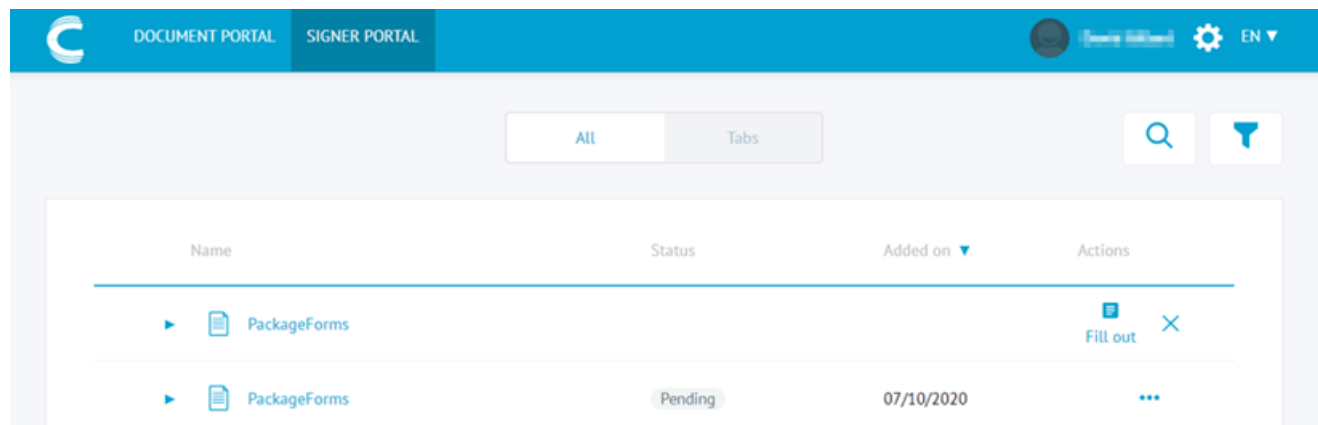
## 3.1 How do I fill out forms in eSignatures?

You can access your form in various ways:

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

### Filling out forms in the Signer Portal

- Log in to your eSignatures account.
- Click the **Signer Portal** tab.
- Click the **Fill out** icon next to the package whose forms you want to fill out.



- You can find the different approval steps in **Filling out forms step-by-step** below.

### Filling out forms 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 (**one-time URL**) 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 fill out the document. If the link expired, click **Request a new email** to receive a new link.

- You can find the different approval steps in **Filling out forms step-by-step**.

### Filling out forms in a mobile app

This works in the same way as approving or signing documents in a mobile app. See the corresponding sections for more information.

### Filling out forms step-by-step

- Access the documents that must be filled out.
- Check the required checkboxes.
- Fill out the required text fields.

1 / 1 Document1

First name: John  
Last name: Doe

I speak and understand (tick all that apply):

- ☒ English
- ☒ Polish
- ☒ German
- ☐ Spanish
- ☒ French

Signature:

Another person will sign this field.

Reject

Save fields

**Tip:** you may also be able to **download** the document before filling it out. This way you can read it in a PDF Viewer or even print it if necessary. To download the forms document, click the download icon.

**Note:** whether the download icon is available depends on whether the initiator enabled it.

- When you're done, click **Save fields**.
- When the document has been filled out, the message **You have filled out the forms in this package** is displayed at the bottom of the screen.
- The package will now be transferred to the next person in line. This may be an approver, or a signer.

#### Rejecting a form

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

Reject

Another person will sign this field.

- 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 will be notified that the document has been rejected and why.

The approver or signer who was supposed to take action after the form had been filled out will not be notified, since there is no longer a document to approve or sign.

## 4. Approvers

The **Approvers** section is intended for end users who exclusively approve documents via email or a mobile app.

## 4.1 How do I approve a document?

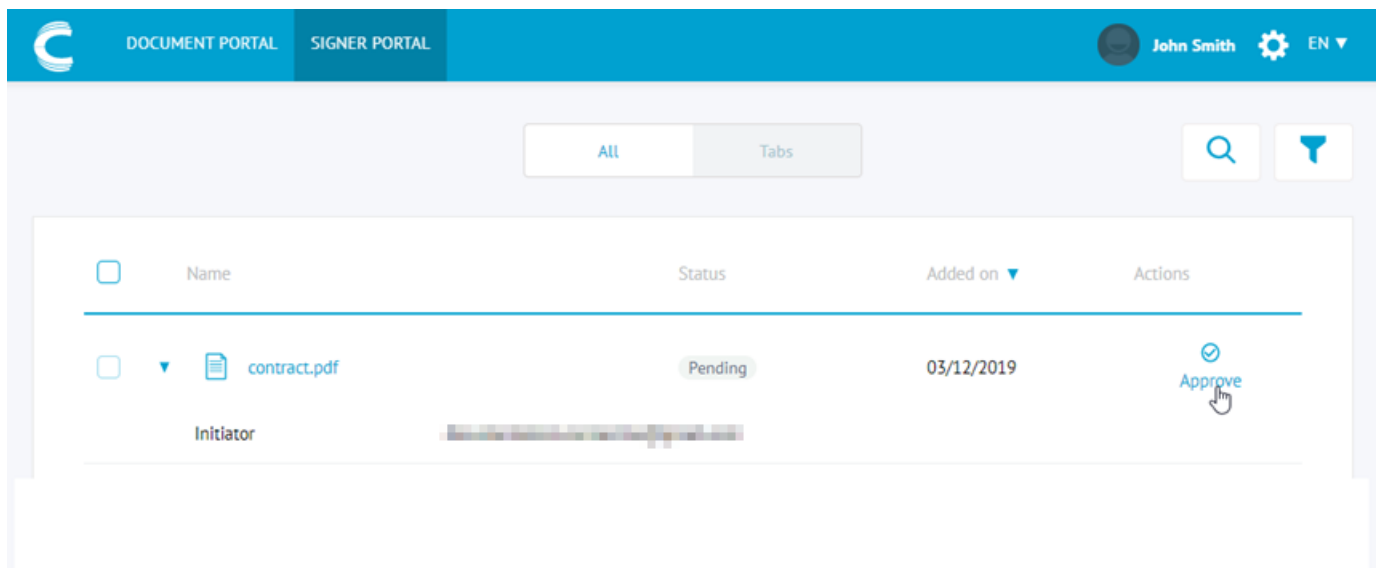
You can approve documents in various ways:

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



#### 4.1.1 Approving in the Signer Portal

- Log in to your eSignatures account.
- Click the **Signer Portal** tab.
- Click the **Approve** icon next to the package you want to approve.



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

#### 4.1.2 Approving via email

- Open the email you received from your eSignatures solution. In our standard demo environment you receive an email from [esigner@connective.be](mailto:esigner@connective.be).
- Click the secure link (one-time URL) 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 approve your document or package with name marker.pdf Inbox x

eSigner    
to me ▼

Dear John Smith,

Please click on the link below to approve your document or package (Name: marker.pdf):

[https://connective.be/esigner/.../marker.pdf?token=...&signature=...](#)

Kind Regards,

John Smith 

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

#### 4.1.3 Approving 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 approve icon next to the document you want to approve.

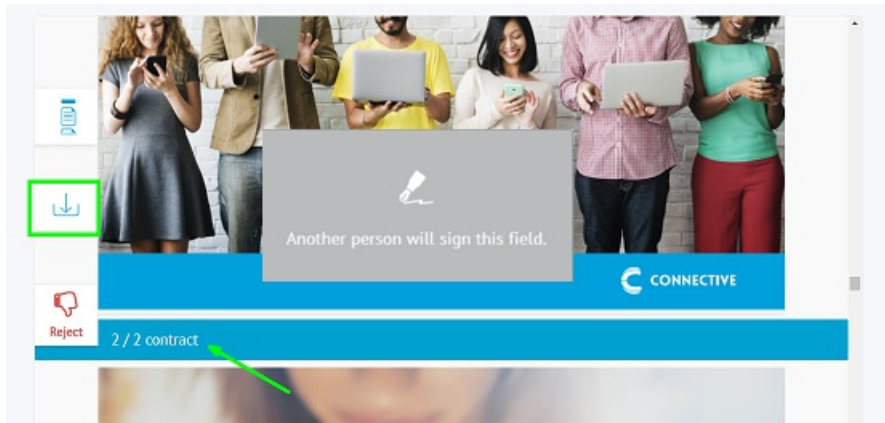
See the separate chapter [Approving in the Connective app](#) general instructions on how to use the standard demo app.

#### 4.1.4 Approving a document step-by-step

##### Approving a package

- Access the documents that require your approval.
- If applicable, verify that any checkboxes and/or text fields have been filled in correctly.
- 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.



**Tip:** you may now also be able to download the document before approving it. This way you can read it in a PDF Viewer or even print it if necessary. To download the approval document, click the download icon.

**Note:** whether the download icon is available depends on whether the initiator enabled it.

Also note that if your document contains form fields, only the original values of the fields will be displayed when downloading the files, not the values that have been filled in or modified by the form filler. This is intended behavior in the current version.

- When you've scrolled down to the last page, the checkbox at the bottom now becomes available. Check it to declare you've read all documents and shall comply with the following policies: Terms of Use, Privacy Policy and Cookie Policy.

**Note:** the message of the checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.

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

- Click **Approve**.

1 / 1 Document1

First name:  \*

Last name:  \*

I speak and understand (tick all that apply):

☒ English


☐ Polish

☐ German


☐ Spanish

☐ French

Signature:



Another person will sign this field.

 **Reject**


☒ I declare that I have read all documents and shall comply with the following policies: [Terms of Use](#), [Privacy Policy](#) and [Cookie Policy](#).

- When the package has been approved, the message **You have approved this package** is displayed at the bottom of the screen.
- The package will now be transferred to the next person in line. This may be another approver, or a signer.

### Rejecting a package

- Access the documents that require your approval.
- Click the **Reject** button on the left-hand side of the document.



 **Reject**

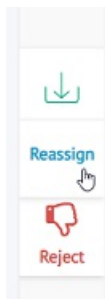


Another person will sign this field.

- 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 will be notified that the document has been rejected and why.
  - The signer who was supposed to sign after your approval will not be notified, since there is no document to sign.

### Reassigning

- Open the document you want to reassign.
- Click the **Reassign** button on the left-hand side of the document.



Enter the following information about the person you are reassign the approval to:

- Email

**Note:** if the email address corresponds to a known user in eSignatures, not only do they receive an email notification to approve their document, but the document will also be available for approval in their Signer Portal.

- Language

The language is by default set to the preferred language of the previous approver. Select a different preferred language if necessary.

The emails the approver receives will be drafted in the language you select here.

- First Name
- Last Name
- Reason for reassignment
- When you're done, click **Reassign**.

All parties involved receive an email notification informing them about the change:

- The initiator who sent the document is notified that one or more approvers have changed.
- The new approvers are notified a document requires their attention.

## 4.2 Approving in the Connective app

4.2.1 How do I install the app?

4.2.2 How do I approve in the Connective app?

## 4.2.1 How do I install the Connective app?

### Minimum System Requirements

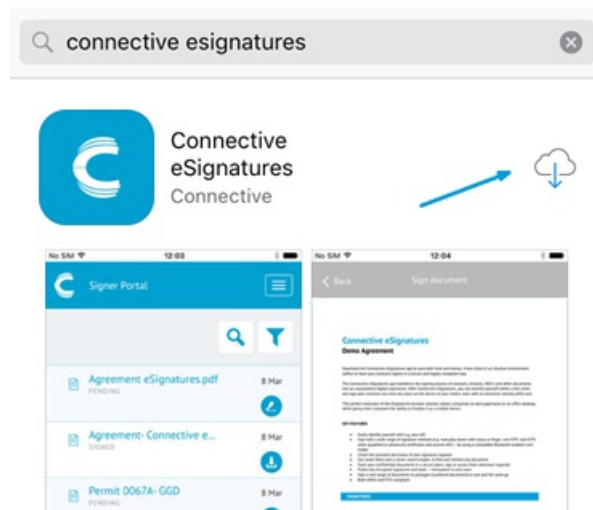
- Android 4.3 and up.
- iOS 9.3 or later. Compatible with iPhone, iPad and iPod Touch.

### 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.



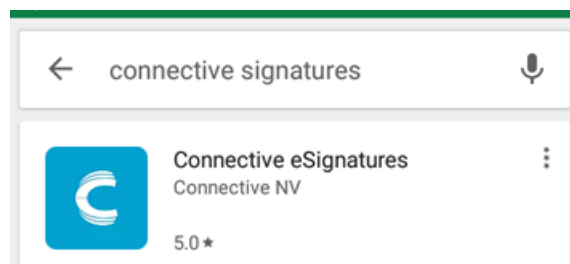
- When the download is complete, the **Open** button is displayed.
- Tap **Open** to open the app.



- 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.

## 4.2.2 How do I approve in the Connective app?

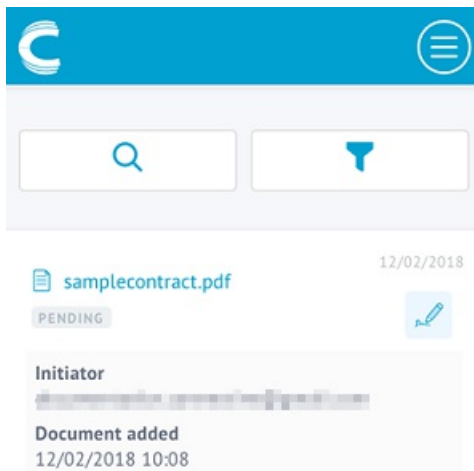
### 1. Open the app

- Click the link in the email you received on behalf of the initiator.
- When you're prompted to open the app, tap **Open**.

**Attention:** When using Safari on an iPad with iOS 13, you won't be prompted to open the Connective app by default, since Safari is by default configured to open in Desktop mode. See [The Connective app doesn't open](#) to learn how to resolve this.

### 2. Select a document to approve

- Tap the document you want to approve. Your document is opened in a new screen.



- Read the entire document and scroll down to the last page.
- When you've scrolled down to the last page, the checkbox at the bottom now becomes available.
- Check it to declare you've read all documents and shall comply with the following policies: Terms of Use, Privacy Policy and Cookie Policy.
- Click **Approve**.
- When the package has been approved, the message You have approved this package is displayed at the bottom of the screen.
- The package will now be transferred to the next person in line. This may be another approver, or a signer.

### 3. Reject a document

- Tap the required document.
- Click the **Reject** button.
- Enter a reason for rejection, and then click **Reject**.

Note that entering a reason for rejection is mandatory.

- The initiator who sent the document will be notified that the document has been rejected and why.
- The signer who was supposed to sign after your approval will not be notified, since there is no document to sign.

## 4.3 Approvers FAQ

In this section we address the questions that approvers might have concerning eSignatures.

The questions are bundled per topic where possible.

## My account

[I can't access my account](#)

[I get a "Bad request" error when trying to log in](#)

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 <b>Forgot Password?</b> 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 administrator.

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 start over.
Another user may have rejected the document you were both required to sign.	You can no longer take action on the document.
The document you're trying to access may have already been signed.	You no longer need to take action on the document.

## Emails

I don't receive any emails with documents to sign

I replied to an email sent by eSignatures but never received a response

When trying to download my signed documents via email, I keep getting the message "Your url has expired, please request a new email"

## I don't receive any emails with documents to approve/sign

POSSIBLE CAUSE	SOLUTION
The emails end up in your Spam or Junk folder.	<p>Check your Spam or Junk folder.</p> <p>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.</p> <p>In our standard demo solution, the email address to add is <a href="mailto:esigner@connective.eu">esigner@connective.eu</a></p>
The eSignatures solution doesn't have your correct email address.	Contact your initiator and communicate your correct email address.
The initiator may have modified your email address in the contact's info, but documents that have already been sent to your previous email address are not updated retroactively.	Contact your initiator and communicate your correct email address.
The approver/signer/receiver is using an external SPAM filter that is blocking the incoming mail.	<p>Check with the approver/signer/receiver which external SPAM filter they are using.</p> <p>Depending on the configuration, the mail address from which eSignatures is sending its emails may need to be added to the Safe Sender list.</p>



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.

## 5. Signers

The **Signers** section is intended for end users who sign documents via email or mobile app, and do not have access to the eSignatures WebPortal.

## 5.1 Signing step-by-step

In this section signers can 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.

### 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 (**one-time URL**) 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



- If your package contains any optional documents, an overview is provided of the **mandatory** and the **optional** documents.

### Select optional documents

This package contains optional documents. Please review and select the documents you would like to retain.

#### Mandatory documents

In this session you will be processing these documents. You can click on the thumbnails to preview them.



Polis - Nieuwe verzekering (117360)



Productfiche (117361)

#### Optional documents

Please click on the thumbnails to preview the document and select the ones you choose to process.

Select all



Algemene voorwaarden (117362)

Select



IPID fiche (117364)

Select

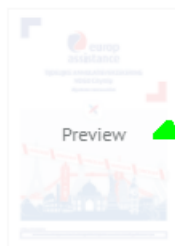
Next

- If you want to sign optional documents, click the **Select** button next to each optional document you want to sign.  
Or to sign all optional documents, click **Select all** beneath **Optional documents**.
- If you don't want to sign any of the optional documents, click **Next** directly.
- **Tip:** Click the thumbnail of an optional document to open it in preview. That way you can read through the document and decide whether you want to sign it. When you're done, click the **Back** button.

### Optional documents

Please click on the thumbnails to preview the document and select the ones you choose to process.

Select all

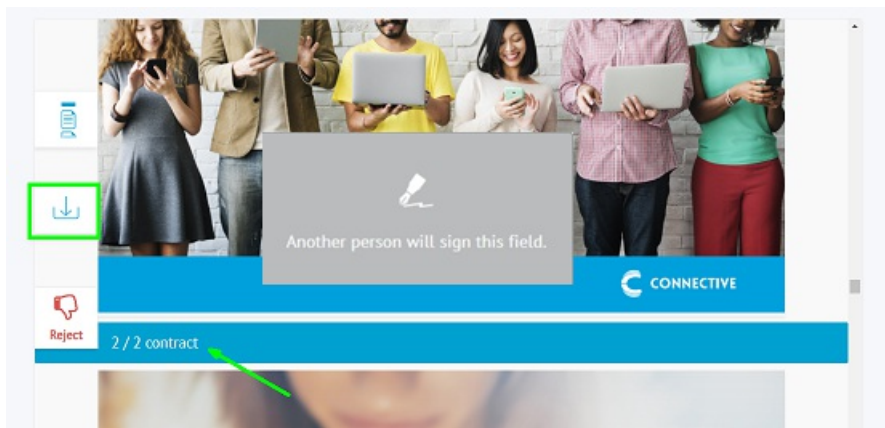


Algemene voorwaarden (117362)

Unselect

Next

- After you've clicked **Next**, read the entire document (if still necessary) 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.



**Tip:** you may now also be able to download the document before signing and print it, to read it on paper for instance. To do so, click the download icon.

**Note:** whether the download icon is available depends on whether the initiator enabled it.

Also note that if your document contains form fields, only the original values of the fields will be displayed when downloading the files, not the values that have been filled in or modified by the form filler. This is intended behavior in the current version.

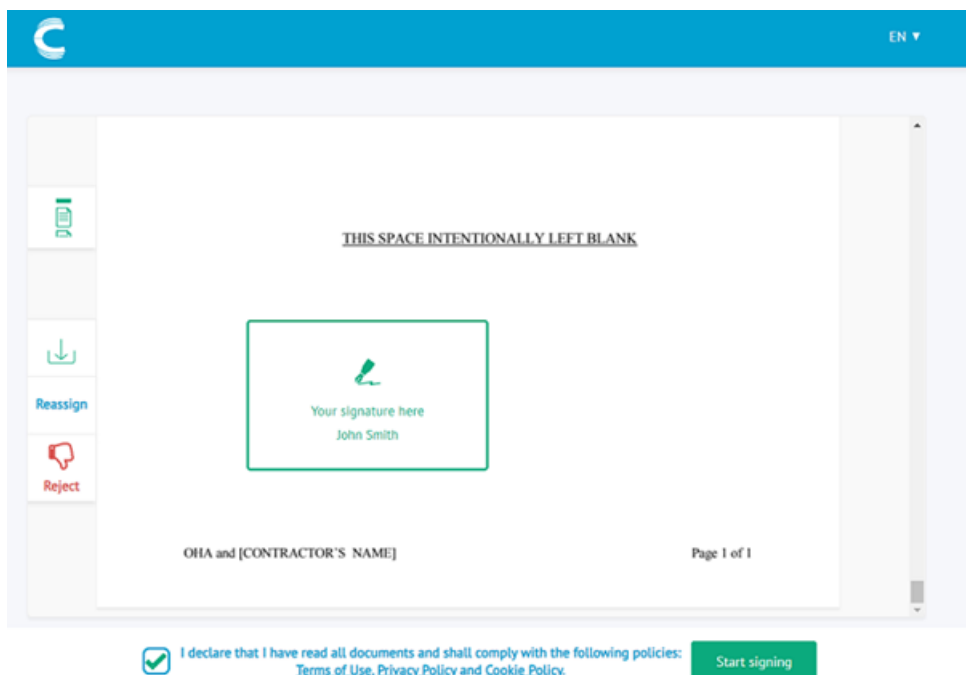
- When you've scrolled down to the last page, the checkbox at the bottom becomes available.
- Check it to declare you've read all documents and shall comply with the following policies: Terms of Use, Privacy Policy and Cookie Policy.
- Then click **Start signing**.

**Note:** this checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.

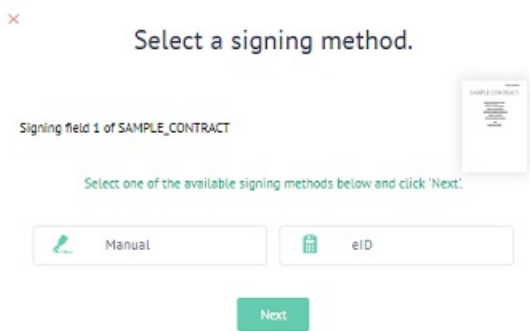
**Tip:** click the links to consult the Terms of Use, Privacy Policy and Cookie Policy respectively.

- If you are not the right person to sign the document, click the **Reassign** button and assign it to someone else.

**Note:** whether the **Reassign** button is available depends on the configuration by the administrator. See [How do I reassign a document?](#) for more information



- 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.

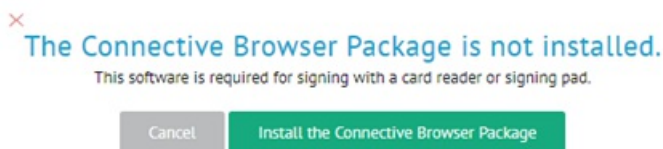


Install the Connective browser package

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.

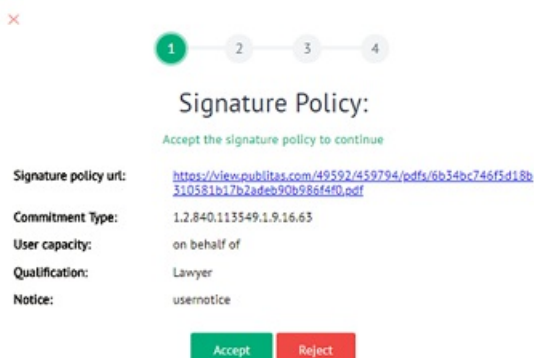


After the browser package has been installed, you'll restart at the first step of the signing process.

**Note:** 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.

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**.

#### 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.

- 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

- When multiple signing methods have been selected for you to choose from, the same signing methods must have been selected for each document of the package. It's not possible to QuickSign a package if you can choose from one pair of signing methods on one document, and choose from another pair on another document.
- The following signing methods cannot be combined with QuickSigning: biometric and itsme. 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.
- When your package contains multiple legal notices, their content must be identical (per signer) for QuickSigning to work.

### 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

### ***Sign Handwritten***

- Type in your full name in the required field.
- Select one of the fonts. Note that these fonts are preconfigured and cannot be modified.



## Sign handwritten

Signing field 1



Sign with your full name in the field below and choose a handwritten font.

☒

John Smith

☐

John Smith

☐

John Smith

☐

John Smith

☐

John Smith

Cancel

Next

- 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.

### 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** (for Windows and macOS) been properly installed. Otherwise eID signing will not work.

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.

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** (for Windows and macOS) been properly installed. Otherwise manual + eID signing will not work.

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.

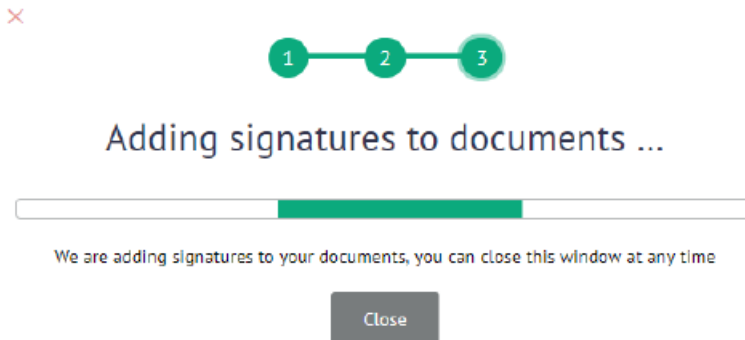
- Enter the code you received and click **Next**.

**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.

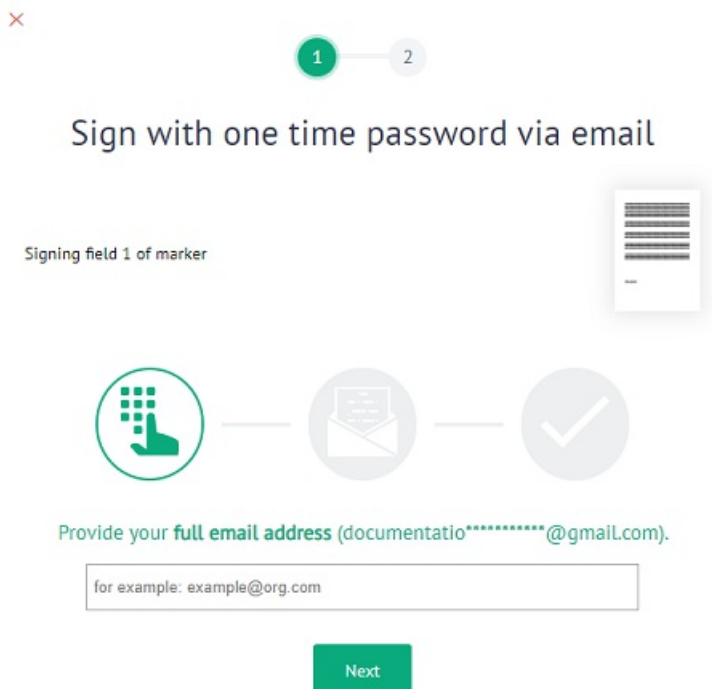
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



**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.



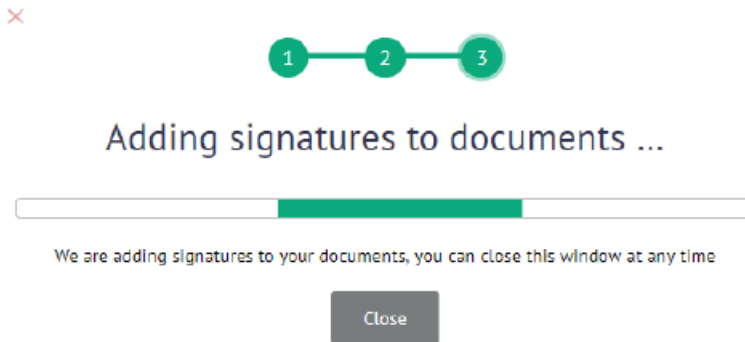
- Click **Next**. If you entered the correct email address, an email is sent to the email address you entered.
- Enter the code you received in the confirmation field.

**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.



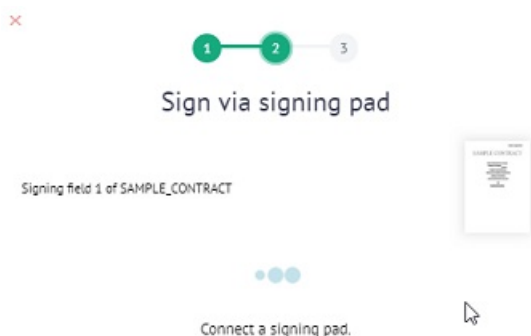
### ***Sign via signing pad (biometric)***

- 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, which means QuickSigning is not supported.



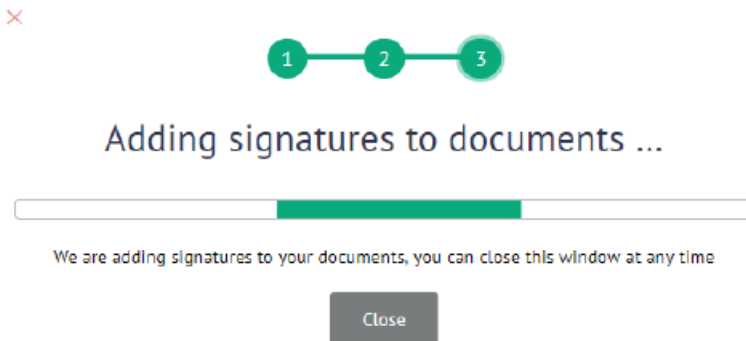
- When the device is successfully connected, "Please sign using your signature pad" appears on-screen.
- Draw your signature on the signature pad using the provided stylus.

To start over, tap **Clear**.

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

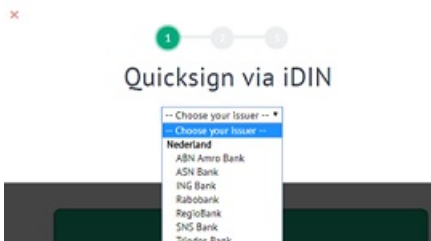


- When you're done, tap **OK** on the signature pad screen using the stylus.
- The document is now being signed.
- If asynchronous signing is supported, you may now click **Close** to close the signing session and continue working on other things.



### ***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.

Make sure the **Connective Browser Package** (for Windows and macOS) been properly installed. Otherwise BeLawyer signing will not work.

- 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.
- Itsme cannot be combined with QuickSigning: each document within a package must be signed individually.

#### 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. *Note that the certificate creation steps don't apply when using itsme through OpenID Connect.*
- Click **Create my certificate**. A message is now sent to your itsme app.



en ▼

### Create your certificate

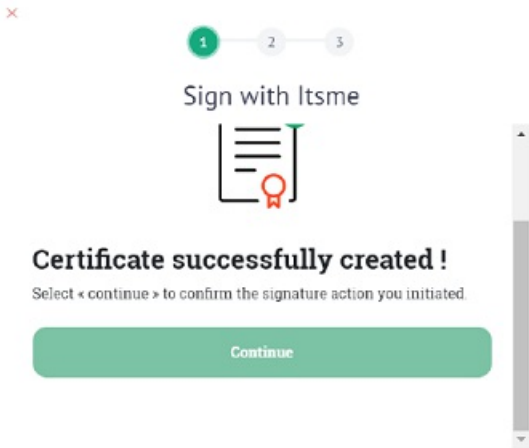
To enable the electronic signature with itsme, a signature certificate linked to your identity needs to be created.

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

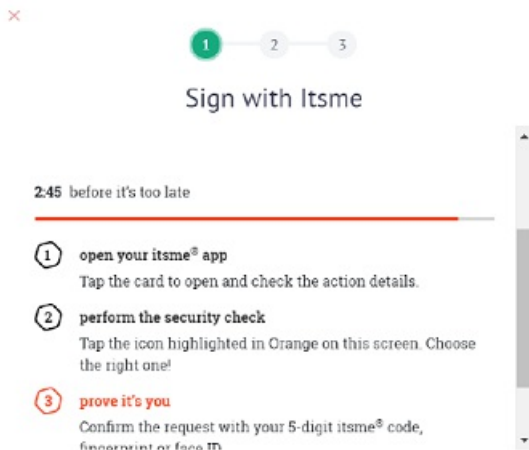
Create my certificate



- 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.



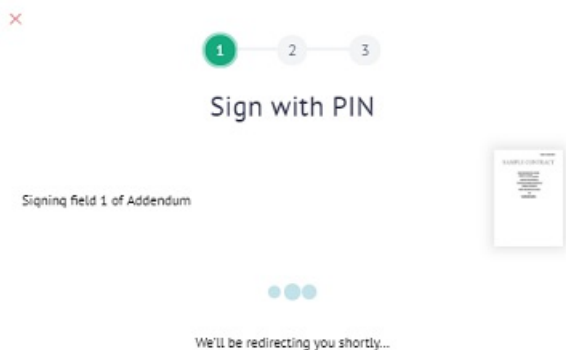
- Your document will now be signed with itsme.

### ***Sign with custom signing type***

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.



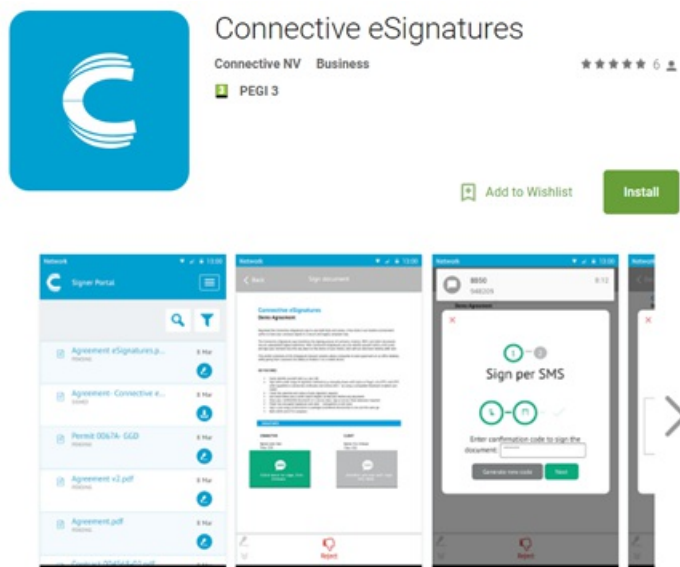
Cancel

- When you are done, close the browser tab.

## 5.2 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](#).

## 5.2.1 How do I install the app?

### Minimum System Requirements

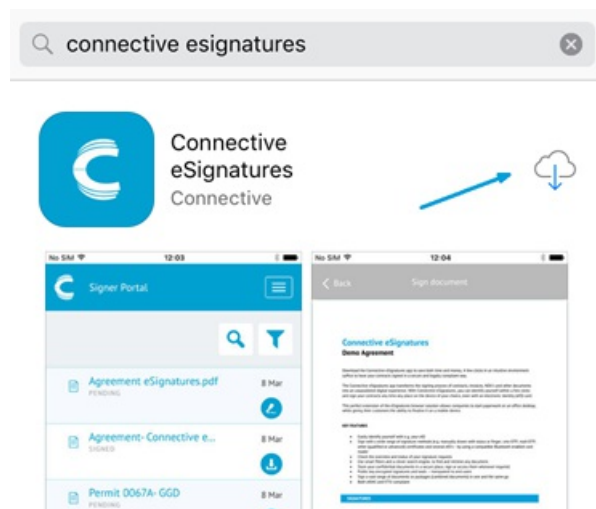
- Android 4.3 and up.
- iOS 9.3 or later. Compatible with iPhone, iPad and iPod Touch.

### 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.



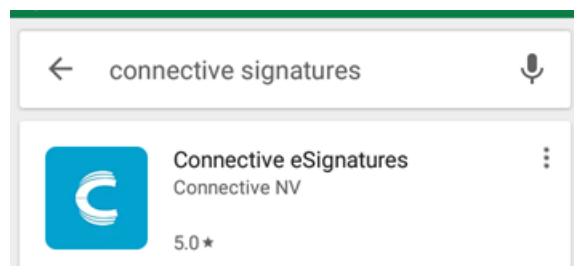
- When the download is complete, the **Open** button is displayed.
- Tap **Open** to open the app.



- 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.

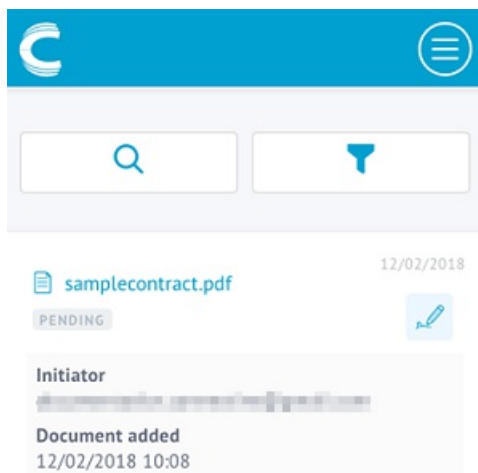
## 5.2.2 How do I sign in the Connective app?

### 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**. **Note:** this checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.
- The same signing steps apply like when signing a document on a computer. See [Signing a document step-by-step](#) for more info.

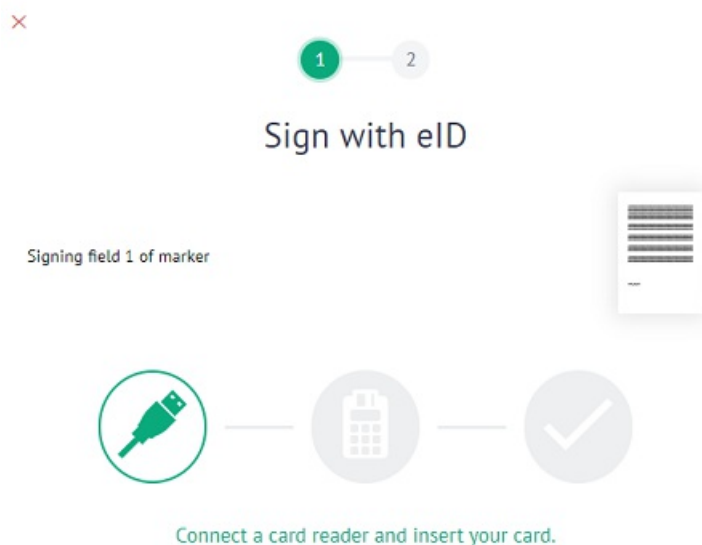
Note about signing with Belgian eID

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***

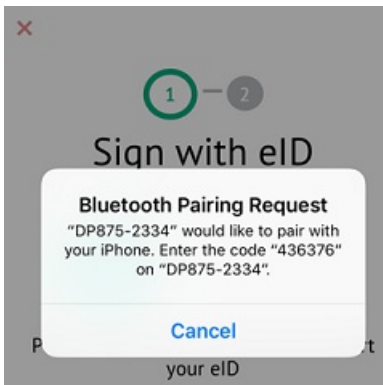
Before you can sign using such a card reader you need to pair it to your mobile device. This is a one-time action.

- 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**.



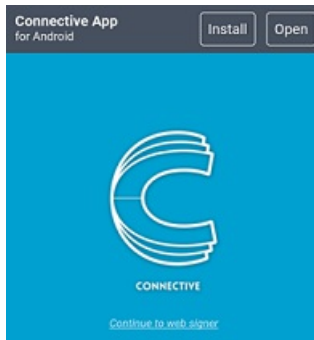
- When the pairing is complete, you're prompted to enter the PIN code of your eID. Enter the PIN and press **OK**. Your document is now being signed. This may take some time, since all data is sent over Bluetooth.

### 5.2.3 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.



## 5.3 Signers FAQ

In this section we address the questions that signers might have concerning eSignatures.

The questions are bundled per topic where possible.

### 5.3.1 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?](#)

### 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 SignID software, see the [SignID section](#) on the documentation website.

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 SignID software, see the [SignID section](#) on the documentation website.

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.

## 2. Which operating systems are supported?

The following operating systems are supported:

### **Microsoft Windows**

- Microsoft Windows 10 (64-bit)
- Microsoft Windows 8.1 (64-bit)

### **Mac OS**

- macOS Big Sur
- macOS Catalina

### **Android**

- Android 11
- Android 10

### **iOS**

- iOS 14
- iOS 13

Other operating systems are officially *not* supported.



## Which web browsers are supported?

The following web browsers are supported:

### Windows

- Microsoft Edge n-2
- Internet Explorer 11

Earlier versions of Internet Explorer are not supported.

- 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.

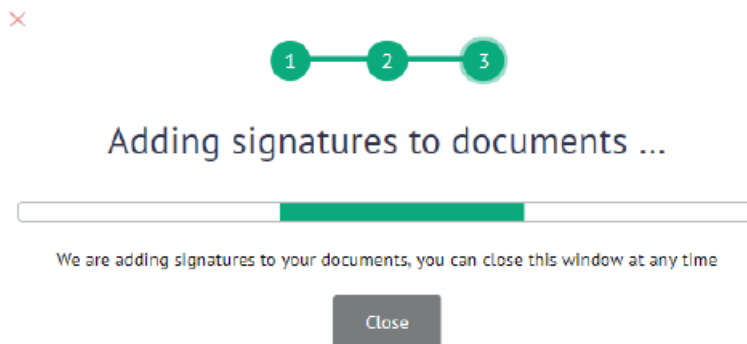
---

## How does asynchronous signing work?

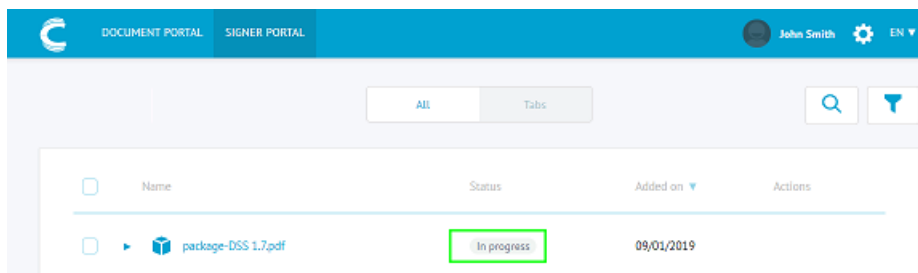
Asynchronous signing allows you to close the signing session and let the signing run in the background. This way, you can work on other things while your documents are being signed. This comes in handy especially when signing large documents that take quite some time to be signed.

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**.  
**Note:** this checkbox may vary, based on the configuration of your environment. The administrator may choose to enable or disable the Terms of Use, the Privacy Policy and the Cookie Policy separately.
- 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.



- 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.

## 5.4 Signers Troubleshooting

In this section you'll find the issues that signers might have concerning eSignatures.

The issues are bundled per topic where possible.

## Emails

I don't receive any emails with documents to sign

I replied to an email sent by eSignatures but never received a response

When trying to download my signed documents via email, I keep getting the message "Your url has expired, please request a new email"

## I don't receive any emails with documents to approve/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 <a href="mailto:esigner@connective.eu">esigner@connective.eu</a>
The eSignatures solution doesn't have your correct email address.	Contact your initiator and communicate your correct email address.
The initiator may have modified your email address in the contact's info, but documents that have already been sent to your previous email address are not updated retroactively.	Contact your initiator and communicate your correct email address.
The approver/signer/receiver is using an external SPAM filter that is blocking the incoming mail.	Check with the approver/signer/receiver which external SPAM filter they are using. Depending on the configuration, the mail address from which eSignatures is sending its emails may need to be added to the Safe Sender list.

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.

When trying to download my signed documents via email, I keep getting the message "Your url has expired, please request a new email"

Even when I click **Request a new mail** and use the new download link, I still keep getting this message.

POSSIBLE CAUSE	SOLUTION
The one-time URLs are being read by your organization's security system, and are therefore already used.	<p>Contact your IT administrator and ask them to take the following actions:</p> <ul style="list-style-type: none"><li>- Add connective.eu to the trusted domains.</li><li>- Whitelist the download notifications.</li><li>- Make sure the mail recipients add the URL to the Microsoft Office 365 Advanced Threat Protection (ATP) <b>Do not rewrite the following URLs</b> list.</li></ul> <p>If the solution above is not an option, the eSignatures admin can create a Signers only user group and add the end users who are experiencing this issue to it. That way, they can access the Signer Portal and sign/download their documents directly in the portal. How to create a Signers only user group is explained <a href="#">here</a>.</p>

### 3.4.2 Signing documents

I get an empty screen when trying to sign

I keep getting the message "Your url has expired, please request a new email"

I'm not mandated to sign

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

The Connective app doesn't open



I get an empty screen when trying to sign

POSSIBLE CAUSE	SOLUTION
You're not using an officially supported web browser.	Use an <a href="#">officially supported web browser</a> .

I keep getting the message "Your url has expired, please request a new email"

Even when I click **Request a new mail** and use the new signing link, I still keep getting this message.

POSSIBLE CAUSE	SOLUTION
The one-time URLs are being read by your organization's security system, and are therefore already used.	Contact your IT administrator and ask them to add connective.eu to the trusted domains.

I'm not mandated to sign

When trying to sign with eID, BeLawyer, iDIN 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, or returned by iDIN.	<p>Make sure the info in your contact matches the certificate info on the eID card, BeLawyer card, iDIN or itsme account.</p> <p>When the system admin applied mandated signing rules in the Config Index to check the names and birthdate, you need to enter the given names and birthdate in the contact info.</p> <p>If the info still doesn't match, contact your administrator to reduce the <b>MatchLevel</b> in the Config Index.</p>

I'm unable to sign with itsme

POSSIBLE CAUSE	SOLUTION
You're using an unsupported Operating System: macOS Mojave v10.14.	Use a <a href="#">supported Operating System</a> .
You're using an unsupported browser: Safari v12.0.	Use a <a href="#">supported browser</a> .

I'm not able to sign a PDF/A document

In the rare case this happens, contact your administrator and ask them to create a ticket on our support website.

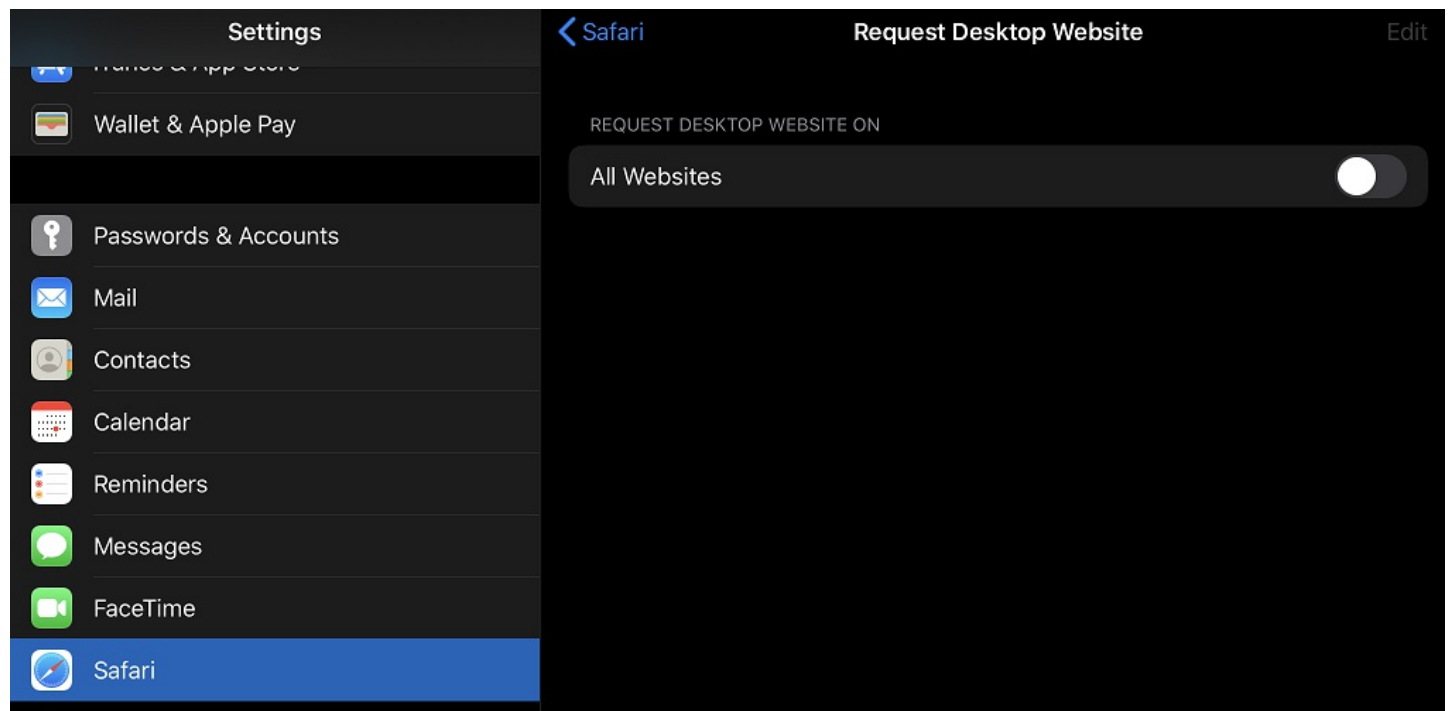
The legal notice is not displayed correctly on my biometric signature pad

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.

## The Connective app doesn't open

When using eSignatures on an iPad, the document opens directly in my mobile browser instead of in the Connective app.

POSSIBLE CAUSE	SOLUTION
You're using Safari on an iPad with iOS 13 and Safari is configured to open in Desktop mode.	Disable the setting to request desktop website. <b>To disable this setting:</b> <ul style="list-style-type: none"><li>-Open <b>Settings</b>.</li><li>-Scroll down to <b>Safari</b>.</li><li>-Disable <b>Request desktop website on</b> for all websites.</li></ul>



# Preface

The entire look and feel of an eSignatures environment can be rebranded to suit clients' requirements. This is done by creating themes in the **Theme** section of the Configuration Index.

This document describes the following:

1. [Access the Theme settings](#)
2. [Create and customize themes](#)
3. [Brandable components](#)
4. [Apply themes to an eSignatures environment](#)

**Important:** if you already used a custom theme of the WYSIWYS in a previous version of eSignatures, that custom theme will also be used when upgrading to eSignatures 6.3. How this works is explained in [1. Access the Theme settings](#).



# Preface

The entire look and feel of an eSignatures environment can be rebranded to suit clients' requirements. This is done by creating themes in the **Theme** section of the Configuration Index.

This document describes the following:

1. [Access the Theme settings](#)
2. [Create and customize themes](#)
3. [Brandable components](#)
4. [Apply themes to an eSignatures environment](#)

**Important:** if you already used a custom theme of the WYSIWYS in a previous version of eSignatures, that custom theme will also be used when upgrading to eSignatures 6.3. How this works is explained in [1. Access the Theme settings](#).

# Revisions

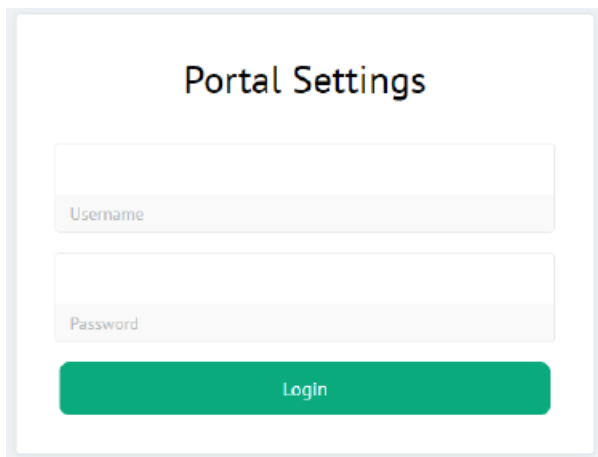
DATE	VERSION	OWNER	TOPIC
2019-08-19	1.0	DGI	Document Creation
2019-09-17	1.1	DGI	Added .ico as supported Favicon file format

# 1. Access the Theme settings

**Prerequisite:** you need to be part of a user group that has the required permissions to access the **Theme** settings of the eSignatures Config Index. If you can't log in to the Config Index, contact your administrator or Connective.

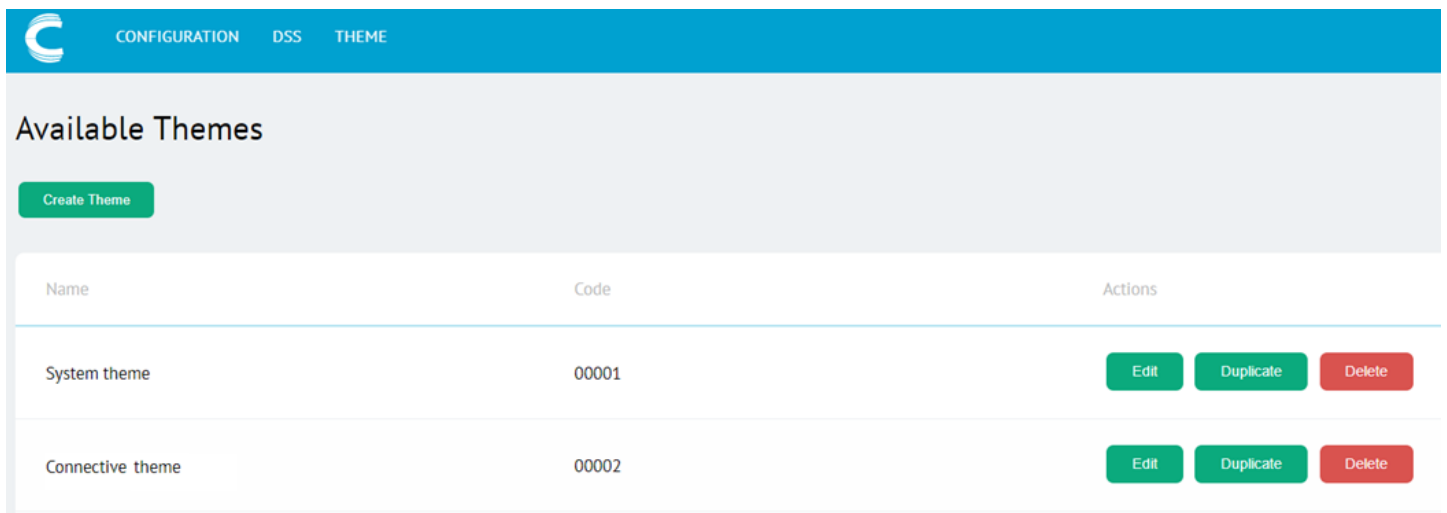
- Go to [https://\[server\]/\[application\]/esig/config/index.html](https://[server]/[application]/esig/config/index.html)
- Enter your **Username** and **Password**.
- Click **Login**.

**Note:** With the introduction of our SSO authentication service, you won't need to enter credentials if you are already logged in to another eSignatures component.



The screenshot shows a login form titled "Portal Settings". It contains two input fields: "Username" and "Password". Below these fields is a green "Login" button.

- Click the **Theme** tab. Note that the **Configuration** and **DSS** tabs may not be accessible, depending on your permissions.



The screenshot shows the "Available Themes" page. At the top, there is a navigation bar with tabs: "CONFIGURATION", "DSS", and "THEME". Below the navigation bar, there is a "Create Theme" button. The main content area is a table with the following data:

Name	Code	Actions
System theme	00001	<button>Edit</button> <button>Duplicate</button> <button>Delete</button>
Connective theme	00002	<button>Edit</button> <button>Duplicate</button> <button>Delete</button>

Two themes are by default available: System theme and Connective theme.

## Connective theme

This is the default Connective theme and uses the standard Connective colours. The default colour code of each component is mentioned in this documentation.

Note that the Connective theme cannot be edited or deleted.

## System theme

If you were already using a custom theme in a previous eSignatures version and you upgraded to eSignatures 5.4, the colours of

the existing theme will be applied to the System theme.

Note however, that in previous versions of eSignatures it was only supported to customize the WYSIWYS. Since version 5.4, you can customize nearly every element of the eSignatures components: Web Portal, Redirect page, Top bar, etc. To facilitate the administrator's job, the system tries to apply the colours of the existing theme to as many configurable elements of the System theme as possible, but it's up to the administrator to check if the migration happened correctly.

If you weren't using a custom theme in the previous version of eSignatures, the System theme will be identical to the Connective theme, with the difference that you're able to edit it.

When creating a new theme, the new theme will be based on the System theme.

When using the **Restore to defaults** buttons they will restore the colours to the ones of your current System theme.

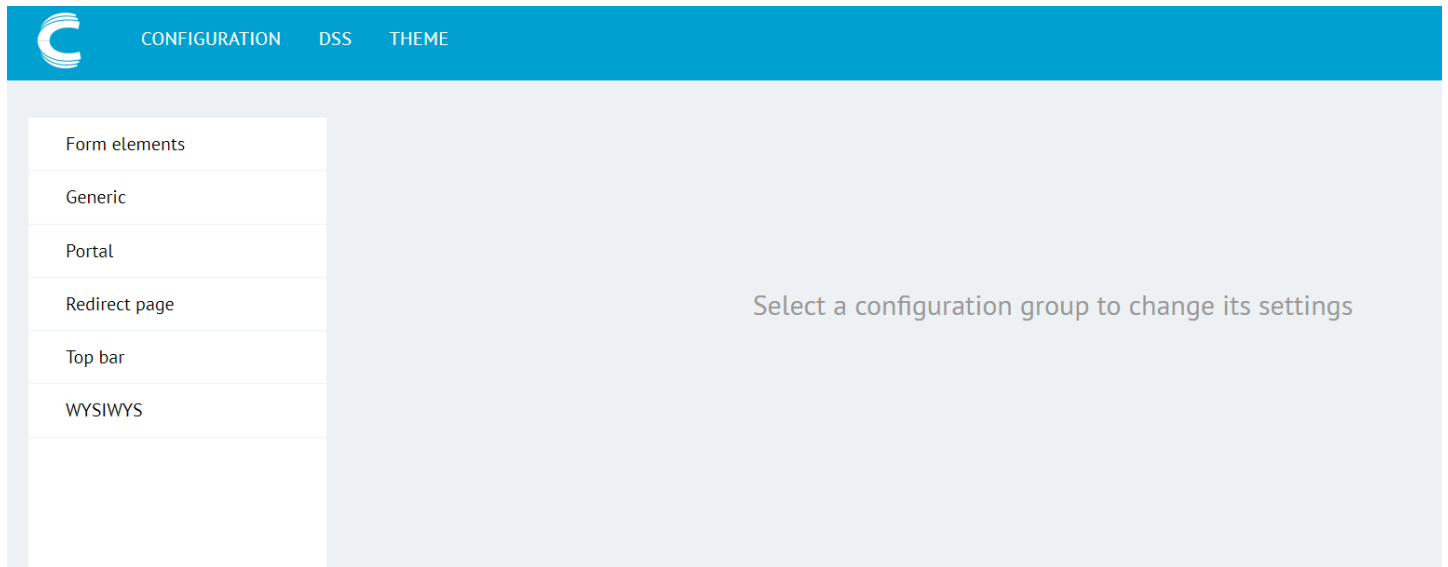
## 2. Create and customize themes

### 2.1 Create a theme

- Click the **Create Theme** button. Any new theme you create is based on the System theme.

**Tip:** to use a different theme as base, duplicate the required theme and edit it, instead of creating a new one.

- The Theme settings are split into different configuration groups, based on the brandable components of eSignatures. The different configuration groups are listed alphabetically and are described in the section [3. Brandable components](#).



#### 2.1.1 Size Limitations

Consider the following size limitations when creating themes:

- A theme must not exceed 10 MB in total.
- A background image file within a theme must not exceed 2 MB.
- A logo file within a theme must not exceed 0.5 MB.
- The number of themes per client must not exceed 10.

### 2.2 Edit a theme

- Click the **Edit** button next to an existing theme. Note that the Connective theme cannot be edited.
- Modify the required components and save your settings.
- To preview what the changes look like, click the **Preview** button at the bottom of the screen.

**Foreground**Reset to defaults

Selected day

Mouse over

Preview

- If the results aren't what you expected, click the **Restore to defaults** buttons next to the components you want to restore.

The colours of the corresponding component will be restored to the default colors of the current System theme.

Form elements

FieldsButtonsColoured links

Fields

Border

Enabled

Unselected

Reset to defaults

Reset to defaults

Reset to defaults

**Note:** in eSignatures 5.5 you can apply a theme on package level (See section [4. Apply themes to an eSignatures environment](#) for more information.)

2.3 Duplicate a theme

- Click the **Duplicate** button next to an existing theme. Duplicating a theme allows you to build a new theme based on an existing one, instead of on the System theme.
- Modify the required components and save your settings.

You have made changes to this page. Don't forget to save them!

CancelSave

2.4 Search for a theme

- Enter the theme name in the **Search** box and press Enter.

2.5 Delete a theme

- Click the **Delete** button next to the theme you want to delete. Note that the Connective theme and System theme cannot be deleted.

### 3. Brandable components

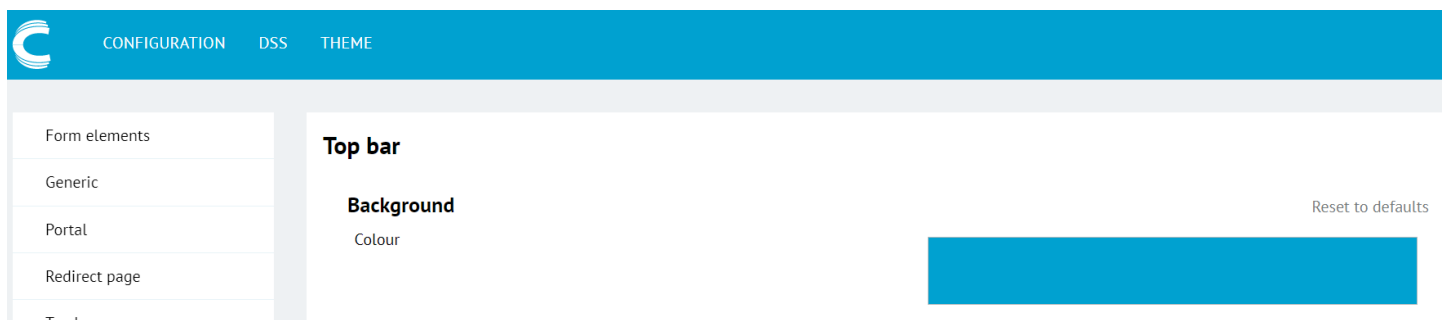
The following components can be rebranded:

- [Generic settings](#)
- [Top bar](#)
- [Form elements](#)
- [WYSIWYS](#)
- [Portal](#)
- [Redirect page](#)

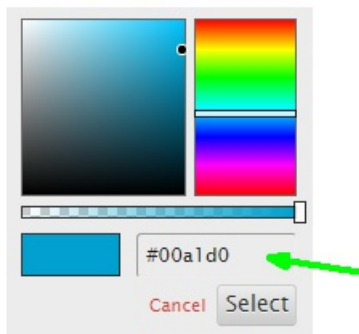
#### Modify the colours of a component

The colour of the different components can be modified to suit clients' requirements. The procedure to modify the colour is identical for every component.

- Click inside the colour field of the component you want to customize.



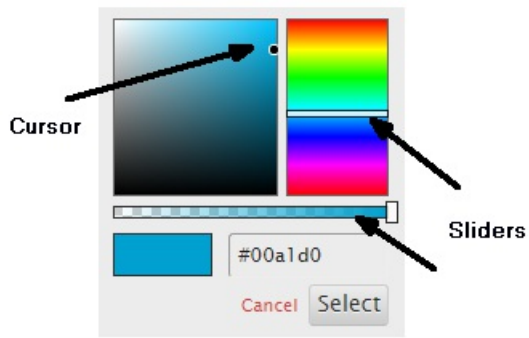
- If you know the specific colour code you want to apply, enter the colour code in hexadecimal format, or in RGBA format if transparency is needed.



- Then click Select.

If you don't know the hex colour code:

- Move the slider of the vertical colour bar to the required colour.
- Move the cursor to the required shade.
- Move the slider of the horizontal colour bar to the required transparency.



- Then click **Select**.

**Important:** after customizing the colours, always check the readability of the text in the rebranded screens. If the customization doesn't fit your needs, click the **Reset to defaults** buttons. The buttons reset the colours to the ones of the current System theme.



## 3.1 Generic

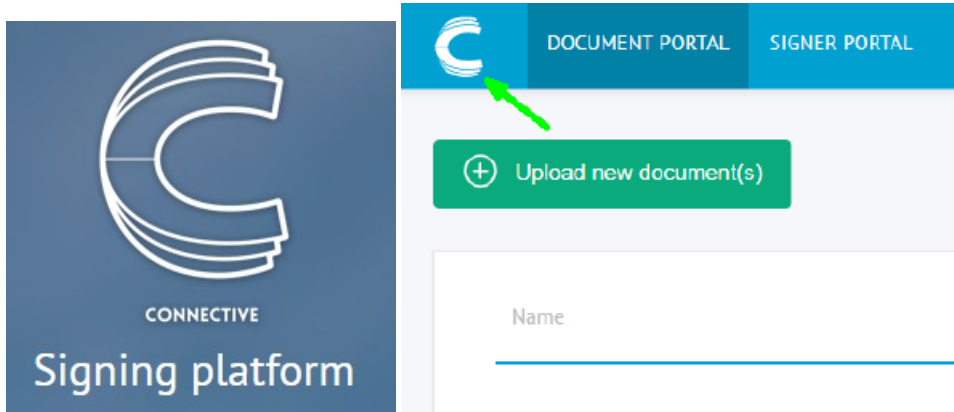
In the **Generic** settings you can customize the application logo, modify the colours of the document separator, and assign a theme name.

### 3.1.1 Settings

#### 3.1.1.1. Application Logo

This setting customizes the application logo. The application logo is displayed in the following locations:

- On the login page of the WebPortal.
- In the top left corner of the top bar when a user has logged in.
- In the center of the Redirect page.



To use a customized logo:

- Select the **Logo type**:
  - Svg
  - Image

#### **Svg**

It's highly recommended to use **svg** as logo type. Svg stands for Scalable Vector Graphics and is an XML-based vector image format used for two-dimensional graphics. Since svg files are scalable, you won't have resolution issues in the interface.

#### **Important notes:**

- To make sure the correct brand colour is used in the background, clients must make sure the "fill" tag contains the required value in the svg file.
- The svg file must not contain an "image" tag. If the svg contains an image tag, it acts like a regular image, which isn't the goal.

#### **Image**

If it's not possible to use svg, use image.

The following image files are supported: .png, .jpeg and .gif.

When image is used, it's not possible to have a different background and foreground. The flat image will be inserted in full in the interface.

- Click the **Choose File** button.

**Important:** the maximum supported file size is 0.5 MB.

- The image file you've selected is shown inside the frame.

- To preview what the modifications will look like, click **Preview** at the bottom of the page.
- Click **Save** to save the settings.

#### 3.1.1.2 Document separator

The Document separator indicates where one document ends inside a package, and where another begins.

##### **Background**

Background colour of the separator.

Default hex colour code: #00a1d0

##### **Foreground**

Foreground (text) colour of the separator.

Default hex colour code: #ffffff

#### 3.1.2. Theme name

Enter a Theme name for your customized theme.

**Note:** to apply a theme, you need to select the theme name in the Configuration Index > Customization Settings > Theme.

#### 3.1.3. Code

- The theme **Code** is assigned automatically.
- The value of the Code can be used as Request parameter in the eSignatures API to apply the desired theme to a package.

## 3.2 Top bar

These settings customize the Top bar of the eSignatures Web Portal.

**Important:** after customizing the colors, always check the readability of the text in the rebranded screens. If the customization doesn't fit your needs, click the Reset to defaults buttons.



### 3.2.1. Background

#### 3.2.1.1. color

Background color of the Top bar.

Default hex color code: #00a1d0



**Background**

### 3.2.2. Button

The Top bar consists of the following buttons:



**Buttons**

#### 3.2.2.1. Background

Background color of the Top bar buttons.

##### Unselected

Background color when a top bar button is unselected. E.g. Signer Portal in the image above.

Default hex color code: #00a1d0

##### Selected

Background color when a top bar button is selected. E.g. Document Portal in the image above.

Default hex color code: #0181a6

#### 3.2.2.2. Foreground

Foreground (text) color of the Top bar buttons.

##### Unselected

Foreground color when a top bar button is unselected.

Default hex color code: #ffffff

##### Selected

Foreground color when a top bar button is selected.

Default hex color code: #ffffff

### 3.2.3. Drop-down

This setting customizes the colors of the drop-down menus of the top bar.

#### 3.2.3.1. Background

##### color

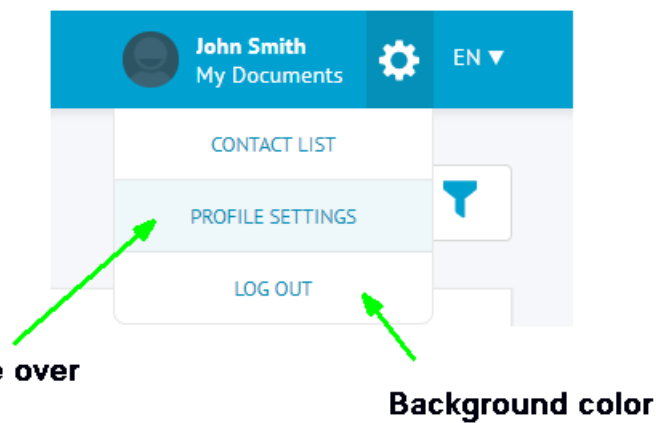
Background color of the drop-down menu background. E.g. Contact list in the image below.

Default hex color code: #ffffff

##### Mouse over

Background color of when a drop-down menu item is moused over. E.g. Profile settings in the image below.

Default hex color code: #f0fcff



#### 3.2.3.2. Foreground

##### color

Foreground (text) color of the drop-down menu background.

Default hex color code: #0490b9

##### Mouse over

Foreground (text) color of when a drop-down menu item is moused over.

Default hex color code: #0490b9

#### 3.2.3.3. Border

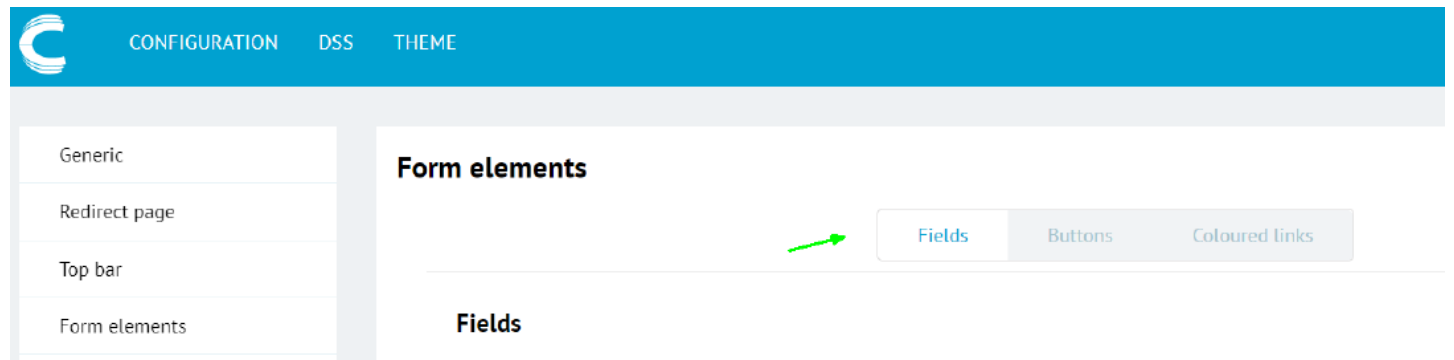
Border within the drop-down menu.

Default color code: rgba(207, 207, 130, 0.81)

## 3.3 Form elements

These settings customize the different form elements that are used in an eSignatures environment. Form elements are **fields**, **buttons** and **coloured** links.

- Click the tabs to go to the required form element configuration.



Every form element consists of multiple components. A button, for instance, has a foreground and a background. A field has a border, a font, possibly a label, a drop-down menu, etc. All these elements can be customized and may vary based on whether the element is enabled, disabled, selected, unselected, etc.

### 3.3.1 Fields

Fields are used to enter contact and user data, document names, legal notices, and so on.

Checkboxes, toggle buttons and date pickers are also considered fields.

#### Single line text field

Placeholder text	Placeholder text
Instructions text	Error text

#### 3.3.1.1 Border

This is the border around each field in eSignatures. When you change this colour, the border colour of every field will be modified.



Placeholder text

Instructions text

#### Enabled

Colour of the border when the field is enabled.

##### *Unselected*

Colour of the border when the field is enabled but unselected.

Default hex colour code: #efebeb

##### *Selected*

Colour of the border when the field is enabled and selected.

Default hex colour code: #efebeb

#### Disabled

Colour of the border when the field is disabled.

Default hex colour code: #efebeb

#### 3.3.1.2 Font

Font colour of the input text or placeholder text inside a field.



Placeholder text

Instructions text

#### Text

##### *Input*

Font colour of the input text inside a field. The input text is the text a user enters.

Default hex colour code: #0baa7d

### *Placeholder*

Font colour of the placeholder text inside a field. The placeholder text is displayed when the user has not entered any text.

Default hex colour code: #b3b3b3

Placeholder text	User's input text
------------------	-------------------

#### 3.3.1.3 Label

The label is the part beneath a field.

John

Enter your first name here.

### **Instruction text**

The instruction text in a label describes which data must be entered in a field.

#### *Foreground*

Foreground (text) colour of the instruction text.

#### *Unselected*

Foreground colour of the text when the field is unselected.

Default hex colour code: #f9f9f9

#### *Selected*

Foreground colour of the text when the field is selected.

Default hex colour code: #ececec

#### *Background*

Background colour of the label when it contains instruction text.

#### *Unselected*

Background colour of the label when the field is unselected.

Default hex colour code: #b9bbbc

#### *Selected*

Background colour of the label when the field is selected.

Default hex colour code: #8c8c8c

### **Error text**

The error text in a label indicates when incompatible or no data has been entered in a field.

Email

This field is required.

### *Foreground*

Foreground (text) colour of the error text.

### *Unselected*

Foreground colour of the text when the field is unselected.

Default hex colour code: #9e1f1c

### *Selected*

Foreground colour of the text when the field is selected.

Default hex colour code: #9e1f1c

### *Background*

Background colour of the label when it contains error text.

### *Unselected*

Background colour of the label when the field is unselected.

Default hex colour code: #f9e4e4

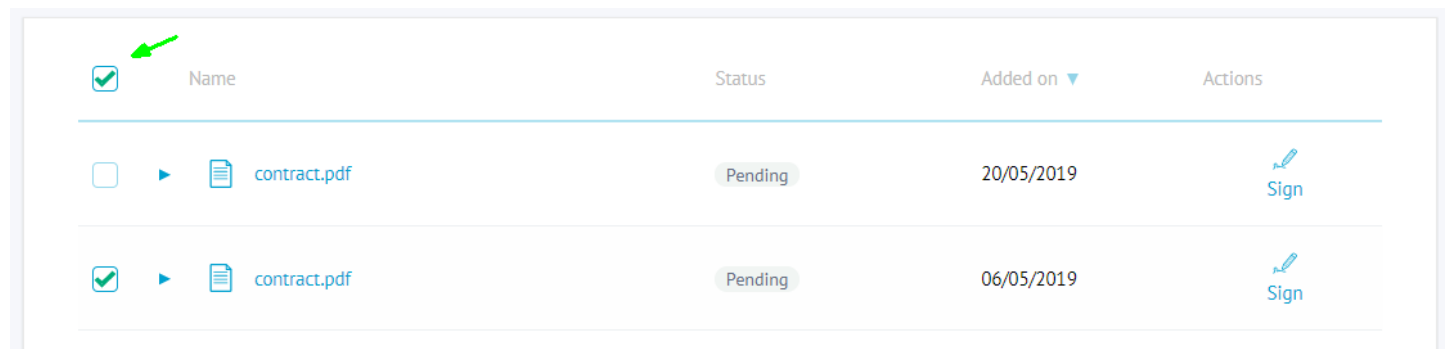
### *Selected*





Background colour of the label when the field is selected.

Default hex colour code: #f7c6c6

## 3.3.1.4 Checkbox

Checkboxes in eSignatures allow users to select and unselect documents in the Signer Portal.



<input checked="" type="checkbox"/>	Name	Status	Added on ▼	Actions
<input type="checkbox"/>	 contract.pdf	Pending	20/05/2019	 Sign
<input checked="" type="checkbox"/>	 contract.pdf	Pending	06/05/2019	 Sign

### **Border**

This is the border around the checkbox.

### *Enabled*

Colour of the checkbox border when the checkbox is enabled (i.e. selectable).

### *Unselected*

Colour of the border when the checkbox is enabled but unselected.

Default hex colour code: #00a1d0

### *Selected*



Colour of the border when the checkbox is enabled and selected.

Default hex colour code: #00a1d0

#### *Disabled*

Colour of the border when the checkbox is disabled (i.e. unselectable).

Default hex colour code: #00a1d0

### **Checkmark**

Colour of the actual checkmark inside the checkbox.



Default hex colour code: #00a1d0

### 3.3.1.5 Toggle

Toggle buttons allow users to change a setting between two states.

Do you want to add a legal notice?



### **Background**

Background colour of the toggle button.



#### *On colour*

Background colour when the toggle button is on.

Default hex colour code: #0baa7d

#### *Off colour*

Background colour when the toggle button is off.

Default hex colour code: #ffffff

#### *Disabled colour*

Background colour when the toggle button is disabled, i.e. unselectable.

Default hex colour code: #0baa7d

### **Foreground**

Foreground colour of the toggle button.



#### *On colour*

Foreground colour when the toggle button is on.

Default hex colour code: #ffffff

*Off colour*

Foreground colour when the toggle button is off.

Default hex colour code: #0baa7d

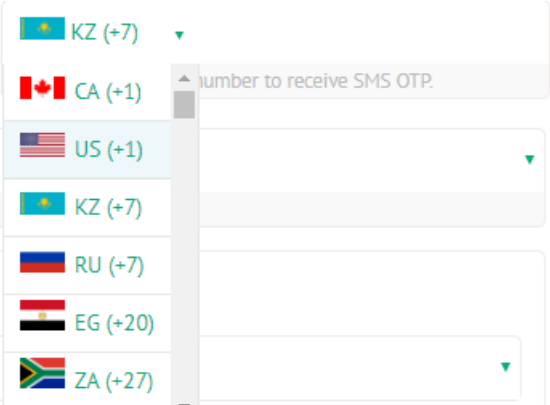
*Disabled colour*

Background colour when the toggle button is disabled, i.e. unselectable.

Default hex colour code: #000000

3.3.1.6 Extended drop-down

Extended drop-down menus are basically very long drop-down lists. Such as the supported SMS OTP countries for instance. Their background colour can be customized.



**Background**

Background colour of the extended drop-down menus.

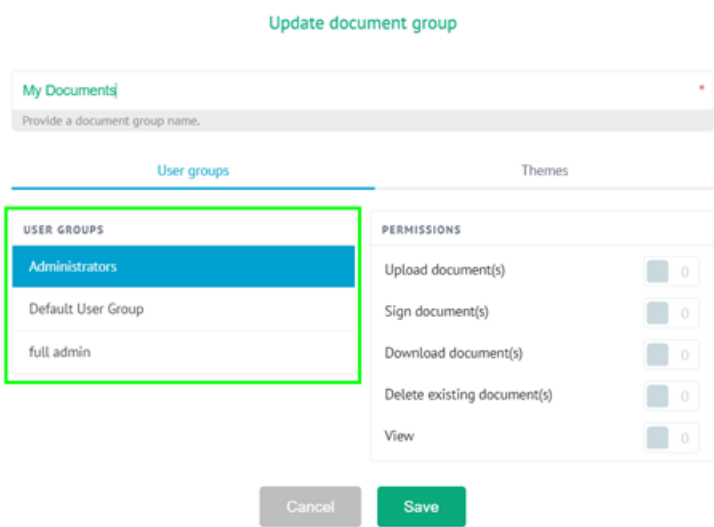
*Mouse over*

Colour of the extended drop-down menu when the item is being moused over.

Default hex colour code: #7fb4c3

3.3.1.7 List detail panel

The list detail panel is displayed when editing a Document group in the **Access Management** section of eSignatures.



Background

Background colour of the list detail panel.

Selected

Background colour when an element from the list detail panel is selected.

Colour

Colour when the element is selected.

Default hex colour code: #00a1d0

Mouse over Colour when the element is moused over.

Default hex colour code: #00a1d0

Unselected Background colour when an element from the list detail panel is unselected.

Colour Colour when the element is unselected.

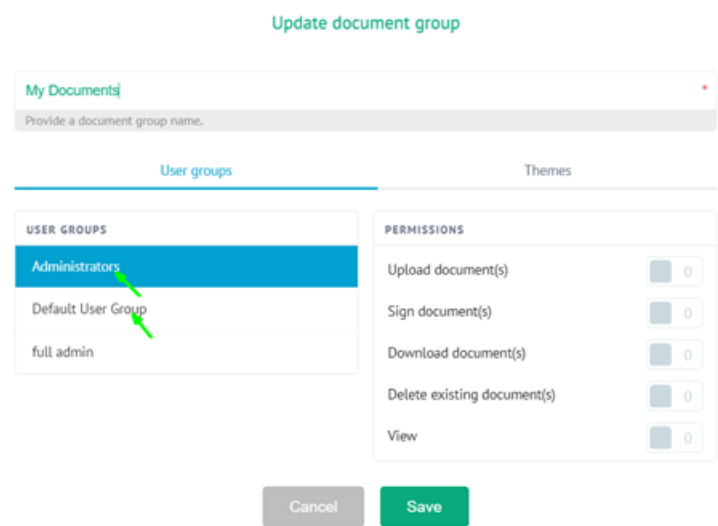
Default hex colour code: #ffffff

Mouse over Colour when the element is moused over.

Default hex colour code: #f5f5f5

Foreground

Foreground colour of the list detail panel.



Selected

Foreground colour when an element from the list detail panel is selected.

Colour

Colour when the element is selected.

Default hex colour code: #ffffff

Mouse over

Colour when the element is moused over.

Default hex colour code: #ffffff

*Unselected*

Foreground colour when an element from the list detail panel is unselected.

*Colour*

Colour when the element is unselected.

Default hex colour code: #2e2e2e

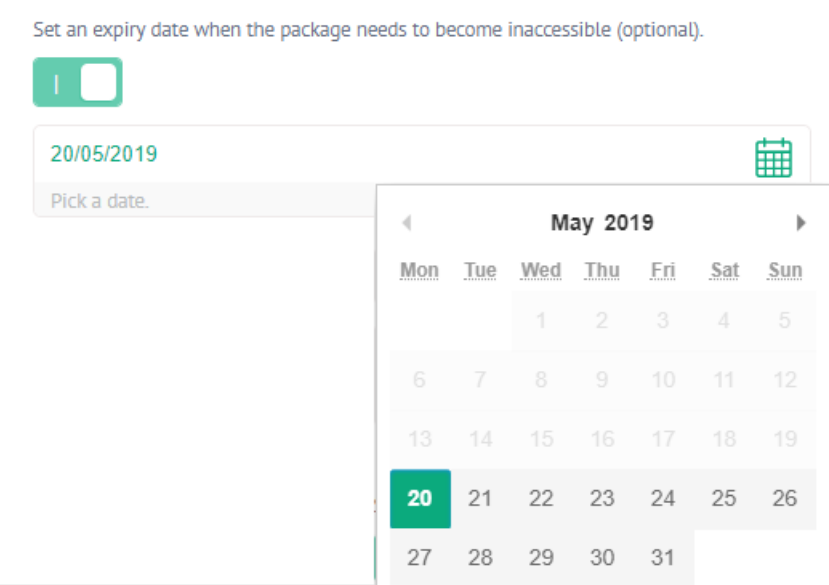
*Mouse over*

Colour when the element is moused over.

Default hex colour code: #2e2e2e

3.3.1.8 Mini calendar

The mini calendar is used to set an expiration date when a package should become inaccessible.



**Today**

Colour in which the current day is highlighted.

Default hex colour code: #0baa7d

**Background**

Background colour of the calendar.

*Selected day*

Background colour when a day is selected.

Default hex colour code: #0baa7d

*Mouse over*

Background colour when a day is moused over.

Default hex colour code: #64ccaf

**Note:** dates in the past, which can't be selected, cannot be customized.

## **Foreground**

Foreground colour of the calendar.

*Selected day*

Foreground colour when a day is selected.

Default hex colour code: #ffffff

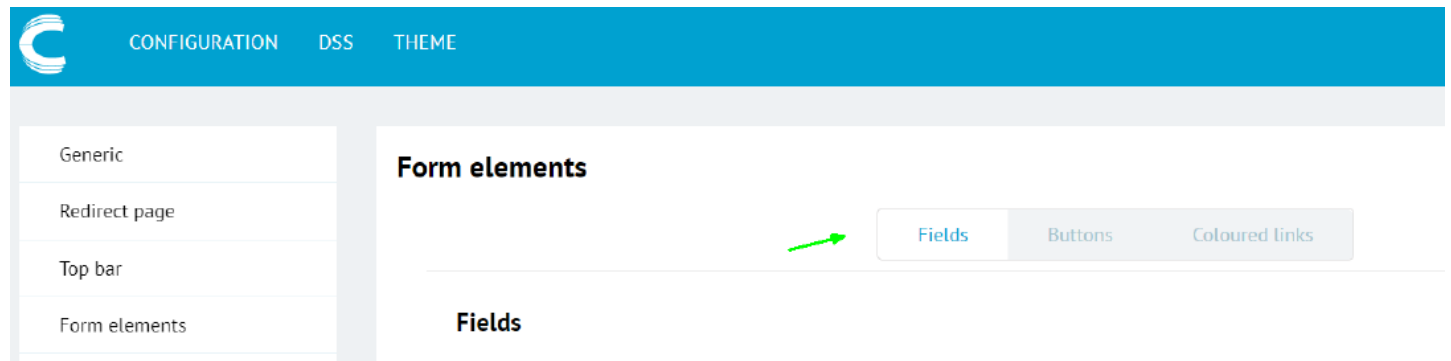
*Mouse over*

Foreground colour when a day is moused over.

Default hex colour code: #ffffff

### 3.3.2 Buttons

Click the **Buttons** tab to access the button settings.

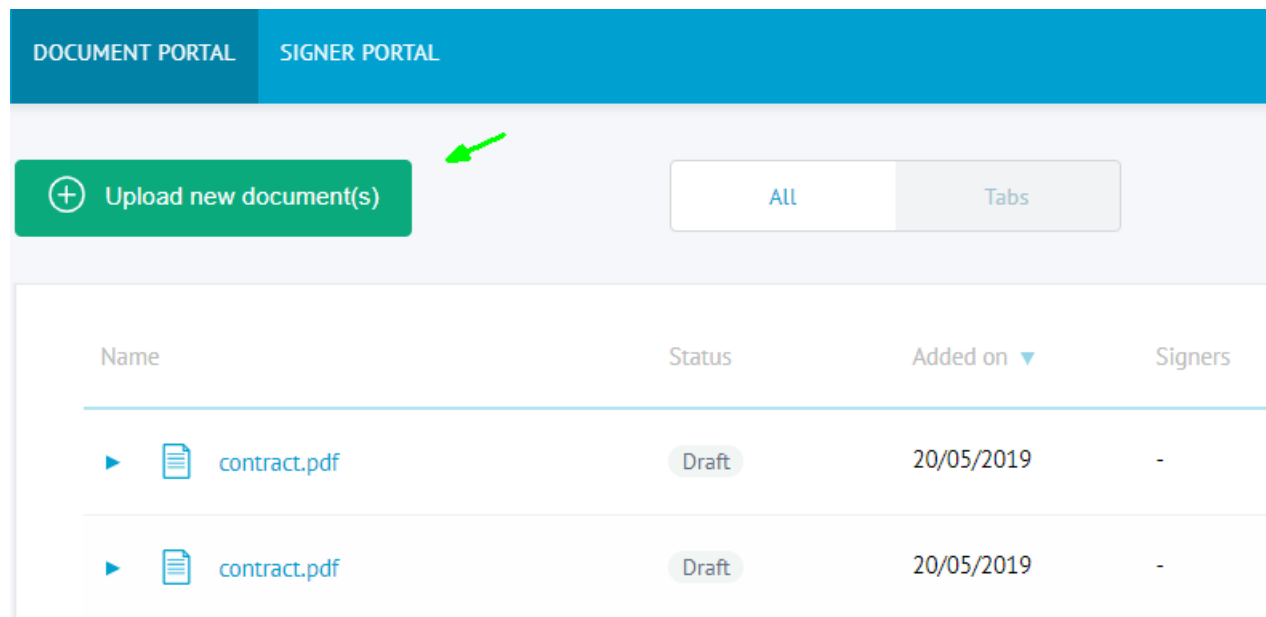


Buttons are divided into the following categories:

- Positive
- Negative
- Neutral
- Neutral alternative
- Cloud
- Small
- Next
- Previous

#### 3.3.2.1 Positive

Positive buttons execute positive actions, like upload a document, select a file, add a signer, etc.



#### Background

Background colour of a positive button.

*Enabled*

Background colour when a positive button is enabled.

Default hex colour code: #0baa7d

*Disabled*

Background colour when a positive button is disabled.

Default hex colour code: #64ccaf

*Mouse over*

Background colour when a positive button is moused over.

Default hex colour code: #033225

**Foreground**

Foreground (text) colour of a positive button.

*Enabled*

Foreground (text) colour when a positive button is enabled.

Default hex colour code: #ffffff

*Disabled*

Foreground (text) colour when a positive button is disabled.

Default hex colour code: #ffffff

*Mouse over*

Foreground (text) colour when a positive button is moused over.

Default hex colour code: #ffffff

**Border**

Border colour of a positive button.

*Enabled*

Border colour when a positive button is enabled.

Default hex colour code: #0baa7d

*Disabled*

Border colour when a positive button is disabled.

Default hex colour code: #64ccaf

*Mouse over*

Border colour when a positive button is moused over.

Default hex colour code: #033225

**3.3.2.2. Negative**

Negative buttons execute negative actions, such as rejecting a document.



## Reject to sign the package.

Reason for rejection:

Reject|

\* The reject reason is required

494

Close

Reject

### Background

Background of a negative button.

#### *Enabled*

Background colour when a negative button is enabled.

Default hex colour code: #d9534f

#### *Disabled*

Background colour when a negative button is disabled.

Default hex colour code: #e68f8d

#### *Mouse over*

Background colour when a negative button is moused over.

Default hex colour code: #8b211e

### Foreground

Foreground (text) of a negative button.

#### *Enabled*

Foreground (text) colour when a negative button is enabled.

Default hex colour code: #ffffff

#### *Disabled*

Foreground (text) colour when a negative button is disabled.

Default hex colour code: #ffffff

#### *Mouse over*

Foreground (text) colour when a negative button is moused over.

Default hex colour code: #ffffff

### Border

Border colour of a negative button.

#### *Enabled*



Background colour when a negative button is enabled.

Default hex colour code: #d9534f

#### *Disabled*

Border colour when a negative button is disabled.

Default hex colour code: #e68f8d

#### *Mouse over*

Border colour when a negative button is moused over.

Default hex colour code: #8b211e

### 3.3.2.3. Neutral

Neutral buttons execute neutral actions, such as closing a window.

#### **Background**

Background of a neutral button.

#### *Enabled*

Background colour when a neutral button is enabled.

Colour code: `rgba(187, 188, 245, 0.73)`

#### *Disabled*

Background colour when a neutral button is disabled.

Default hex colour code: #a3a6a7

#### *Mouse over*

Background colour when a neutral button is moused over.

Default hex colour code: #d0d1d2

#### **Foreground**

Foreground (text) of a neutral button.

#### *Enabled*

Foreground (text) colour when a neutral button is enabled.

Default hex colour code: #ffffff

#### *Disabled*

Foreground (text) colour when a neutral button is disabled.

Default hex colour code: #ffffff

#### *Mouse over*

Foreground (text) colour when a neutral button is moused over.

Default hex colour code: #ffffff

## Border

Border colour of a neutral button.

### *Enabled*

Border colour when a neutral button is enabled.

Colour code: `rgba(187, 188, 245, 0.73)`

### *Disabled*

Border colour when a neutral button is disabled.

Default hex colour code: `#a3a6a7`

### *Mouse over*

Border colour when a neutral button is moused over.

Default hex colour code: `#d0d1d2`

## 3.3.2.4. Neutral alternative

Neutral alternative buttons also execute neutral actions, such as signing.

### Start signing

Select a signer below

Sign immediately as Signer 01



## Background

Background colour of a neutral alternative button.

### *Enabled*

Background colour when a neutral alternative button is enabled.

Default hex colour code: `#00a1d0`

### *Disabled*

Background colour when a neutral alternative button is disabled.

Default hex colour code: `#7fb4c3`

### *Mouse over*

Background colour when a neutral alternative button is moused over.

Default hex colour code: `#0181a6`

## Foreground

Foreground (text) colour of a neutral alternative button.

### *Enabled*

Foreground colour when a neutral alternative button is enabled.

Default hex colour code: #ffffff

#### *Disabled*

Foreground colour when a neutral alternative button is disabled.

Default hex colour code: #ffffff

#### *Mouse over*

Foreground colour when a neutral alternative button is moused over.

Default hex colour code: #ffffff

### **Border**

Border colour of a neutral alternative button.

#### *Enabled*

Border colour when a neutral alternative button is enabled.

Default hex colour code: #00a1d0

#### *Disabled*

Border colour when a neutral alternative button is disabled.

Default hex colour code: #7fb4c3

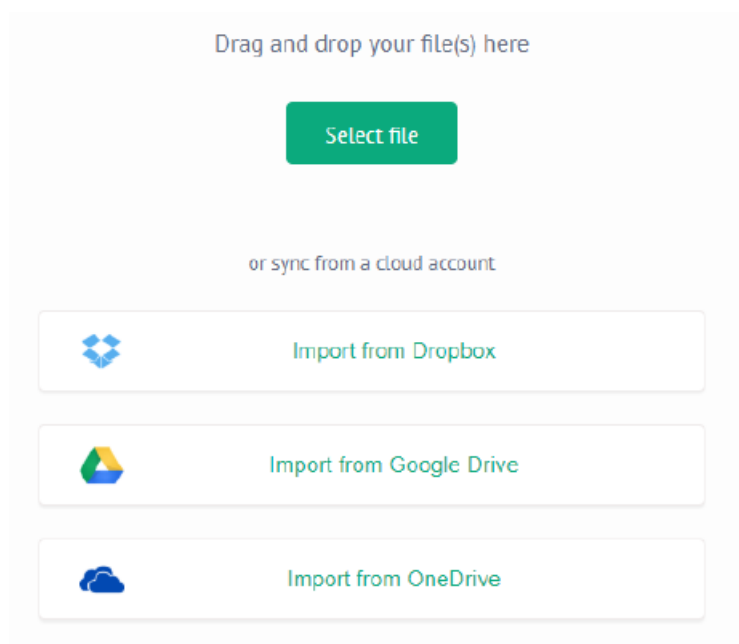
#### *Mouse over*

Border colour when a neutral alternative button is moused over.

Default hex colour code: #0181a6

### 3.3.2.5. Cloud

The Cloud buttons allow users to upload documents from Dropbox, Google Drive and OneDrive.



### **Background**

Background colour of a Cloud button.

*Enabled*

Background colour when a Cloud button is enabled.

Default hex colour code: #ffffff

*Disabled*

Background colour when a Cloud button is disabled.

Default hex colour code: #ffffff

*Mouse over*

Background colour when a Cloud button is moused over.

Default hex colour code: #ffffff

**Foreground**

Foreground (text) colour of a Cloud button.

*Enabled*

Foreground colour when a Cloud button is enabled.

Default hex colour code: #00a1d0

*Disabled*

Foreground colour when a Cloud button is disabled.

Default hex colour code: #7fb4c3

*Mouse over*

Foreground colour when a Cloud button is moused over.

Default hex colour code: #0181a6

**Border**

Border colour of a Cloud button.

*Enabled*

Border colour when a Cloud button is enabled.

Default hex colour code: #efebeb

*Disabled*

Border colour when a Cloud button is disabled.

Default hex colour code: #efebeb

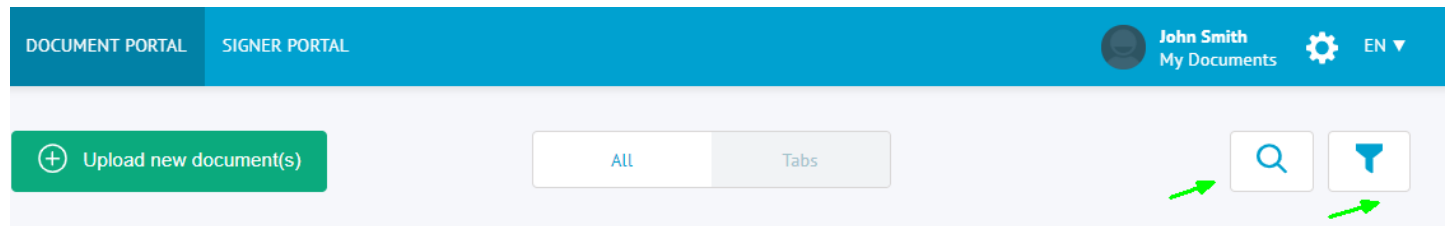
*Mouse over*

Border colour when a Cloud button is moused over.

Default hex colour code: #0181a6

### 3.3.2.6. Small

Small buttons are for instance the search button and filter buttons in the Document Portal.



#### Background

Background colour of the small buttons.

##### *Enabled*

Background colour when a small button is enabled.

Default hex colour code: #ffffff

##### *Disabled*

Background colour when a small button is disabled.

Default hex colour code: #ffffff

##### *Mouse over*

Background colour when a small button is moused over.

Default hex colour code: #ffffff

#### Forereground

Foreground colour of the small buttons.

##### *Enabled*

Foreground colour when a small button is enabled.

Default hex colour code: #00a1d0

##### *Disabled*

Foreground colour when a small button is disabled.

Default hex colour code: #7fb4c3

##### *Mouse over*

Foreground colour when a small button is moused over.

Default hex colour code: #0181a6

#### Border

Border colour of a Cloud button.

##### *Enabled*

Border colour when a Cloud button is enabled.

Default hex colour code: #efebeb

#### *Disabled*

Border colour when a Cloud button is disabled.

Default hex colour code: #efebeb

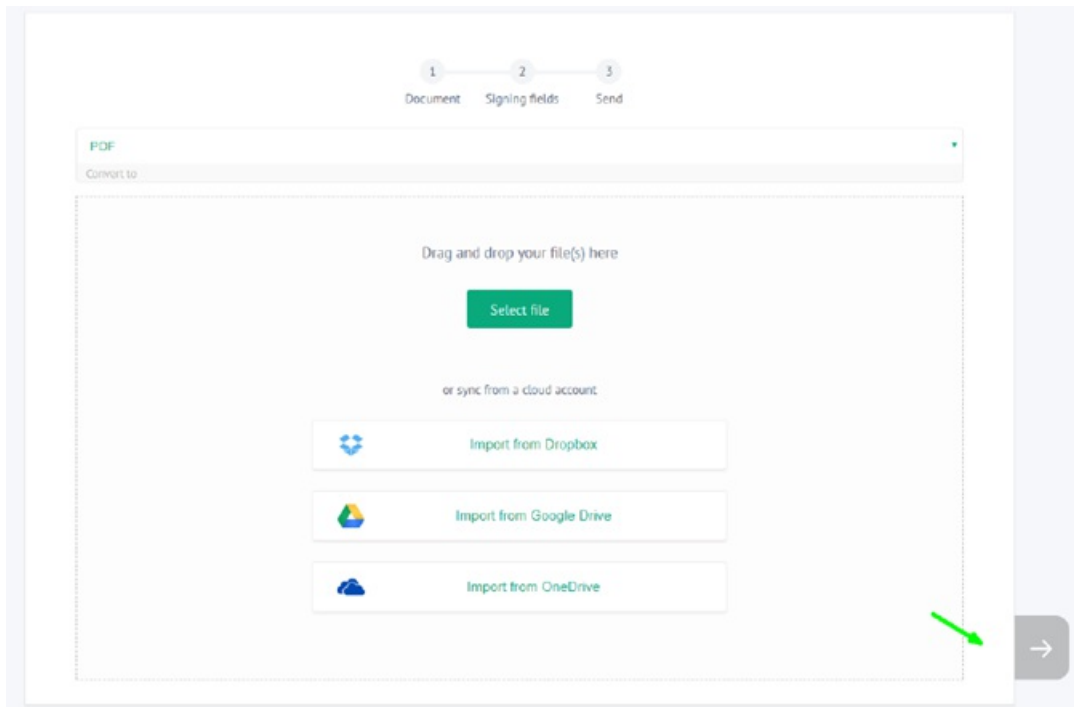
#### *Mouse over*

Border colour when a Cloud button is moused over.

Default hex colour code: #0181a6

### 3.3.2.7. Next

Next buttons allow users to go to the next steps in eSignatures.



#### **Background**

Background colour of the Next buttons.

#### *Enabled*

Background colour when a Next button is enabled.

Default hex colour code: #0baa7d

#### *Disabled*

Background colour when a Next button is disabled.

Default RGBA colour code: rgba(187, 188, 245, 0.73)

#### *Mouse over*

Background colour when a Next button is moused over.

Default hex colour code: #033225

#### **Foreground**

Foreground colour of the Next buttons.

#### *Enabled*

Foreground colour when a Next button is enabled.

Default hex colour code: #ffffff

#### *Disabled*

Foreground colour when a Next button is disabled.

Default hex colour code: #ffffff

#### *Mouse over*

Foreground colour when a Next button is moused over.

Default hex colour code: #ffffff

### **Border**

Border colour of the Next buttons.

#### *Enabled*

Border colour when a Next button is enabled.

Default hex colour code: #0baa7d

#### *Disabled*

Border colour when a Next button is disabled.

Default RGBA colour code: rgba(187, 188, 245, 0.73)

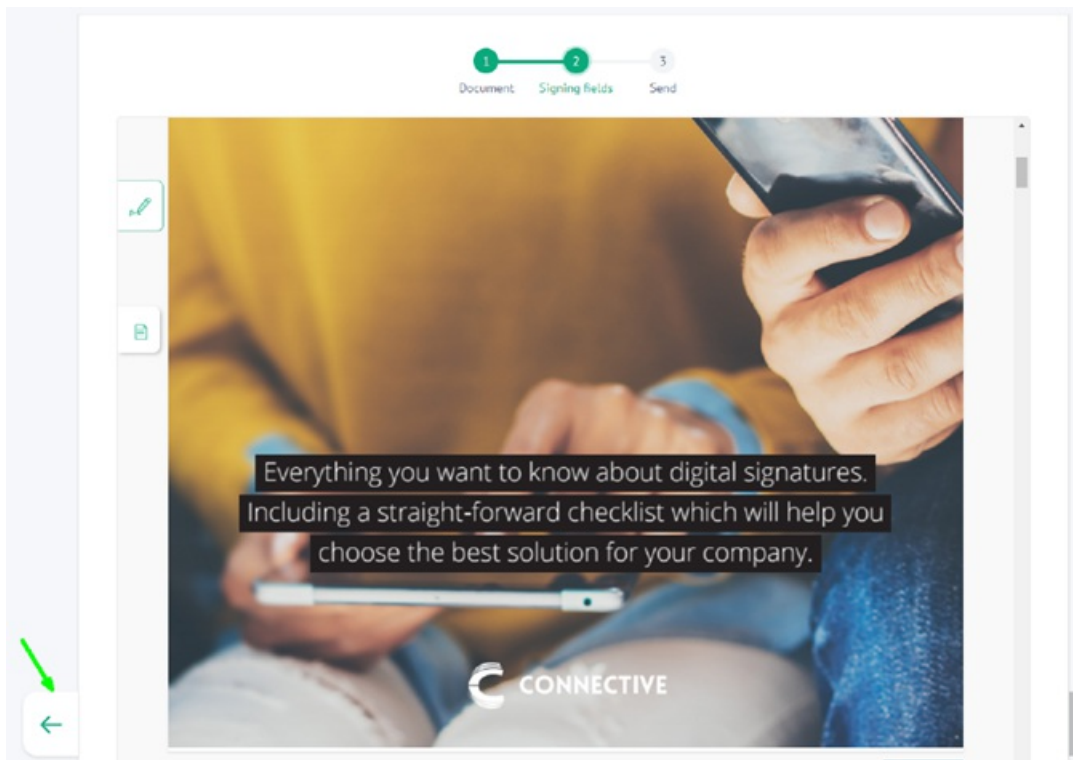
#### *Mouse over*

Border colour when a Next button is moused over.

Default hex colour code: #033225

### **3.3.2.8. Previous**

Previous buttons allow users to go to the previous steps in eSignatures.



## Background

Background colour of the Previous buttons.

### *Enabled*

Background colour when a Previous button is enabled.

Default hex colour code: #ffffff

### *Disabled*

Background colour when a Previous button is disabled.

Default RGBA colour code: rgba(187, 188, 245, 0.73)

### *Mouse over*

Background colour when a Previous button is moused over.

Default hex colour code: #f5f5f5

## Foreground

Foreground colour of the Previous buttons.

### *Enabled*

Foreground colour when a Previous button is enabled.

Default hex colour code: #0baa7d

### *Disabled*

Foreground colour when a Previous button is disabled.

Default hex colour code: #ffffff

### *Mouse over*



Foreground colour when a Previous button is moused over.

Default hex colour code: #0baa7d

## **Border**

Border colour of the Previous buttons.

### *Enabled*

Border colour when a Previous button is enabled.

Default hex colour code: #ffffff

### *Disabled*

Border colour when a Previous button is disabled.

Default RGBA colour code: rgba(187, 188, 245, 0.73)

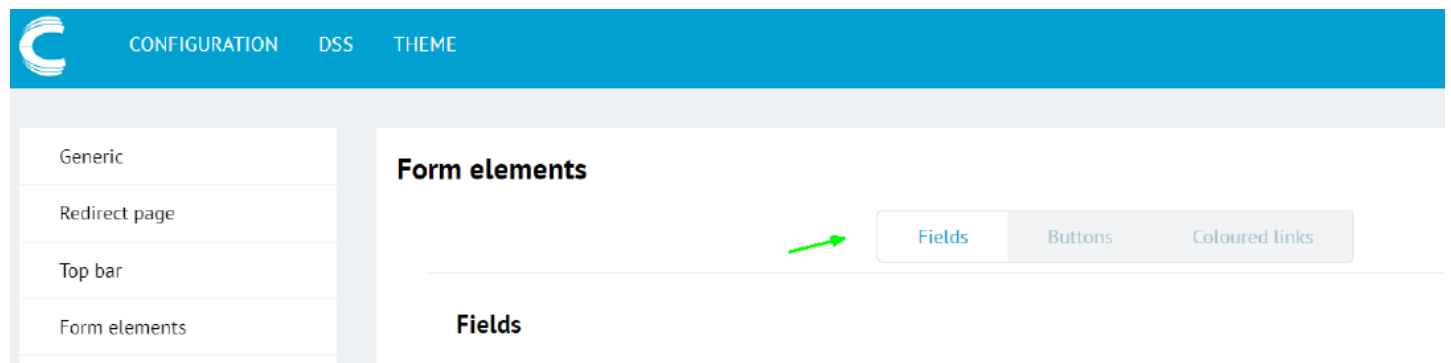
### *Mouse over*

Border colour when a Previous button is moused over.

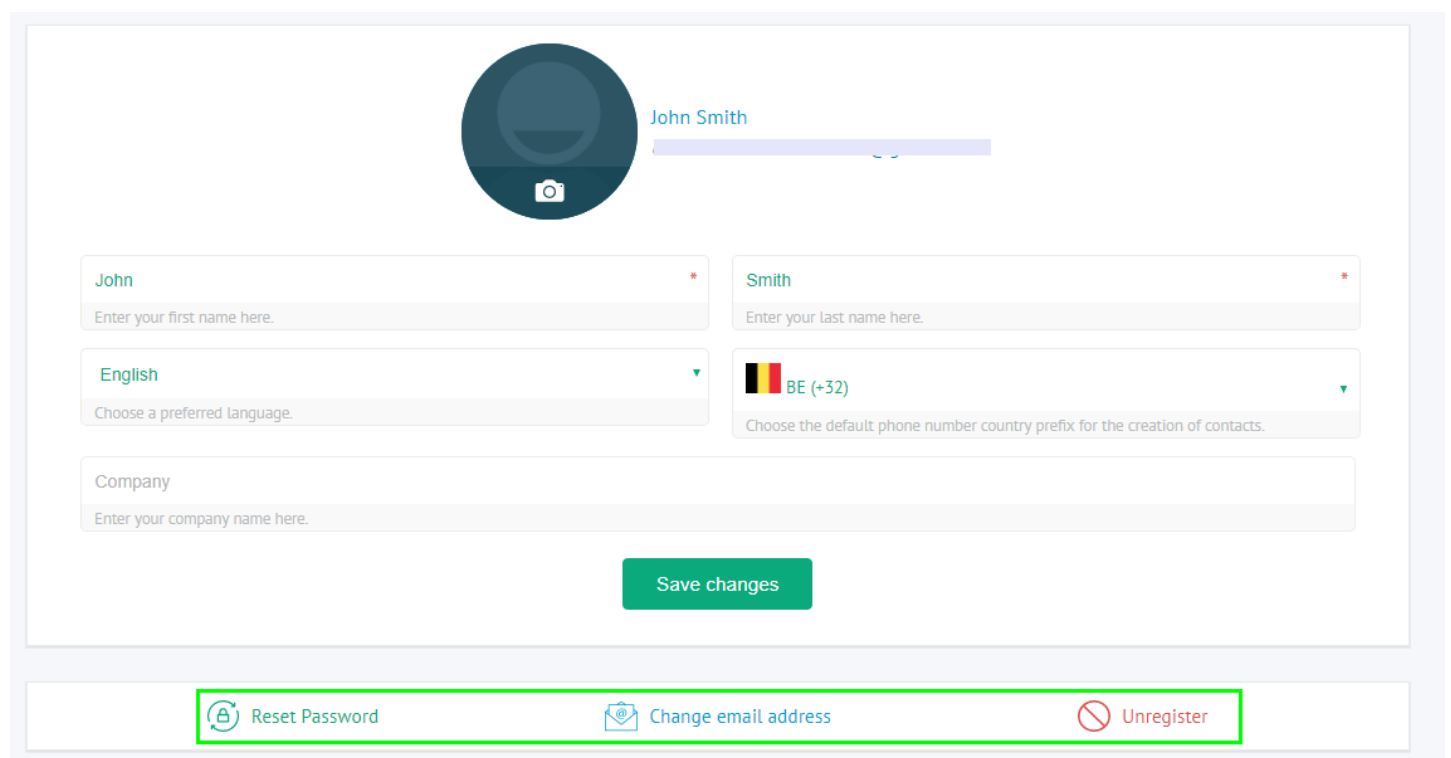
Default hex colour code: #f5f5f5

### 3.3.3 colored links

Click the **colored links** tab to access the corresponding settings.



Colored links are used in eSignatures to reset passwords, change a user's email address, unregister a user's account, etc.



#### 3.3.3.1. Positive

Positive colored links execute positive actions, such as resetting your password.

##### Foreground

Foreground (text) color of a positive link.

##### Enabled

Foreground color when a positive link is enabled.

Default hex color code: #0baa7d

##### Disabled

Foreground color when a positive is disabled.

Default hex color code: #64ccaf

#### 3.3.3.2. Negative

Negative colored links execute negative actions, such as unregistering your account.

### **Foreground**

Foreground (text) color of a negative link.

#### *Enabled*

Foreground color when a negative link is enabled.

Default hex color code: #d9534f

#### *Disabled*

Foreground color when a negative link is disabled.

Default hex color code : #e68f8d

### **3.3.3.3. Neutral**

Neutral colored links execute neutral actions, such as changing an email address.

### **Foreground**

Foreground (text) color of a neutral link.

#### *Enabled*

Foreground color when a neutral link is enabled.

Default color code: rgba(187, 188, 245, 0.73)

#### *Disabled*

Foreground color when a neutral link is disabled.

Default color code: rgba(187, 188, 245, 0.73)

### **3.3.3.4. Neutral alternative**

Neutral alternative colored links also execute neutral actions.

### **Foreground**

Foreground (text) color of a neutral alternative link.

#### *Enabled*

Foreground color when a neutral alternative link is enabled.

Default hex color code: #00a1d0.

#### *Disabled*

Foreground color when a neutral alternative link is disabled.

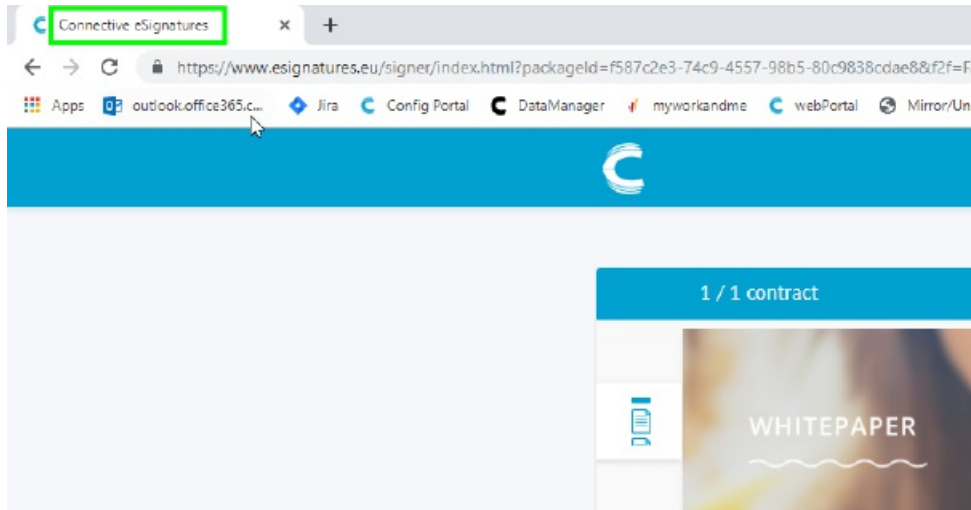
Default hex color code: #7fb4c3.

## 3.4 WYSIWYS

The WYSIWYS settings customize the look and feel of the What you sign is what you sign page (WYSIWYS). This is the page that end users, i.e. signers, will see.

### Tab title

The Tab title is the title of the Web browser tab in which the WYSIWYS is opened.



### Favicon

The WYSIWYS icon is displayed on the Tab title.

Click **Choose File** to browse for the icon you want to use.

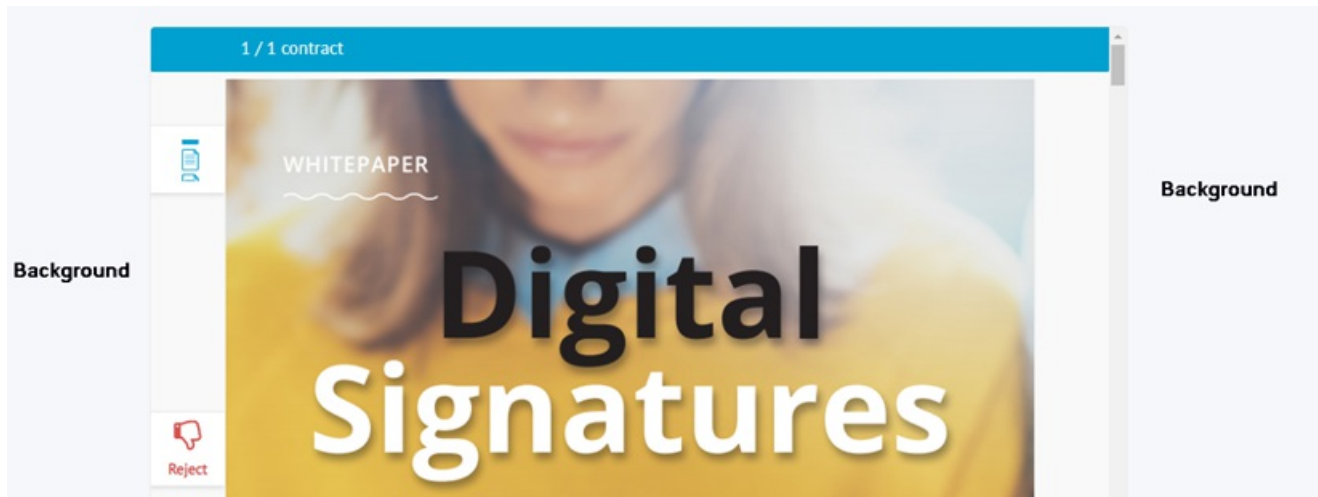
### Attention:

- Currently only **.ico** is supported as Favicon file format.
- Maximum size of the image file is 0.5 MB.

### 3.4.1 Page

The Page settings customize the look and feel of the WYSIWYS landing page.

#### 3.4.1.1. Background



#### Background type

Select the Background type from the list:

- **image** When you select image, an image will be shown as background of the WYSIWYS.
- **colour** When you select colour, you can customize the background colour of the WYSIWYS, without background image.

#### Background image

This setting applies only if you select image as Background type above.

- Click **Choose File** to browse for the required background image. The following image files are supported: .png, jpeg and .gif. **Attention:** maximum file size is 2 MB.
- Select the required file and click **Open**.

#### Background image style

This setting applies only if you select image as Background type above.

Select the Background image style from the list:

- **Repeat:** repeats the background image over the horizontal and vertical axes of the background.
- **Cover:** the background image covers the complete background of the page.

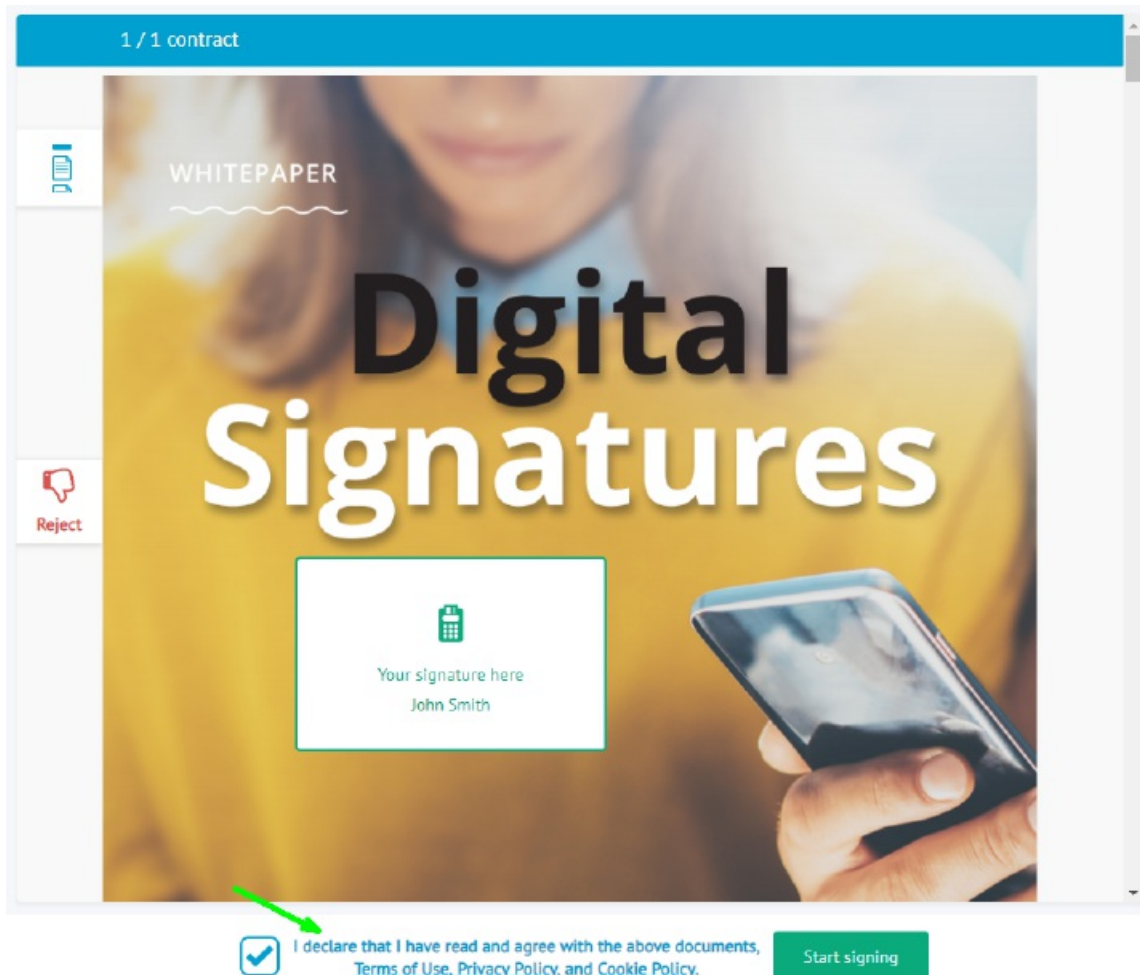
#### Background colour

Background colour of the WYSIWYS.

Default hex colour code: #f6f7fb

#### 3.4.1.2. Body

Body text of the WYSIWYS.



## Default

Colour of the default text.

Default hex colour code: #00a1d0

## Negative

Colour of negative text, such as in the Reject button.

Default hex colour code: #d9534f

## Alternative

Colour of alternative text, such as in the download unsigned documents icon.

Default hex colour code: #0baa7d

### 3.4.1.3. Signature placeholder

The signature placeholder indicates where the user must place their signature.



## Background

Background colour of the signature placeholder.

Default hex colour code: #ffffff

### **Foreground**


Foreground (text) colour of the signature placeholder.

Default hex colour code: #0baa7d

### 3.4.2 Modal

Click the **Modal** tab to display the Modal Settings.

#### WYSIWYS

Tab title	Connective eSignatures
Favicon	<div>Choose File No file chosen</div> <div></div>
<div>Page Modal</div>	

The **Modal** settings customize the look and feel of the signing modal. The signing modal is the pop-up screen in which you sign.

×

1

2


### Sign manually

Signing field 1 of contract

Sign in the field below.

Retry

Next



#### 3.4.2.1. Background

The **Background** settings customize the background of the signing modal.



## Modal

[Reset to defaults](#)

### Background

[Reset to defaults](#)

Background type

image ▼

Background image

Choose File

No file chosen

Background image style

repeat ▼

Background colour

### Background type

Select the **Background type** from the list:

- **Image** When you select image, an image will be shown as background of the modal.
- **Colour** When you select colour, you can customize the background colour of the modal, without background image.

### Background image

This setting applies only if you select image as Background type above.

- Click **Choose File** to browse for the required background image. The following image files are supported: .png, jpeg and .gif. **Attention:** maximum file size is 2 MB.
- Select the required file and click Open.

### Background image style

This setting applies only if you select image as Background type above.

Select the Background image style from the list:

- **Repeat:** repeats the background image over the horizontal and vertical axes of the background.
- **Cover:** the background image covers the complete background of the page.

### Background colour

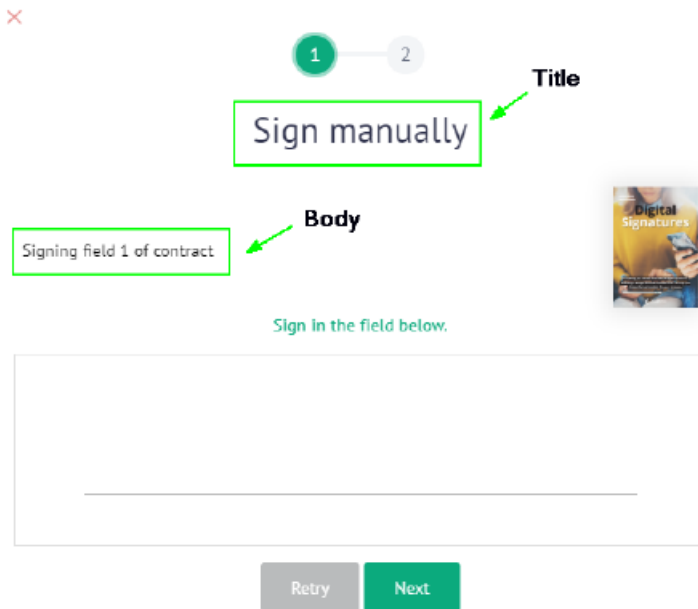
This setting applies only if you select colour as **Background type** above.

Default hex colour code: #ffffff

#### 3.4.2.2. Title

This setting customizes the text colour of the title displayed in the modal.

Default hex colour code: #31344e



#### 3.4.2.3. Body

Body text of the Modal.

##### **Default**

Colour of the default text.

Default hex colour code: #000000

##### **Negative**

Colour of the negative text.

Default hex colour code: #d9534f

##### **Alternative**

Colour of the alternative text.

Default hex colour code: #0baa7d

#### 3.4.2.4. Error

These settings customize the look of error messages in the Modal.

##### **Background**

Background colour of error messages.

Default hex colour code: #e68f8d

##### **Foreground**

Foreground (text) colour of error messages.

Default hex colour code: #d9534f

#### 3.4.2.5. Numeric progress bar

The Numeric progress bar shows the different steps of the process the signer is doing in the WYSIWYS.



Signing field 1 of contract



Sign in the field below.

Retry Next

## Step number

The step numbers indicate the number of steps in the process.

### *Background*

Background colour of the step numbers.

### *Visited*

Background colour when a step has been visited.

Default hex colour code: #0baa7d

### *Unvisited*

Background colour when a step has not been visited yet.

Default hex colour code: #f3f5f7

### *Active*

Background colour of the current, active step.

Default hex colour code: #0baa7d

### *Foreground*

Foreground (text) colour of the step numbers.

### *Visited*

Foreground colour when a step has been visited.

Default hex colour code: #ffffff

### *Unvisited*

Foreground colour when a step has not been visited yet.

Default hex colour code: #646c7f

### *Active*

Foreground colour of the current, active step.

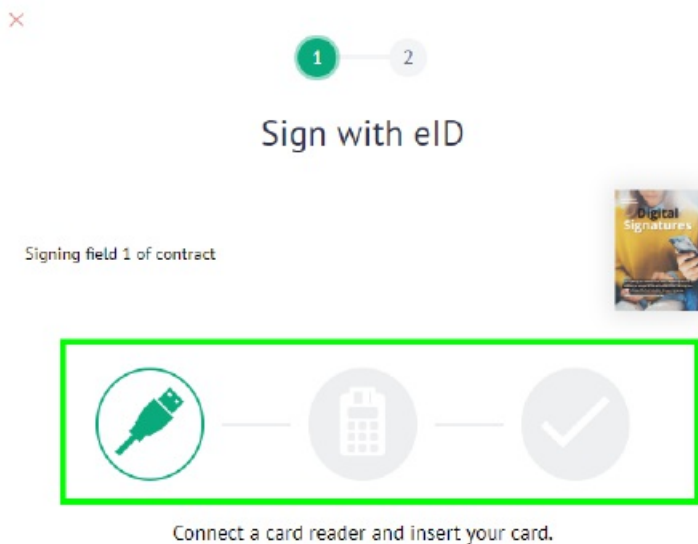
Default hex colour code: #ffffff

## **Step label**

This setting is currently not in use, and is reserved for future use.

### 3.4.2.6. Progress Indicator

The Progress indicator is similar to the Numeric progress bar. It also shows which steps a user has covered but does so by means of icons instead of numbers.



## **Step number**

### *Background*

Background colour of the step.

*Visited* Background colour when the step has been visited.

Default hex colour code: #0baa7d

### *Unvisited*

Background colour when the step has not been visited yet.

Default hex colour code: #edeff0

### 3.4.2.7. Selection button

Selection buttons allow users to select signing methods in the WYSIWYS.



## Select a signing method.

Signing field 1 of contract



Select one of the available signing methods below and click 'Next'.

 eID

 Manual

Next

### Background

Background of the selection buttons.

*Selected*

*Colour*

Background colour when the button is selected.

Default hex colour code: #0baa7d

*Mouse over*

Background colour when the button is moused over.

Default hex colour code: #0baa7d

*Unselected*

*Colour*

Background colour when the button is unselected.

Default hex colour code: #ffffff

*Mouse over*

Background colour when the button is unselected.

Default hex colour code: #ffffff

### Foreground

Foreground of the selection buttons.

*Selected*

*Colour*

Foreground colour when the button is selected.

Default hex colour code: #ffffff

*Mouse over*

Foreground colour when the button is moused over.

Default hex colour code: #ffffff

### *Unselected*

#### *Colour*

Foreground colour when the button is unselected.

Default hex colour code: #646c7f

#### *Mouse over*

Foreground colour when the button is unselected.

Default hex colour code: #0baa7d

## 3.5 Portal

The **Portal** settings allow to customize the eSignatures WebPortal before and after a user has logged in.

### Tab title

The Tab title is the title of the Web browser tab in which the WebPortal is opened.



### Favicon

The WebPortal icon is displayed on the Tab title.

Click **Choose File** to browse for the icon you want to use.

### Attention:

- Currently only **.ico** is supported as Favicon file format.
- Maximum size of the image file is 0.5 MB.

### 3.5.1 Before login

The **Before login** settings customize the login screen.

#### Portal

Tab title

Connective eSignatures

Favicon

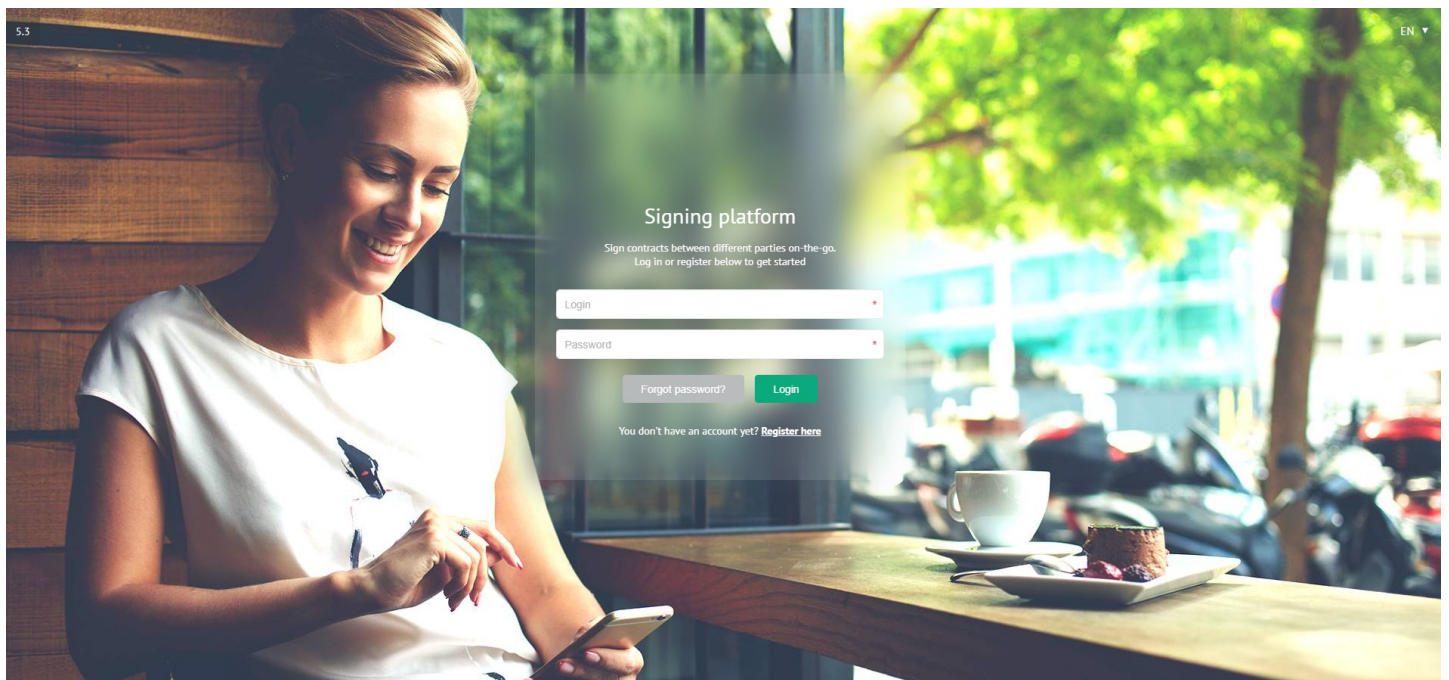
Choose File No file chosen



Before login

After login

The default login screen looks as follows:



#### 3.5.1.1. Background

##### Background image style

Select the Background image style from the list:

- **Repeat:** repeats the background image over the horizontal and vertical axes of the background.
- **Cover:** the background image covers the complete background of the page.

##### Background image

- Click **Choose File** to browse for the required background image. The following image files are supported: .png, .jpeg and .gif. **Attention:** maximum file size is 2 MB.
- Select the required file and click Open.

#### 3.5.1.2 Title

##### Colour

Text colour of the title on the login page title.



Default hex colour code: #ffffff

Note that you can't change the actual text of the title. It always says "Signing Platform".

#### 3.5.1.3. Body

Body text displayed on the login page.

##### **Default**

Default body text colour displayed on the login page.

Default hex colour code: #ffffff

##### **Alternative**

This setting is currently not in use and is reserved for future use.

#### 3.5.1.4. Error

Customizes the look of error messages when displayed on the login screen.

##### **Background**

Background colour of the error messages.

Default hex colour code: #f9e4e4

##### **Foreground**

Foreground (text) colours of the error messages.

Default hex colour code: #9e1f1c

3.5.2 After login

These settings determine what the eSignatures WebPortal looks like after a user has logged in.

Click the **After login** tab to access them.

Portal


Tab title

Connective eSignatures

Favicon

Choose File

No file chosen



Before login

After login

3.5.2.1. Title

Colour

This setting determines the colour of the titels displayed in modals opened in the Document Portal. of the title.

Default hex colour code: #0baa7d

Start signing  
Select a signer below


Sign immediately as john smith










Sign

Done Sign all

3.5.2.2. Content wrapper

The Content wrapper is the frame around the displayed documents.



Name	Status	Added on ▼	Signers	Receivers	Actions
 contract.pdf				 Sign  Notify  Revoke 	
 contract.pdf	Failed	06/05/2019	1/1	-	...
 contract.pdf	Pending	30/04/2019	0/1	-	...
 contract.pdf	Signed	30/04/2019	1/1	-	...
 contract.pdf	Signed	30/04/2019	1/1	-	...

Colour

Background colour of the Content wrapper.

Default hex colour code: #ffffff

### 3.5.2.3. Body

Body text displayed in the WebPortal.

#### **Default**

Default colour of body text in the WebPortal.

Default hex colour code: #000000

#### **Negative alternative**

Colour of negative alternative body text in the WebPortal.

Default hex colour code: #d9534f

#### **Neutral**

Color of neutral body text in the WebPortal.

Default hex colour code: #b9bbbc

#### **Neutral Alternative**

Color of neutral alternative body text in the WebPortal.

Default hex colour code: #00a1d0

#### **Alternative**

Color of alternative body text in the WebPortal.

Default hex colour code: #0baa7d

### 3.5.2.4. Error

These settings customize the look of error messages.

You can't change your email address to the same address as you currently have.

Cancel

Confirm

#### **Background**

Background colour of error messages.

Default hex colour code: #e68f8d

#### **Foreground**

Foreground (text) colour of error messages.

Default hex colour code: #d9534f

### 3.5.2.5. Numeric progress bar

The Numeric progress bar shows the different steps of the process the user is doing in the WebPortal.



**Attention:** in the WYSIWYS you also have a numeric progress bar. This bar can be configured in 6. WYSIWYS settings.

### Step number

The step numbers indicate the number of steps in the process.

#### *Background*

Background colour of the step numbers.

#### *Visited*

Background colour when a step has been visited.

Default hex colour code: #0baa7d

#### *Unvisited*

Background colour when a step has not been visited yet.

Default hex colour code: #f3f5f7

#### *Active*

Background colour of the current, active step.

Default hex colour code: #0baa7d

#### *Foreground*

Foreground (text) colour of the step numbers.

#### *Visited*

Foreground colour when a step has been visited.

Default hex colour code: #ffffff

#### *Unvisited*

Foreground colour when a step has not been visited yet.

Default hex colour code: #646c7f

#### *Active*

Foreground colour of the current, active step.

Default hex colour code: #ffffff

### Step label

The step label is the description beneath each step of the numeric progress bar.



#### Foreground

Foreground (text) colour of the step label.

#### Visited

Foreground colour when a step has not been visited yet.

Default hex colour code: #646c7f

#### Unvisited

Foreground colour when a step has not been visited yet.

Default hex colour code: #646c7f

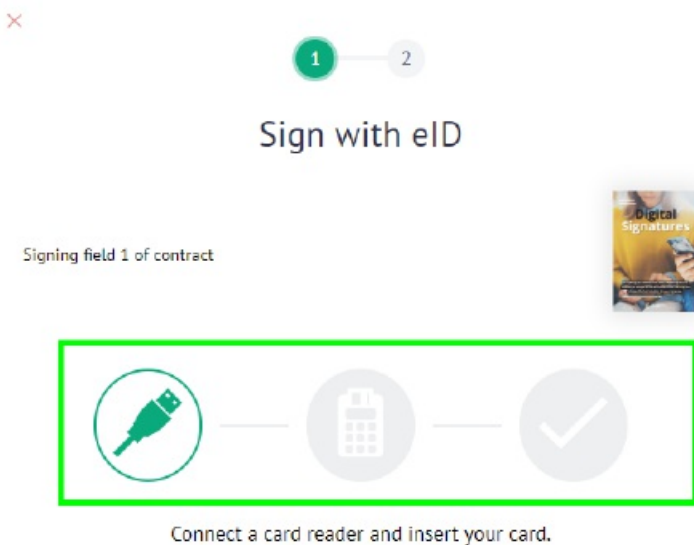
#### Active

Foreground colour of the current, active step.

Default hex colour code: #0baa7d

### 3.5.2.6. Progress Indicator

The Progress indicator is similar to the Numeric progress bar. It also shows which steps a user has covered but does so by means of icons instead of numbers.



### Step number

#### Background

Background colour of the step.

*Visited*

Background colour when the step has been visited.

Default hex colour code: #0baa7d



*Unvisited*

Background colour when the step has not been visited yet.

Default hex colour code: #edeff0

3.5.2.7. Pager

The Pager indicates over how many pages the documents are spread in the WebPortal.

▶	 contract.pdf	Signed	30/04/2019	1/1	-	...
▶	 contract.pdf	Signed	30/04/2019	1/1	-	...
<div><div>1</div><div>2</div><div>&gt;</div></div>						

**Background**

Background colour of the pager.

*Selected*

Background colour when the page is selected.

Default hex colour code: #00a1d0

*Unselected*

Background colour when the page is unselected.

Default hex colour code: #ffffff

*Mouse over*

Background colour when the page is moused over.

Default hex colour code: #eef1f3

**Foreground**

Foreground (text) colour of the pager.

*Selected*

Foreground colour when the page is selected.

Default hex colour code: #ffffff

*Unselected*

Foreground colour when the page is unselected.

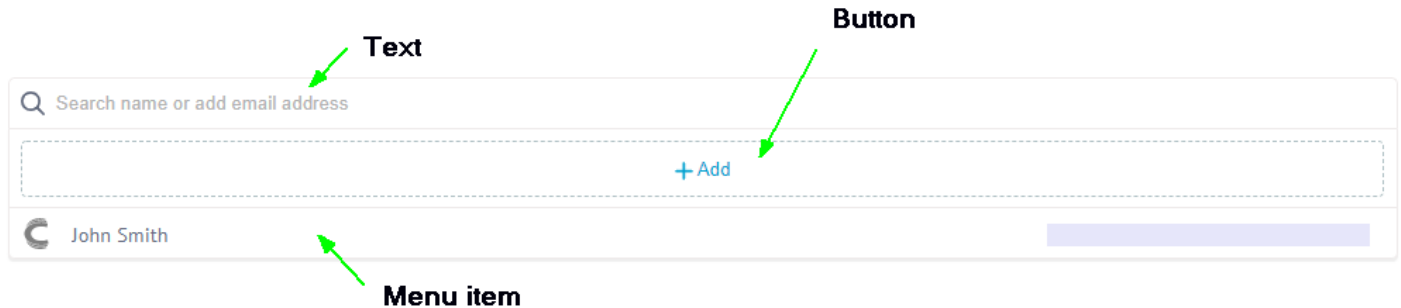
Default hex colour code: #00a1d0

*Mouse over* Foreground colour when the page is moused over.

Default hex colour code: #eef1f3

### 3.5.2.8. Contact selection field

The Contact selection field is the field where you select a signer or receiver. The Contact selection field consists of multiple customizable components: text, button and menu item.



#### **Text**

Text displayed in the Contact selection field.

##### *Input*

Colour of the input text a user enters in the content selection field.

Default hex colour code: #0baa7d

##### *Placeholder*

Colour of the placeholder text in the content selection field.

Default hex colour code: #b3b3b3

#### **Button**

"Add" button in the Contact selection field.

##### *Background*

Background colour of the Add button.

*Unselected* Background colour when the Add button is unselected.

Default hex colour code: #ffffff

##### *Mouse over*

Background colour when the Add button is moused over.

Default hex colour code: #0baa7d

##### *Foreground*

Foreground colour of the Add button.

##### *Unselected*

Foreground colour when the Add button is unselected.

Default hex colour code: #00a1d0

*Mouse over*

Foreground colour when the Add button is moused over.

Default hex colour code: #ffffff

**Menu item**

Menu items are the items that are displayed in the Contact selection field.

**Foreground**

Foreground colour of the menu item.

*Mouse over*

Foreground colour when the menu item is moused over.

Default hex colour code: #00a1d0

*Body text*

Colour of the body text of the menu item.

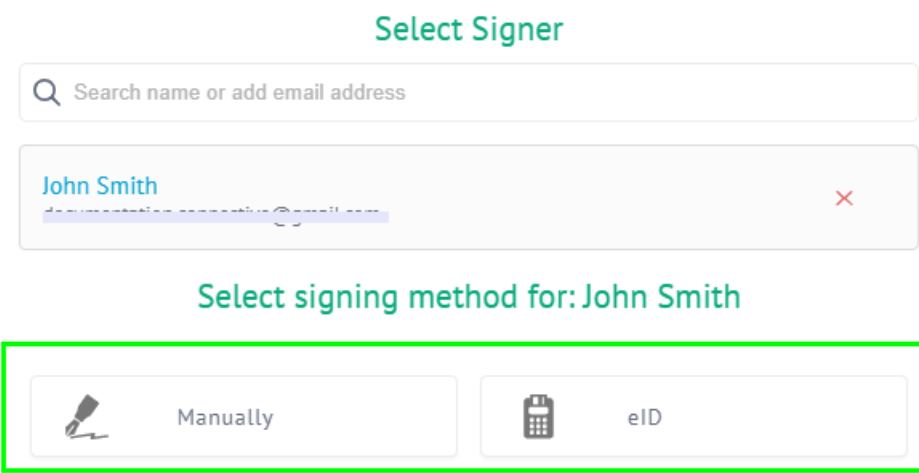
*Light*

Colour of the light text displayed on the right-hand side of the menu item. I.e. the email address.

Default hex colour code: rgba(185, 187, 188, 0.96)

**Selection button**

Selection buttons allow users to select signing methods.



*Background*

Background of the selection buttons.

*Selected*

*Colour*

Background colour when the button is selected.

Default hex colour code: #0baa7d



#### *Mouse over*

Background colour when the button is moused over.

Default hex colour code: #033225

#### *Unselected*

##### *Colour*

Background colour when the button is unselected.

Default hex colour code: #ffffff

#### *Mouse over*

Background colour when the button is unselected.

Default hex colour code: #ffffff

#### *Foreground*

Foreground of the selection buttons.

#### *Selected*

##### *Colour*

Foreground colour when the button is selected.

Default hex colour code: #ffffff

#### *Mouse over*

Foreground colour when the button is moused over.

Default hex colour code: #ffffff

#### *Unselected*

##### *Colour*

Foreground colour when the button is unselected.

Default hex colour code: #646c7f

#### *Mouse over*

Foreground colour when the button is unselected.

Default hex colour code: #0baa7d

### 3.5.2.9. Signature placeholder

The signature placeholder indicates where the user must place their signature.



**Background**

Background colour of the signature placeholder.

Default hex colour code: #64ccaf

**Foreground**

Foreground (text) colour of the signature placeholder.

Default hex colour code: #0baa7d

3.5.2.10. Separator

The Separator separates the listed documents from the tabs.

**Colour**

Colour of the separator.

Default hex colour code: #7fb4c3

Name	Status	Added on ▼	Signers	Receivers	Actions
▶  contract.pdf				Sign    Notify    Revoke	
▶  contract.pdf	Failed	06/05/2019	1/1	-	...
▶  contract.pdf	Pending	30/04/2019	0/1	-	...
▶  contract.pdf	Signed	30/04/2019	1/1	-	...
▶  contract.pdf	Signed	30/04/2019	1/1	-	...

3.5.2.11. Big tabs

Upload new document(s)		All   Tabs			
Draft	Pending	Revoked	Rejected	Expired	Signed
Name	Added on ▼	Signers	Receivers	Actions	

**Background**

Background colour of the big tabs.

### *Unselected*

Background colour when the big tabs are unselected.

### *Colour*

Background colour when a big tab is unselected.

Default hex colour code: #00a1d0

### *Mouse over*

Colour when a big tab is moused over.

Default hex colour code: #ffffff

## **Foreground**

Foreground colour of the big tabs.

### *Selected*

Foreground colour when the big tabs are selected.

### *Colour*

Foreground colour when a big tab is selected.

Default hex colour code: #00a1d0

### *Mouse over*

Foreground colour when a big tab is moused over.

Default hex colour code: #00a1d0

### *Unselected*

Foreground colour when the big tabs are unselected.

### *Colour*

Foreground colour when a big tab is unselected.

Default hex colour code: #ffffff

### *Mouse over*

Foreground colour when a big tab is unselected.

Default hex colour code: #00a1d0

## 3.5.2.12. Small tabs

Small tabs are displayed when editing user groups in the Access Management section.

Update

admin

General

Users

Document groups

Template groups

## Foreground

Foreground colour of the small tabs.

### *Selected*

Foreground colour when the small tabs are selected.

### *Colour*

Foreground colour when a small tab is selected.

Default hex colour code: #ffffff

### *Mouse over*

Foreground colour when a small tab is moused over.

Default hex colour code: #ffffff

### *Unselected*

Foreground colour when the small tabs are unselected.

### *Colour*

Foreground colour when a small tab is unselected.

Default hex colour code: #646c7f

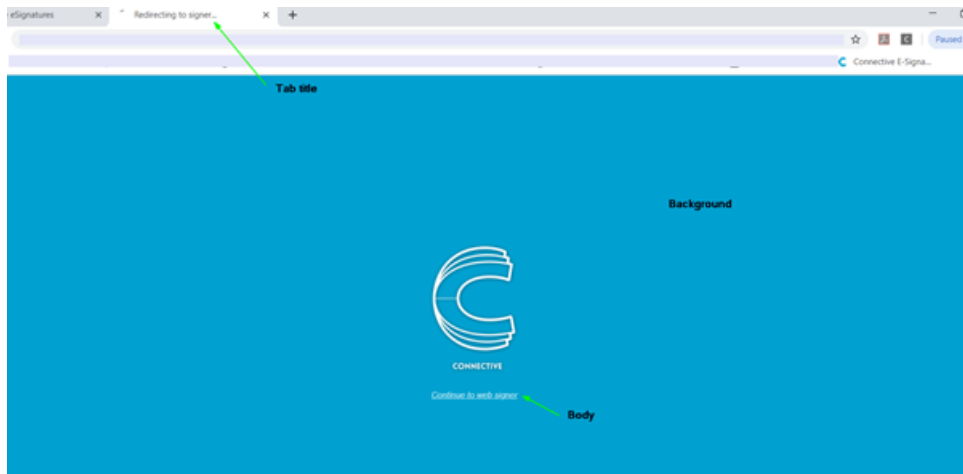
### *Mouse over*

Foreground colour when a small tab is unselected.

Default hex colour code: #0baa7d

## 3.6 Redirect page

The Redirect page is briefly displayed after you clicked **Sign**, and before you reach the **What you see is what you sign page** (WYSIWYS).



This is also the page displayed in the Connective app.

### Tab title

This is the title of the Web browser tab during the redirect action. Note that you see this title only very briefly.

### Favicon

The redirect icon is displayed on the Tab title.

Click **Choose File** to browse for the icon you want to use.

**Attention:** maximum size of the image file is 0.5 MB.

### 3.6.1 Background

Background of the redirect page.

### Colour

Background colour of the redirect page.

Default hex colour code: #00a1d0

### 3.6.2. Body

Body (text) of the redirect page.

### Default

Default body text colour of the redirect page.

Default hex colour code: #ffffff

**Note:** the logo in the center of the Redirect page is configured in the [Generic settings](#).

## 4. Apply themes to an eSignatures environment

Once a theme has been created, it must be applied for eSignatures to have a rebranded look.

Themes can be applied on different levels:

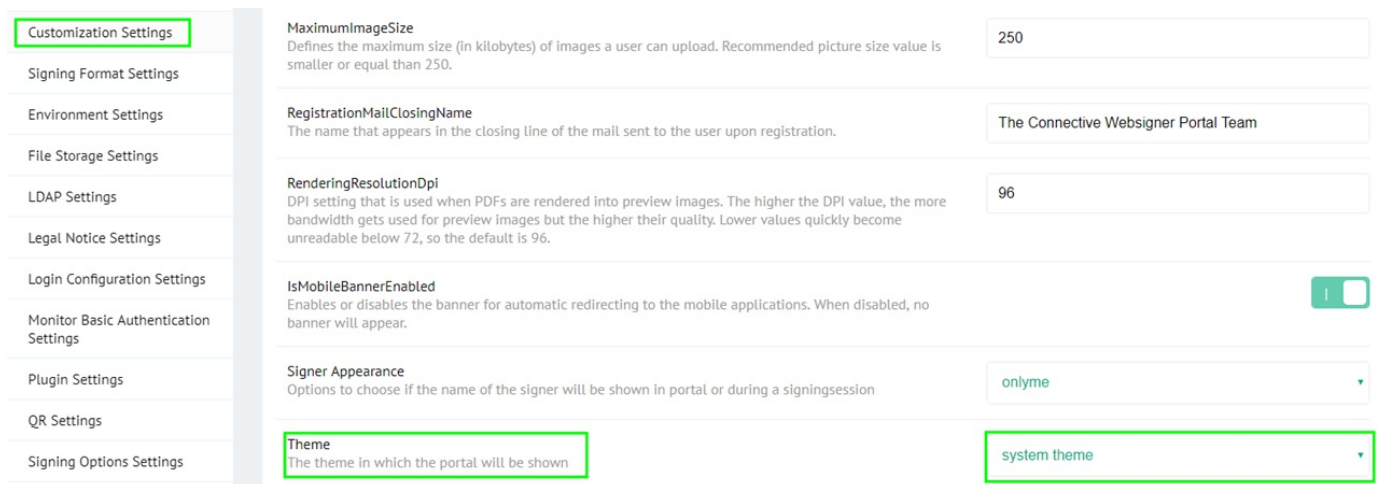
- Environment level
- Document groups level
- Package level

### Environment level

When you apply a theme on environment level, the entire eSignatures environment will have the same look and feel. Every user and signer will see the same interface.

#### To apply a theme on environment level:

- Access the Configuration Index. To do so, you need administrator rights. If you don't have credentials yet, or don't know them, contact your system administrator or Connective.
- Go to **Configuration > Customization Settings**.
- Click the drop-down list next to Theme and select one of the available themes.



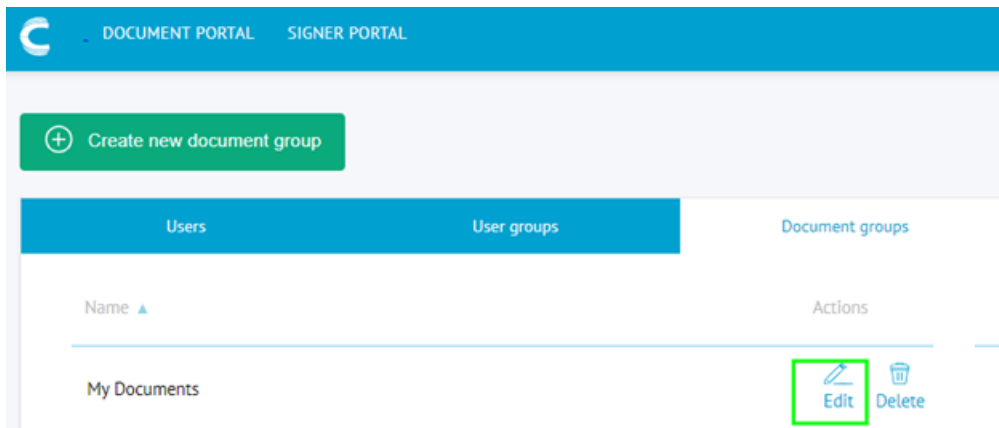
Customization Settings	MaximumImageSize Defines the maximum size (in kilobytes) of images a user can upload. Recommended picture size value is smaller or equal than 250.	250
Signing Format Settings	RegistrationMailClosingName The name that appears in the closing line of the mail sent to the user upon registration.	The Connective Websigner Portal Team
Environment Settings	RenderingResolutionDpi DPI setting that is used when PDFs are rendered into preview images. The higher the DPI value, the more bandwidth gets used for preview images but the higher their quality. Lower values quickly become unreadable below 72, so the default is 96.	96
File Storage Settings	IsMobileBannerEnabled Enables or disables the banner for automatic redirecting to the mobile applications. When disabled, no banner will appear.	<input checked="" type="checkbox"/>
LDAP Settings	Signer Appearance Options to choose if the name of the signer will be shown in portal or during a signingsession	onlyme
Legal Notice Settings	Theme The theme in which the portal will be shown	system theme
Login Configuration Settings		
Monitor Basic Authentication Settings		
Plugin Settings		
QR Settings		
Signing Options Settings		

### Document groups level

You can also choose to apply a different theme per document group. This way, all documents that are uploaded to one document group, say Sales for instance, will have their own WYSIWYS look, while others belonging to HR Documents will have a different one.

#### To apply a theme on document group level:

- Log in to eSignatures as administrator, with access to the **Access Management** section.
- Click the settings icon and click **Access Management**.
- Click the **Document groups** tab.
- Click **Edit** next to the Document to which you want to apply a custom theme.



- Click the **Themes** tab.
- Move the required theme(s) to the column "**Themes in the Document Group**".

**Note:** if you were using a custom theme in a previous version of eSignatures, the custom theme will be applied to each document group automatically.

Update document group

My Documents \*

Provide a document group name.

User groups
Themes

FA test ▼

Default theme

Search for a theme

**THEMES NOT IN THE DOCUMENT GROUP**

Ultimate testing theme	+
New theme	+
New theme 12	+
Lars breaks the portal	+
Connective Default - New Branding	+

**THEMES IN THE DOCUMENT GROUP**

System theme	×
FA test	×

1
2
3
4
>

Cancel
Save

- Click **Save**.

**Note:** when you place a single theme in the right-hand column, each package uploaded to that document group will use the same theme. If you place multiple themes in the right-hand column, initiators will be able to decide on package level which theme to apply. This way, different packages uploaded to the same document group may use different themes.

### Package level

To apply a different theme on package level, multiple themes must be available in the **Themes in the Document Group** column as described above.

### To apply a different theme on package level:

- Upload your package
- In step 1 of the upload process, select the required Document group.
- Beneath the selected document group, select one of the available themes to apply.

1 2 3  
Document Signing fields Send

contract.pdf  
Enter a package title (optional)

French  
Select the document language

My Documents  
Select a document group

FA test  
System theme  
FA test

- Proceed with the upload as usual. The new theme will be used in the document.

**Important:** if an administrator edits or deletes a theme that's already used in a package, the changes will not be reflected. The existing theme will still be used as such.



# Migration Guide

The Migration Guide describes how to approach the API changes between eSignatures API v3 and v4.

# 1. Introduction

In eSignatures 6.0 we've introduced API v4.

The main goals of API v4 are the following:

- Flexible creation and management of the different resources
- Easier implementation of new features

Keeping these principles in mind, we'll go over the big changes between eSignatures API v3 and v4 below. Rest assured however that the standard flow remains largely the same, meaning the main attention points are updated routes and new functionalities.

Note that API v3 is still available for use but is considered deprecated. This means no new functionality will be added to API v3 and the routes will be removed once API v5 will be released. We therefore strongly recommend making use of API v4 both for new and existing integrations.

API v2 routes have been removed and are no longer accessible. Functionalities listed in [2. Deprecated Functionalities](#) have also been removed and are no longer available.

**Important note:** when upgrading to eSignatures 6.x, make sure to add **/esig** to the API endpoint of all API calls (both when using API v3 and v4). The default URL is now **[https://\[servername\]:\[port\]/esig/webportalapi/v4/](https://[servername]:[port]/esig/webportalapi/v4/)**

For a full explanation of the various API v4 calls, see [eSignatures 6.3 - API v4 Technical Documentation](#).

Should you require further information or assistance, don't hesitate to contact us via [service@connective.eu](mailto:service@connective.eu) or <https://connective.eu/contact-support>. You can also request an on-site workshop!

## 2. Deprecated functionalities

The features that were marked as 'deprecated' in API v3 have now been removed.

### Templates

The Templates features and its corresponding calls has been removed.

### User groups

User groups have been removed. Instead, you can use contact groups and group stakeholders (since eSignatures 5.4).

### Notification Callback Details

The notificationType parameter is now a string, instead of a number.

The NotificationTypeKey parameter has been removed.

Please keep this change in mind when you're upgrading an existing integration!

Also note the switch to camel case in the parameter naming.

### Mandated Signer Validation parameters

The MandatedSignerValidation parameters you could pass in the API v3 Set Process Information call are no longer available in API v4.

As of eSignatures 6.2 the mandated signing rules are no longer limited to **matchid** or **nameandbirthdate**, but they are configurable and customizable in the Config Index. Now that you can configure custom rules, it is no longer supported to override them with API parameters.

### 3. Resource based calls

The API v4 routes allow a more granular control over the different nodes in each route. They offer more flexibility in how much or how little information about the different parts of a package you pass along in each step of the flow.

In practice, this also removes the distinction between a *package* and an *instant package*. With the new Create package call you can simply create an empty package and add documents, stakeholders, actors, etc. afterwards, like you do in an old Create package call. Or you can define all this in one elaborated call, containing an array of documents, an array of stakeholders, etc., like you would do in an InstantPackage creation call.

## 4. Elements

In API v4 the *element* concept is introduced. An element is an item that is placed on a document and may be assigned to an actor.

A document may contain multiple elements grouped in an Elements array.

Prior to eSignatures 6.3, one type of element is supported: SigningField.

As of eSignatures 6.3, two new element types have been added: TextBoxField and CheckBoxField.

**SigningField** elements determine where on the document a signature must be placed. SigningFields are always mandatory.

**TextBoxField** elements determine where on the document a textbox field will be placed. A textbox may contain a default, prefilled value, which can be modified afterwards by the end user. A textbox may be mandatory or optional.

**CheckBoxField** elements determine where on the document a checkbox field will be placed. A checkbox may be already checked by default, but can still be modified by the end user. Like textboxes, checkboxes may also be mandatory or optional.

In a document, each element type can be added by means of:

- MarkerId
- FieldId
- Combination of a location and size dimensions

## 5. documentOptions object

In API v4, a new object is mandatory in the Create document call: **documentOptions**. Within the documentOptions object the following data are be defined:

- **"contentType": "application/pdf"**

This is the source media type of the uploaded document. The goal is to define the different type of documents that can be uploaded: .txt, .doc, .docx, .pdf and .xml. This to better define the documents that are uploaded.

- **"targetType": "application/pdf"**

The target media type of the to be downloaded document.

- **pdfOptions": {"targetFormat": "pdfa1a"}** These options define specific PDF options, e.g. conversions that must be done.

# API v4 Technical Documentation

This section documents the technical specifications of the Connective eSignatures API version 4.

API v4 uses more resource based routes compared to API v3, which allows a flexible creation and management of the different resources, and an easier implementation of new features.

API v3 is still available for use, but is considered deprecated. This means no new functionality will be added to API v3 and the v3 routes will be removed once v5 will be released.

Although API v3 routes can still be used in combination with v4 routes, it is strongly recommended to use API v4 as starting point of any new integration or update.

**Tip:** the API v4 Technical Documentation is also available as Postman API documentation project on <https://apidocs.connective.eu>.

## Revisions

<u>DATE</u>	<u>OWNER</u>	<u>TOPIC</u>
2020-01-21	DGI	Document creation
2020-03-20	SDE	Updates - CreatePackage <i>SendApproverUrl</i> added for NotificationCallbackDetails - Added <i>f2fSigningUrl</i> to Package Entity - Updated paging return model for GET /packages - Paging has been removed from documents and stakeholders - Added <i>CompletedDate</i> on each element
2020-08-27	DGI	Correction in base URL
2020-09-24	DGI	Added F2fSigningUrl in Create Package response parameters
2020-10-01	DGI	Updated CallbackUrl section + Create stakeholder section
2020-10-09	DGI	Update to version 6.3.0
2020-10-26	DGI	- Update to version 6.3.1 - Removed limitation that PDFA_1A or PDFA_2A could not be set as <b>TargetFormat</b> when using the <b>FieldId</b> parameter to detect form fields. - Added <i>VerifiedName</i> response parameter
2020-10-29	DGI	Correction on Process calls.
2020-11-16	DGI	Replaced px by points (PDF point units)
2020-12-07	DGI	Added RedirectUrl Details to the Create actor and Add Process Step Action calls, and F2FRRedirectUrl Details to the Create package call.



# 1. Introduction

This document describes the technical specifications of the Connective eSignatures API v4.

The eSignatures API is a REST API. It is a web service that allows external systems to make requests for eSignatures sources through URL paths. More concretely, it allows external systems to interact with the eSignatures platform and use its features 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 of 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 were changed to v4 in eSignatures version 6.0. The support period for the API with v3 URLs outlined in the eSignatures version 5.x API Technical Documentation is governed by the Connective eSignatures support Contract.

The API calls with v2 URLs are no longer supported since eSignatures version 6.0.

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.

**Important:** Always use conforming JSON data in the requests. When the Web Application Firewall in front of eSignatures is enabled, the firewall validates all JSON passing through it and does not accept invalid JSON (e.g. which contains embedded comments such as `//` or `/* */` and extra commas).

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 JSON data format and using UTF-8 encoding.

The default URL is:

**`https://[servername]:[port]/webportalapi/v4/`**

**Important:**

When upgrading to eSignatures 6.3.x, you need to add `/esig` to your API endpoint for the API calls, as shown in the default URL

above.

### Important note about MTLS:

When the setting **IsMtlsEnabled** is enabled in the Configuration Index, API users must make 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.

## 1.3 HTTP statuses

Below you find an overview of the HTTP statuses that are returned, and their meaning.

### Successful responses

#### 200 OK

The response body contains a representation of the requested / created resource.

#### 201 Created

The request led to the creation of a resource. The created resource is in the body of the response and its location is found in the Location header.

#### 204 No Content

The request was successful and there is no additional content to send in the response body.

### Client Error responses

#### 400 Bad Request

The server cannot or will not process the request due to something that is perceived to be a client error (e.g. failed validation).

#### 401 Unauthorized

The request has not been applied because it lacks valid authentication credentials for the target resource.

#### 404 Not Found

The origin server did not find a current representation of the target resource.

#### 409 Conflict

The request could not be completed due to a conflict with the current state of the target resource. This code is used in situations where the user might be able to resolve the conflict and resubmit the request.

### Server error responses

#### 500 Internal Server Error

The server encountered an unexpected condition that prevented it from fulfilling the request.

## 2. Concepts Overview

This section explains the different concepts that are used in the eSignatures API.

### **Package**

Packages lie at the core of eSignatures. A package is a container for digital documents that are sent for signing.

A package may contain one or more documents that need to be signed. A package may also have multiple stakeholders who will take actions on the package's documents.

### **Document**

A document is a digital file that is sent for signing. A document is always part of a package and is signed by a stakeholder.

A document may contain multiple elements. Currently, an element is always a signing field. In future versions, multiple element types may be added.

### **Element**

An element is an item that is placed on a document and may be assigned to an actor. For instance, a signing field that can be assigned to a signer actor.

### **Stakeholder**

A stakeholder is an entity that has an interest in a package. A single stakeholder may contain multiple actors, i.e. actions the stakeholders must do. For instance, approve, sign or receive.

A stakeholder may be a single person, or a group of people. When a group of people is defined, any member of the group will be able to approve or sign on behalf of the entire group.

### **Actor**

An actor is a single action that a stakeholder must do on a package.

The different actor types are: approver, signer and receiver.

### **Process and process steps**

A package's process defines which stakeholder will do their action at which moment in time. A process may consist of multiple process steps. This way, you can set up parallel, sequential or complex signing flows. All actors that are put in the same process step will be completed in parallel.

**Tip:** Consult the [Glossary](#) section for an overview of more eSignatures related terms.

## 3. Authentication and Security

### 3.1 Authentication

#### Basic 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 must be changed through configuration at installation time should have been communicated to you by email or sms.

**Note:** depending on the Configuration Index settings, the use of an Mtls Client certificate might be required.

#### Authentication using JWT Access Token

Authentication can be done using a JWT access token following the Oauth 2.0 protocol using the "Client Credentials Grant" flow.

The access token can be retrieved by calling the token endpoint of the authentication server. The Url of the token endpoint can be retrieved from the discovery endpoint available at `/.well-known/openid-configuration`.

The access token should be cached and reused for its lifetime. A new access token should be retrieved using the refresh token when it is expired.

Getting the token with curl:

```
curl -A "ClientApplication/1.0" -d
"grant_type=client_credentials&client_id=${client_id}&client_secret=${client_secret}&scope=${scope}"
"${token_endpoint}"
```

**client\_id** is the value of the client id (also called application id) provided by Connective **client\_secret** is the "password" for the client id, provided by Connective **scope** must contain "<https://connective.eu/scopes/esignatures>" for eSignatures

**token\_endpoint** is the token endpoint on the authentication server

The response is a json document of the form:

```
{
  "access_token": "eyJhb.....A",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhb.....A",
  "token_type": "bearer",
  "not-before-policy": 1591634674,
  "session_state": "0f65368c-90df-4e2f-b039-5320a0b34871",
  "scope": "email https://connective.eu/scopes/esignatures profile"
}
```

When calling the API, the access token must be placed in an Authorization HTTP Header of type "Bearer". See <https://tools.ietf.org/html/rfc6750>

**NB:** a valid User Agent string is required for calls. The default user agent for curl or wget can be blocked by the Web Application Firewall protecting the application.

### 3.2 Security through one-time URLs

The URL lifetime is by default limited to one-time use (unless the administrator disabled the **IsOneTimeUrlEnabled** setting in the Config Index, which is not recommended). It is therefore recommended to request the stakeholder's actor links just before redirecting them. This way, you make sure they receive the latest generated link.

When end users open an action URL they already used, they will either see a warning or be redirected to a given URL (if available). The only way to achieve this is to request a new URL and provide it to the user. You can do this by retrieving the Actor information via any of the API status calls.

When end users try to take action on a document that has been deleted, a 404 Not found error will occur. This cannot be recovered from, since all information has been purged from eSignatures.

## 4. Error handling responses

eSignatures API v4 uses HTTP error codes to give an idea of whether a call succeeded or failed, and a system of error codes in the response body to give more information in case 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 explained in [Error code descriptions](#).

The error responses are JSON containing the following list of objects:

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Errors (array of objects)</b>	List of the errors	Array
<b>ErrorCode</b>	Error code with variable information	String
<b>Message</b>	Error message detail text, not localized	String

An eSignatures error code is always displayed as follows:

```
Resource.Error:parameter
```

E.g. Package.NotFound:00000000-0000-0000-0000-000000000000. The GUID behind the colon is the packageId that wasn't found.

Error codes can be reused: if the HTTP response code is HTTP 400 Bad Request, an error code like "Document.InvalidTargetFormat" indicates a value that is not supported by eSignatures. If the HTTP response code is HTTP 409 Conflict, an error code like "Document.InvalidTargetFormat" 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. Packages

The **Packages** collection contains all resources and methods for creating and managing packages and package data.

### What is a package?

Packages lie at the core of eSignatures. A package is a container for digital documents that are sent for signing.

A package may contain one or more documents that need to be signed. A package may also have multiple stakeholders who will take actions on the package's documents.

### Package methods

- Create package
- Update package expiry date
- Update package status
- Get packages
- Get package by ID
- Get package expiry date
- Get package status
- Download package
- Delete package by ID
- Get package warnings
- Send reminders



# Create package

## Description

The POST package call creates an empty package, to which documents can be added later on.

In its simplest form, this method contains only two request parameters: the package **name** and the **initiator**, which is the user who will be sending the packages. Afterwards you can build on this method by adding additional resources. You can add documents, elements, stakeholders, actors, etc. by using their respective calls.

You can also choose to define all parameters in a single POST package call, making it a so-called "super call".

## Size limitations

Consider the following size limitations when creating packages.

- 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.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages

## HTTP Method

POST

## MIME Type

application/json

## Request parameters

The parameters with an asterisk are mandatory. All others are optional.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Name*	<p>Name of the package.</p> <p>The name is displayed in the eSignatures WebPortal, and used as file name when you download the package as .zip file from the WebPortal.</p> <p><b>Note:</b> do not add an extension to the Name value <b>Important:</b> Pay attention when choosing a package name. Don't use forbidden file name characters such as slash (/), backslash (), question mark (?), percent (%), asterisk (), colon (:), pipe ( ), quote ('), double quote ("), less than (&lt;), greater than (&gt;). <i>Note however, that is list is not exhaustive.</i> <i>Don't use characters that are HTML-sensitive such as ampersand (&amp;) or apostrophe (').</i></p> <p><i>Note*:</i> when using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.</p> <p>Minimum length: 1</p> <p>Maximum length: 150</p>	String (GUID)
Initiator*	<p>The email address of the user who will send the package.</p> <p>The user must be a known user in the eSignatures WebPortal.</p>	String (email)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Status	<p>The status the package will be in when created. The possible values are: draft or pending.</p> <p>By default, a package is in draft after creation.</p>	String (enum)
DocumentGroupCode	<p>The documentGroupCode is the identifier of the document group to which the package must be uploaded.</p> <p>In a default configuration, an initiator uploads packages to their personal MyDocuments folder, which cannot be shared with others. To enable collaboration however, an eSignatures admin can configure multiple document groups to which different users may have access. This way, they can collaborate on one another's' documents.</p> <p>To upload packages to the MyDocuments folder, do not use this parameter, or specify value 00001.</p> <p>To upload packages to a specific document group, enter its documentGroupCode.</p> <p><b>Tip:</b> to know which document groups have been configured in eSignatures, use the Get document groups call in API v3.</p>	String
ExpiryDate	<p>Date and time when this package expires and can no longer be approved/signed. Documents within the package expire on the same date.</p> <p>Format is ISO 8601 date-time. E.g. 2018-01-23T12:34:00.000Z</p>	String (date-time)
Documents	<p>Documents in base64 encoded format that must be added to the package.</p> <p>The request parameters regarding documents are identical to those of the <a href="#">Add document</a> call and are described there.</p>	Array of objects
Stakeholders	<p>Stakeholders that must be added to the package.</p> <p>A stakeholder is an object that provides information about any person who is involved with the package.</p> <p>The request parameters regarding stakeholders are identical to those of the <a href="#">Create stakeholder</a> call and are described there.</p>	Array of objects
DefaultLegalNotice	<p>The default legal notice that will be added to a signer when no legal notice is specified.</p> <p>See the section <b>defaultLegalNotice</b> below for more info.</p>	Object
ThemeCode	<p>Defines which themeCode must be applied to the package.</p> <p>An eSignatures admin can rebrand the look and feel of the eSignatures WebPortal by creating and configuring new themes. Each theme has a themeCode to identify it.</p> <p>When this parameter is not used, the theme that has been configured on environment level in the Configuration Index, or on document group level if applicable will be applied.</p>	String
CallbackUrl	<p>The <b>Callback URL</b> 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.</p> <p>See <b>Callback URL details</b> below.</p>	String (url)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
NotificationCallBackUrl	<p>URL that will be called each time the signer requests a new signing URL.</p> <p>Since eSignatures by default uses one-time URLs, the link to sign a package only works once. As soon as it has been clicked, the link expires and a new signing link must be requested.</p>	String (url)
F2fRedirectUrl	<p>URL to which the end user is redirected after all fields have been signed or rejected face to face in the Document Portal. See the <b>F2fRedirectUrl Details</b> section below for more information.</p> <p><b>Attention:</b> Don't confuse the f2fRedirectUrl with the regular RedirectUrl. The f2fRedirectUrl only applies when signing face to face in the Document Portal. The redirect occurs after signing or rejecting. This field must be a valid absolute url.</p> <p><b>Note:</b> 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 F2FRedirectUrl is configured, the Close button will be unavailable, and a message is displayed informing the signers they will be redirected.</p>	String (url)
IsUnsignedContentDownloadable	<p>Determines whether an actor can download the package from the WYSIWYS page (What You See Is What You Sign), before approving, signing or rejecting.</p> <p>Enter 'true' if you want actors to be able to download the package before approving/signing. This way they can print it and read it on paper for instance.</p> <p>Enter 'false' to hide the download icon and prevent actors to be able to download packages from the WYSIWYS.</p> <p>When this parameter is not used, the value from the Config Index setting <b>IsDownloadUnsignedFilesEnabled</b> under <b>Customization Settings</b> is used.</p>	Boolean
IsReassignEnabled	<p>Determines whether a stakeholder may reassign their action to another party.</p> <p>Enter 'true' if you want actors to be able to reassign the package.</p> <p>Enter 'false' to hide the reassign button and prevent actors to be able to reassign packages from the WYSIWYS.</p> <p>When no value is entered, this parameter takes its value from the Config Index setting <b>IsReassignEnabled</b> under <b>Customization Settings</b>.</p>	Boolean
ExternalReference	<p>External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.</p> <p>Maximum length: 256</p>	String
ActionUrlExpirationPeriodInDays	<p>This parameter determines after how many days the action URLs must expire when they are not used. When no value is entered, this parameter takes its value from the Config Index setting <b>IsActionUrlExpirationEnabled</b> under <b>Customization Settings</b>.</p>	Integer
ProofCorrelationId	<p>Identifier to correlate this package to other resources in the proofs system.</p>	String (GUID)

## DefaultLegalNotice

The defaultLegalNotice object defines the default legal notice that will be applied to the signer when no legal notice is specified.

A legal notice is a text the signer must retype before they are able to place their signature. Note that legal notices are case-sensitive\*.

\*When using legal notices in which the signer must enter variable values, these values are not case-sensitive.

When defining a defaultLegalNotice, you can choose from two defaultLegalNotice types: **name** or **text**. The maximum length of the text value is in total 255 characters (including fill-in values).

PARAMETER	CONTENT / DESCRIPTION	TYPE
Text	Use the text parameter to use a custom legal notice, i.e. one that is not configured in the Config Index.  The legal notice should be written in the same language as the documents of the package.  Maximum length: 255	String
Name	The name of the legal notice that will be added to the signing field. To correctly use this parameter, you need to know the name of the different legal notices that are configured in the Config Index. E.g. LEGALNOTICE1. The value of the legal notices is also set in the Config Index. Note that the language in which the legal notice is displayed depends on the language of the document.  Maximum length: 20	String (enum)

## Instant Package

API v4 allows to mimic the 'Instant Package' functionality available within API v3. This is possible by combining all different objects within one call

- Create Package
- Add document(s)
- Add element(s)
- Add stakeholder(s)
- Add actor(s)
- Add process

This does bring one additional concept to take into account when adding multiple documents, as you will need to define for which document you are adding the stakeholder(s) / actor(s). As at this point you have not yet received the documentId, the concept of **documentIndex** has been introduced on element level. The documentIndex is being used to identify the document for which the actor is being added, 0 being the first document in the call, 1 the second and so on.

## Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Id	Unique identifier of the package.	String (GUID)
Name	Name of the package.  Minimum length: 1  Maximum length: 150	String
Status	Status of the package.  Possible values are: draft and pending.	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
CreationDate	Date and time when the package was created according to the server.  Format is ISO 8601 date-time. E.g. 2020-01-23T12:34:00.000Z	String (date-time)
ExpiryDate	Date and time when the package will expire.  Format is ISO 8601 date-time. E.g. 2020-01-25T13:00:00.000Z	String (date-time)
Initiator	Email address of the user who sent the package. This must be a known user of the WebPortal.	String (email)
Documents	The documents that were added to the package.  The response parameters regarding documents are identical to those of the <a href="#">Add document</a> call and are described there.	Array of objects
Stakeholders	The stakeholders that were added to the package.  The response parameters regarding stakeholders are identical to those of the <a href="#">Create stakeholder</a> call and are described there.	Array of objects
DefaultLegalNotice	Default legal notice that has been added to the signer.	Object
ExternalReference	External reference to this resource given by a calling application.  This parameter is not used by eSignatures itself.  Maximum length: 256	String
DocumentGroupCode	Identifier of the document group to which the package will be added.	String
ThemeCode	Identifier of the themeCode that has been applied to this package.	String
CallbackUrl	URL that will be called upon each status change. See <b>CallbackUrl Details</b> below for more info.	String (url)
NotificationCallbackUrl	URL that will be called each time the signer requests a new signing URL	String (url)
F2fSiningUrl	Link to the package which allows to start a face to face signing session.	String (url)
F2fRedirectUrl	URL to which the end user is redirected when all fields have been signed, or when a field has been rejected. <b>Attention:</b> not to be confused with a "regular" redirectUrl.	String (url)
IsUnsignedContentDownloadable	Determines whether an actor can download the package from the WYSIWYS page (What You See Is What You Sign), before approving, signing or rejecting.	Boolean
ActionUrlExpirationPeriodInDays	This parameter determines after how many days the action URLs must expire when they are not used. When no value is entered, this parameter takes its value from the Config Index setting IsActionUrlExpirationEnabled under Customization Settings.	Integer

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ProofCorrelationId	Identifier to correlate this package to other resources in the proofs system.	String
Warnings	Warning about the package, e.g. missing data.	Object

## DefaultLegalNotice

The defaultLegalNotice object defines the default legal notice that is applied to the signing field.

A legal notice is a text the signer must retype before they are able to place their signature. Note that legal notices are case-sensitive\*.

\*When using legal notices in which the signer must enter variable values, these values are not case-sensitive.

There are two defaultLegalNotice types: **name** or **text**.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Text	Text of the legal notice.	String

Warning schema:

<b>PROPERTY</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
ResourceType	The subject of the warning e.g. "process"	String
Code	The warning code, Follows the format of an eSignatures errorCode.	String
Message	A human-readable explanation of the warning.	String

When resourceType is "process" these properties are added:

<b>PROPERTY</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
Stakeholder	The stakeholder involved	Stakeholder
Actor	The actor involved	Actor

Stakeholder schema:

<b>PARAMETER</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
Id	The stakeholder's id	String (GUID)
ExternalReference	The stakeholder's external reference	String

Actor schema:

<b>PARAMETER</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
Id	The actor's id	String (GUID)

## Example request

```
{
  "Name": "example pending package",
  "Initiator": "hello@world.test",
  "Status": "pending",
  "ExpiryDate": "2020-01-17T12:33:47.923Z",
  "DefaultLegalNotice": {
    "Name": "LegalNotice1"
  },
  "Documents": [
    {
      "Name": "my first document",
      "Language": "en",
      "IsOptional": false,
      "ExternalReference": "doc0",
      "DocumentOptions": {
        "TargetType": "application/pdf",
        "PdfOptions": {
          "TargetFormat": "pdfa1a"
        }
      },
      "Base64data": "string",
      "ContentType": "application/pdf"
    },
    {
      "Name": "my second document",
      "Language": "en",
      "IsOptional": false,
      "ExternalReference": "doc1",
      "DocumentOptions": {
        "TargetType": "application/pdf",
        "Base64data": "string",
        "ContentType": "application/pdf"
      }
    }
  ],
  "Stakeholders": [
    {
      "Type": "person",
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16",
      "PhoneNumber": "+32xxxxxxxxx",
      "Actors": [
        {
          "Type": "signer",
          "ProcessStep": 0,
          "Elements": [
            {
              "Type": "signingField",
              "DocumentIndex": 0,
              "Location": {
                "Page": 2,
                "Top": 200,
                "Left": 200
              },
              "Dimensions": {
                "Width": 200,
                "Height": 200
              }
            }
          ]
        }
      ]
    }
  ]
}
```

```

    }
  ],
},
{
  "Type": "group",
  "GroupName": "Great mates",
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
    }
  ],
  "Actors": [
    {
      "Type": "signer",
      "ProcessStep": 1,
      "Elements": [
        {
          "Type": "signingField",
          "DocumentIndex": 1,
          "Location": {
            "Page": 2,
            "Top": 200,
            "Left": 200
          },
          "Dimensions": {
            "Width": 200,
            "Height": 200
          }
        }
      ]
    }
  ]
}
],
"ThemeCode": "string",
"CallBackUrl": "https://callback.test",
"NotificationCallBackUrl": "https://notifications.test",
"IsUnsignedContentDownloadable": true,
"ExternalReference": "myFirstPackage"
}

```

### Example response

```

{
  "Id": "CD758AE3-5382-48B2-A4F3-DD76C7200EE0",
  "Name": "example pending package",
  "Initiator": "hello@world.test",
  "Status": "pending",
  "ExpiryDate": "2020-01-17T12:33:47.923Z",
  "DefaultLegalNotice": {
    "Name": "LegalNotice1"
  },
  "Documents": [
    {
      "Id": "1F501B90-9E0E-4E12-9A09-6C737C6391E8"
    }
  ]
}

```



```

    "Id": "1E991B55-5101-4112-9AB3-0C737C0334E8",
    "Name": "my first document",
    "Language": "en",
    "IsOptional": false,
    "ExternalReference": "doc0",
    "DocumentOptions": {
      "TargetType": "application/pdf",
      "PdfOptions": {
        "TargetFormat": "pdfa1a"
      },
      "Base64data": "string",
      "ContentType": "application/pdf"
    }
  },
  {
    "Id": "1BAE1A85-BF29-426C-A9B6-D7BD330CF4CD",
    "Name": "my second document",
    "Language": "en",
    "IsOptional": false,
    "ExternalReference": "doc1",
    "DocumentOptions": {
      "TargetType": "application/pdf",
      "Base64data": "string",
      "ContentType": "application/pdf"
    }
  }
],
"stakeholders": [
  {
    "Id": "C2CEBEEF-A9BB-4BF8-9FD9-7F78EBC80CC2",
    "Type": "person",
    "Language": "en",
    "FirstName": "John",
    "LastName": "Doe",
    "EmailAddress": "john@doe.test",
    "BirthDate": "1990-01-16",
    "PhoneNumber": "+32xxxxxxxx",
    "Actors": [
      {
        "Id": "5D197888-8660-4374-BB42-102A4F6E35C4",
        "Type": "signer",
        "ProcessStep": 0,
        "Elements": [
          {
            "Id": "F77A4B19-ADEB-495B-A098-1E99DB0E42FD",
            "Type": "signingField",
            "DocumentIndex": 0,
            "Location": {
              "Page": 2,
              "Top": 200,
              "Left": 200
            },
            "Dimensions": {
              "Width": 200,
              "Height": 200
            }
          }
        ]
      }
    ]
  }
]
},
{
  "Id": "57274DC0-2B9F-4A1E-B9FD-CFD1C1893683",
  "Type": "group",
  "GroupName": "Great mates",
  "Members": [

```

```

{
  "Id": "750FC8F4-00D0-4247-AA5A-9323A1E1F740",
  "Language": "en",
  "FirstName": "John",
  "LastName": "Doe",
  "EmailAddress": "john@doe.test",
  "BirthDate": "1990-01-16"
},
{
  "Id": "BE89E158-4278-41E7-BF33-7EE6F8B2ECFB",
  "Language": "en",
  "FirstName": "Zu",
  "LastName": "Li",
  "EmailAddress": "zu@li.test",
  "BirthDate": "1991-03-14"
}
],
"Actors": [
  {
    "Id": "F62C2D0D-42E2-48E9-BA4B-C9825290C474",
    "Type": "signer",
    "ProcessStep": 1,
    "Elements": [
      {
        "Id": "E62DB347-5ED1-48DA-91A3-90A660AA18CC",
        "Type": "signingField",
        "DocumentIndex": 1,
        "Location": {
          "page": 2,
          "top": 200,
          "left": 200
        },
        "Dimensions": {
          "width": 200,
          "height": 200
        }
      }
    ]
  }
]
}
],
"ThemeCode": "string",
"CallbackUrl": "https://callback.test",
"NotificationCallbackUrl": "https://notifications.test",
"IsUnsignedContentDownloadable": true,
"ExternalReference": "myFirstPackage"
}

```

Response codes

RESPONSE STATUS CODE	DESCRIPTION
201 Created	The package was created successfully.
400 Bad request	Request validation failed. The package could not be created due to invalid parameters in the request.
401 Unauthorized	Caller is unauthorized to perform this operation.
409 Conflict	The package could not be created.

Error codes

HTTP CODE	CODE
400	Request.RequiredFieldsMissing
400	Request.FieldMaxLength
400	Package.InitiatorInvalid
400	Url.Invalid
400	Request.UnsupportedValue
400	Document.UnsupportedLanguage
400	Base64Data.InvalidLength
400	Document.InvalidTargetFileType
400	PdfErrorHandling.InvalidType
400	Document.DataInvalid
400	Request.OneOfFieldsMissing
400	Request.FieldMaxLength
400	SigningType.Invalid
400	Request.FieldMinimumValue
400	SigningField.InvalidWidthCoordinate
400	SigningField.InvalidHeightCoordinate
400	Stakeholder.UnsupportedLanguage
400	Stakeholder.EmailAddressInvalid
400	Stakeholder.BirthDayInvalid
400	Stakeholder.BirthDayInFuture
400	Stakeholder.InvalidPhoneNumber
400	GroupMember.RequiredFieldsMissing
400	GroupMember.UnsupportedLanguage
400	GroupMember.EmailAddressInvalid
400	Stakeholder.BirthDayInvalid

HTTP CODE	CODE
400	Stakeholder.BirthDayInFuture
400	Stakeholder.InvalidPhoneNumber
400	GroupMember.RequiredFieldsMissing
400	GroupMember.UnsupportedLanguage
400	GroupMember.EmailAddressInvalid
400	GroupMember.BirthDayInvalid
400	GroupMember.BirthDayInFuture
400	GroupMember.InvalidPhoneNumber
400	Request.OneOfFieldsIsMissing
400	SigningType.Invalid
400	Request.FieldMinimumValue
400	SigningField.InvalidWidthCoordinate
400	SigningField.InvalidHeightCoordinate
400	LegalNotice.CodeAndTextUsedSimultaneously
401	Authentication failed, username or password incorrect
404	DocumentGroup.NotFoundWithCode
404	Theme.NotFoundWithCode
404	User.NotFound
409	Package.InvalidStatus
409	Package.ApiVersionMismatch
409	User.NotFound
409	Document.CorrelationIdAlreadyExists
409	Document.Xml.UploadNotWellFormed
409	Package.PackageAcceptsDocumentType
409	Document.Xml.AlreadySignedNotAllowed

HTTP CODE	CODE
409	ContactGroup.NotFound
409	Package.ContainsEmptyContactGroup
409	Stakeholder.TypeInvalid
409	Document.DataMissing
404	Actor.NotFound
409	SigningField.NotFound
409	Package.InvalidStatus
409	ProcessStep.MixedTypes
404	Stakeholder.NotFound

### CallbackUrl 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.

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.

The POST request by default uses an application/json body containing:

- packageId
- packageStatus

E.g. {"packageId":"d2912916-c555-47af-b666-1e8cf2eb9dfa","packageStatus":"FullySigned"}

If the external system requires extra information, then a Get package by ID call can be used.

**Important:** If you are upgrading from eSignatures 5.x, and you want to append the packageId and packageStatus to the base URL, as was the standard behavior in eSignatures prior to version 6.0, make sure to select the **legacy** or **both** option under **API Settings** in the Config Index. Note however that there is a difference in casing between the two eSignatures versions. As of version 6.0, camel casing is used. If your integration is case-sensitive, it will fail on the callback verification, since it expects the parameters "PackageId" and "PackageStatus" in the CallbackURL instead of "packageId" and "packageStatus". This casing issue has been solved in eSignatures 6.4.1.

**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.

**Note:** in case a Callback error should occur, eSignatures by default retries to do the callback 3 times, during 3 retry cycles. System administrators may customize this retry mechanism in the Worker config file. See **Connective - eSignatures 6.3.x - Installation Documentation - Limited Public – OP** for more information.

### 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.

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:

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Packageld</b>	Unique identifier of the package	String (GUID)
<b>ActorId</b>	Identifier of the actor for which the notification was triggered	String (GUID)
<b>Language</b>	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
<b>NotificationType</b>	Which kind of notification was requested	String

A callback may currently happen in the following cases:

<u>NOTIFICATIONTYPE</u>	<u>USE CASE</u>
<b>SendApproverUrl</b>	The single-use approver link has been used already and the end user requested a new link.
<b>SendSignerUrl</b>	The single-use signing link has been used already and the end user requested a new link.
<b>SendDownloadUrl</b>	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 SuppressNotifications parameter which is separate from the NotificationCallbackUrl.

**Note:** in case a Callback error should occur, eSignatures now by default retries to do the callback 3 times, during 3 retry cycles. System administrators may customize this retry mechanism in the Worker config file. See **Connective - eSignatures 6.3.x - Installation Documentation - Limited Public – OP** for more information.

#### F2FRedirectUrl Details

The **F2FRedirectUrl** is used to redirect the end user after all fields have been signed or rejected **face to face** in the Document Portal.

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 Create Package call and is appended with the following query parameters:

- SessionID (unique identifier of the package signing session, i.e. Packageld)
- ExternalReference (as found in the ExternalReference parameter of the Create Stakeholder call)
- Status (SIGNED, REJECTED or INVALIDTOKEN)
- PackageExternalReference (as found in the **ExternalPackageReference** parameter of the Create Package call)

The **F2FRedirectUrl** 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 F2FRedirectUrl will be:

[https://myserver.example.org/rental/services/testredirect?\\*\\*SessionID\\*\\*=5a2aff04-3cfa-4278-9480-64ac39f74734&\\*\\*ExternalReference\\*\\*=user1@example.org&\\*\\*Status\\*\\*=REJECTED&\\*\\*PackageExternalReference\\*\\*=dossier-3592](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 F2FRedirectUrl because the redirection happens *client-side*! Either establish a second secure channel by means of the CallbackUrl, or verify through session state that the end user returned from the correct session id. Afterwards a call to Get Package Status 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 F2FRedirectUrl 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.

# Update package expiry date

## Description

This call allows you to update the expiry date of a package. This way, you can make a document that was expired available again.

The only template parameter you need is `packageld`.

## URL

`https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/expirydate`

## HTTP Method

PUT

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package whose expiry date you want to update.	String (GUID)

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ExpiryDate	Date and time when this package expires and can no longer be approved/signed. Documents within the package expire on the same date.  Format is ISO 8601 date-time.	String (date-time)

## Example request

```
"2020-01-20T21:29:21.687Z"
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Package expiry date was successfully updated.
<b>400 Bad request</b>	The package could not be updated due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	The package could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound



# Update package status

## Description

This call allows you to update the status of a package.

Possible combinations:

- Draft to pending
- Pending to revoked

### Draft to pending

When the package is ready to be sent to the stakeholders to take action, change the status to 'pending'. This will make the package visible to each of the stakeholders in their Signer Portal.

### Pending to revoked

Packages that have been sent to stakeholders, but haven't been approved/signed yet, or packages of which the signing has failed can be revoked. You can do so by updating their status to 'revoked'. The stakeholders will no longer be able to take action on their package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/status

## HTTP Method

PUT

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose status you want to retrieve.	String (GUID)

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Status	Status to which the package must be updated. The possible values are: pending and revoked.	String (enum)

## Example request

```
pending
```

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Status	The possible values are: pending, revoked.	String (enum)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Package status was successfully updated.

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>400 Bad request</b>	The package could not be updated due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	The package could not be found.
<b>409 Conflict</b>	Package status could not be updated.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

# Get packages

## Description

This call retrieves all packages that currently exist within eSignatures, and their current status.

Packages are always sorted on creationDate.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages](https://[servername]:[port]/esig/webportalapi/v4/packages)

## Query parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PageNumber	The page to retrieve.  Default value is 0.  Example: when you set the PageNumber to 0 and the PageSize is 20, the first 20 packages are retrieved. When you set the PageNumber to 1 and the PageSize is 20, the next 20 packages are retrieved.  Note: when the value you pass is higher than the actual number of pages, an empty list will be returned.	Integer
PageSize	Maximum number of records in the response  Default value is 20.	Integer
CreatedBefore	Gets only the packages created before this date. ISO 8601 date format.	String (date-time)
CreatedAfter	Gets only the packages created after this date. ISO 8601 date format.	String (date-time)
Status	Gets items that have a specific status. Supported status values are: draft, pending, inProgress, ending, finished, rejected, revoked, expired, failed.	String (enum)
DocumentGroupCode	Gets items that have given documentGroupCode. An admin may configure document groups in eSignatures to which users can upload their documents. The documentGroupCode is the identifier of the document group to which the package has been uploaded.	String
Sort	Determines how the result is sorted. Available values are asc, desc. Default value: desc	String (enum)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PageSize	Maximum number of records in the response  Default value is 20.	Integer



```
"PackageId": "00000000-0000-0000-0000-000000000000",
"Type": "person",
"Actors": [
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "signer",
    "Status": "available",
    "Links": [
      "https://dothething.test"
    ]
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "signer",
    "Status": "unavailable"
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "signer",
    "Status": "skipped"
  }
],
"ExternalReference": "string",
"Language": "en",
"FirstName": "string",
"LastName": "string",
"EmailAddress": "hello@world.test",
"BirthDate": "2020-01-22"
},
{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Type": "group",
  "Actors": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "Type": "signer",
      "Status": "available",
      "MemberLinks": [
        {
          "Email": "zu@li.test",
          "Link": "https://dothething.test"
        },
        {
          "Email": "john@doe.test",
          "Link": "https://alsocompletelyouraction.test"
        }
      ]
    }
  ]
},
"GroupName": "Group123",
"Members": [
  {
    "Language": "en",
    "FirstName": "John",
    "LastName": "Doe",
    "EmailAddress": "john@doe.test",
    "BirthDate": "1990-01-16"
  },
  {
    "Language": "en",
    "FirstName": "Zu",
    "LastName": "Li",
    "EmailAddress": "zu@li.test",
    "BirthDate": "1991-03-14"
  }
]
```

```

    ],
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "contactGroup",
    "ContactGroupCode": "00002",
    "Actors": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "available",
        "MemberLinks": [
          {
            "Email": "john@doe.test",
            "Link": "https://alsocompleteyouraction.test"
          },
          {
            "Email": "zu@li.test",
            "Link": "https://dothething.test"
          }
        ]
      }
    ]
  },
  "ExternalReference": "string",
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
    }
  ]
},
"DefaultLegalNotice": {
  "Text": "Read and approved"
},
"ExternalReference": "string",
"DocumentGroupCode": "string",
"ThemeCode": "string",
"CallBackUrl": "string",
"NotificationCallBackUrl": "string",
"F2fSigningUrl": "string",
"F2fRedirectUrl": "string",
"IsUnsignedContentDownloadable": false,
"ActionUrlExpirationPeriodInDays": 0,
"ProofCorrelationId": "string",
"Warnings": [
  {
    "ResourceType": "process",
    "Code": "ContactGroup.SomeMissingPhoneNumbers",
    "Message": "Some of the group members don't have a phone number required for [SmsOTP] signing
type.",
    "Stakeholder": {
      "Id": "82c6b4f4-70fe-4fff-b176-6df0f4e79293",
      "ExternalReference": "userXYZ"
    }
  }
]

```

```
    },
    "Actor": {
      "Id": "73462bf5-8430-411d-befb-b35c92f35e10"
    }
  }
]
}
]
```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All packages are successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect

# Get package by ID

## Description

This call allows you to retrieve a package by its ID.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	The unique identifier of the package you want to retrieve.	String (GUID)

## Response parameters

The Response parameter are identical to the ones of the [Create package](#) call and are described there.

## Example response

```
{
  "Id": "00000000-0000-0000-0000-000000000000",
  "Name": "example package",
  "Status": "draft",
  "CreationDate": "2020-08-05T13:16:08.472Z",
  "ExpiryDate": "2020-08-05T13:16:08.472Z",
  "Initiator": "hello@world.test",
  "Documents": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "PackageId": "00000000-0000-0000-0000-000000000000",
      "Name": "string",
      "CreationDate": "2020-08-05T13:16:08.472Z",
      "MediaType": "application/pdf",
      "Language": "en",
      "IsOptional": false,
      "Status": "draft",
      "Elements": [
        {
          "Id": "00000000-0000-0000-0000-000000000000",
          "ActorId": "00000000-0000-0000-0000-000000000000",
          "Type": "SigningField",
          "Location": {
            "Page": 2,
            "Top": 200,
            "Left": 200
          },
          "Dimensions": {
            "Width": 200,
            "Height": 200
          },
          "Status": "pending",
          "ExternalReference": "string",
          "CompletedDate": "string",
          "UsedSigningMethod": "string",
          "SigningMethods": [
            "manual"
          ],
          "LegalNotice": {
            "Text": "Read and approved"
          }
        }
      ]
    }
  ]
}
```



```

    ],
    "ExternalReference": "string"
  }
],
"Stakeholders": [
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "person",
    "Actors": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "available",
        "Links": [
          "https://dothething.test"
        ]
      },
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "unavailable"
      },
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "skipped"
      }
    ],
    "ExternalReference": "string",
    "Language": "en",
    "FirstName": "string",
    "LastName": "string",
    "EmailAddress": "hello@world.test",
    "BirthDate": "2020-01-22"
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "group",
    "Actors": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "available",
        "MemberLinks": [
          {
            "Email": "zu@li.test",
            "Link": "https://dothething.test"
          },
          {
            "Email": "john@doe.test",
            "Link": "https://alsocompleteyouraction.test"
          }
        ]
      }
    ],
    "GroupName": "Group123",
    "Members": [
      {
        "Language": "en",
        "FirstName": "John",
        "LastName": "Doe",
        "EmailAddress": "john@doe.test",
        "BirthDate": "1990-01-16"
      },

```

```

    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
    }
  ]
},
{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Type": "ContactGroup",
  "ContactGroupCode": "00002",
  "Actors": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "Type": "signer",
      "Status": "available",
      "MemberLinks": [
        {
          "Email": "john@doe.test",
          "Link": "https://alsocompleteyouraction.test"
        },
        {
          "Email": "zu@li.test",
          "Link": "https://dothething.test"
        }
      ]
    }
  ],
  "ExternalReference": "string",
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
    }
  ]
}
],
"DefaultLegalNotice": {
  "Text": "Read and approved"
},
"ExternalReference": "string",
"DocumentGroupCode": "string",
"ThemeCode": "string",
"CallbackUrl": "string",
"NotificationCallbackUrl": "string",
"F2fSigningUrl": "string",
"F2fRedirectUrl": "string",
"IsUnsignedContentDownloadable": false,
"ActionUrlExpirationPeriodInDays": 0,
"ProofCorrelationId": "string",
"Warnings": [
  {

```

```
    "ResourceType": "process",
    "Code": "ContactGroup.SomeMissingPhoneNumbers",
    "Message": "Some of the group members don't have a phone number required for [SmsOTP] signing type.",
    "Stakeholder": {
      "Id": "82c6b4f4-70fe-4fff-b176-6df0f4e79293",
      "ExternalReference": "userXYZ"
    },
    "Actor": {
      "Id": "73462bf5-8430-411d-befb-b35c92f35e10"
    }
  }
]
```

Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	Package is successfully retrieved.
404 Not found	Package could not be found.

Error codes

HTTP CODE	CODE
401	Authentication failed, username or password incorrect
404	Package.NotFound

# Get package expiry date

## Description

This method allows you to retrieve the expiry date of a package.

The only template parameter you need to use is packageId.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/expirydate

## HTTP Method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose expiry date you want to retrieve.	String (GUID)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ExpiryDate	Date and time when this package expires and can no longer be approved/signed. Documents within the package expire on the same date.  Format is ISO 8601 date-time.	String (date-time)

## Example response

2020-01-20T21:21:07.981Z
--------------------------

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Package expiry date was successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	The package could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound

# Get package status

## Description

This method allows you to retrieve the status of a package.

The only template parameter you need to use is packageld.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/status

## HTTP Method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package whose status you want to retrieve.	String (GUID)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Status	The possible values are: draft, pending, inProgress, ending, finished, rejected, revoked, expired, failed.	String (enum)

## Example response

draft
-------

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	Status was successfully retrieved.
401 Unauthorized	The caller is unauthorized to perform this operation.
404 Not found	The package could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
401	Authentication failed, username or password incorrect
404	Package.NotFound

## Download package

### Description

This call retrieves all documents of a specified package as a .zip file.

Note that only fully signed packages can be downloaded.

### URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageId}download](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}download)

### Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	The unique identifier of the package whose documents you want to download as .zip file.	String (GUID)

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The package's contents are successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Specified package could not be found.
<b>409 Conflict</b>	The specified package has not been fully signed.

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

# Delete package by ID

## Description

This call allows you to delete a package from the database.

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, you can use the DELETE packages by ID call.

**Note:** you can only delete a package that has status **draft** or one of the final states: **finished**, **rejected** or **revoked**.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}

## HTTP Method

Delete

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package that must be deleted.	String (GUID)

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
DelayedDeletionTime	Time expressed in number of days after which the audit proofs of this package will be deleted, once the package has been deleted. When this parameter is not passed, the value set under <b>Delayed Deletion Time</b> in the Configuration Index will be applied. <b>Tip:</b> to keep audit proofs indefinitely until a manual Delete audit proofs call is done, enter '0' as value.	Integer

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Package was successfully deleted.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.
<b>409 Conflict</b>	Package could not be deleted.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

# Get package warnings

## Description

This call retrieves all warnings there might be about a package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/warnings

## HTTP method

GET

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package whose warnings you want to retrieve.	String (GUID)

## Response parameters

The response is a list of warnings.

Warning schema:

PROPERTY	CONTENT / DESCRIPTION	TYPE
ResourceType	The subject of the warning e.g. "process"	String
Code	The warning code, Follows the format of an eSignatures errorCode.	String
Message	A human-readable explanation of the warning.	String

When resourceType is "process" these properties are added:

PROPERTY	CONTENT / DESCRIPTION	TYPE
Stakeholder	The stakeholder involved	Stakeholder
Actor	The actor involved	Actor

Stakeholder schema:

PARAMETER	CONTENT / DESCRIPTION	TYPE
Id	The stakeholder's id	String (GUID)
ExternalReference	The stakeholder's external reference	String

Actor schema:

PARAMETER	CONTENT / DESCRIPTION	TYPE
Id	The actor's id	String (GUID)



Example response

```
[
  {
    "ResourceType": "process",
    "Code": "ContactGroup.SomeMissingPhoneNumbers",
    "Message": "Some of the group members don't have a phone number required for [SmsOTP] signing type.",
    "Stakeholder": {
      "Id": "82c6b4f4-70fe-4fff-b176-6df0f4e79293",
      "ExternalReference": "userXYZ"
    },
    "Actor": {
      "Id": "73462bf5-8430-411d-befb-b35c92f35e10"
    }
  }
]
```

Error codes

HTTP CODE	CODE
401	Authentication failed, username or password incorrect
404	Package.NotFound

# Send reminders

## Description

The POST reminders call sends a reminder to the actors of the current process step.

Company policy might require that a document is handled within a given timespan. Doing a POST reminders call looks up all actors within the current process step who haven't approved/signed their document and sends them an extra notification as a reminder.

Note that only the next available approver(s)/signer(s) in the workflow are notified. This means approvers/signers waiting for someone else to approve/sign first in a serial workflow will not receive a notification.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/reminders

## HTTP Method

POST

## MIME Type

application/json

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose actors must receive a reminder	String (GUID)

## Response parameters

Empty response body

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The reminders have been sent.
<b>404 Not found</b>	The package with the given id could not be found.
<b>409 Conflict</b>	The package did not have status Pending.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

## 6. Documents

The **Documents** collection contains all resources and methods for creating and managing documents and documents data.

### What is a document?

A document is a digital file that is sent for signing. A document is always part of a package and is signed by a stakeholder. Note that as of eSignatures 6.3, a document may also be marked as optional. In that case, the end user may choose not to sign it.

A document may contain multiple elements. As of eSignatures 6.3, there are three types of elements: SigningField elements, CheckBoxField elements and TextBoxField elements.

### Package methods

- Add document to package
- Get documents
- Get document by ID
- Get document by OrderIndex
- Download document
- Delete document by ID

# Add document to package

## Description

The POST document call adds a document to an existing package.

Once a document has been added, you can build on this method by adding additional resources, such as elements (i.e. SigningFields, CheckBoxFields and TextBoxFields). You can also choose to define all this in a single POST document call, by passing all necessary parameters.

As of eSignatures 6.3 documents may be marked as optional. This allows you to create packages that contain both mandatory and optional documents. End users may then choose not to sign the optional documents within the package, while still signing the mandatory ones.

## Size limitations

Consider the following size limitations when creating documents.

- 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.

## Notes

- *Uploading PDF/A documents is only allowed if the format is pdfa1a or pdfa2a. When using itsme as signing method, it is mandatory to use pdfa1a or pdfa2a as TargetFormat. Note however that Connective does not perform any checks whether this TargetFormat has been selected.*
- *Uploading PDF/A documents to which you add form fields using eSignatures API v4 breaks the "/A" part of the PDF. As a result, the documents end up a regular PDF.*

*As a workaround, you can set the **\*\*TargetFormat\*** parameter in the POST document call to pdfa1a or pdfa2a (depending on the source format). The breaking will still happen but after all form fields have been filled or saved, the document will be converted to the set target format, resulting in a PDF/A document again.\**

- *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.*

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents

HTTP Method

POST

MIME Type (JSON + Base64)

application/json

MIME Type (Multiform)

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.

Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package to which the document will be added.	String (GUID)

Request parameters (for multipart/form-data)

<u>PARAMETERS</u>	<u>CONTENT TYPE</u>	<u>DESCRIPTION</u>
Parameters*	application/json;charset=utf 8 The supported parameters are identical to those of application/json described in the table below, except for contentType and base64data.	Parameters for creating a document
Document*	Supported values: application/msword, application/vnd.openxmlformats-officedocument.wordprocessingml.document, application/pdf, text/plain, application/xml	Multipart attached document that needs to be signed.
Representation	application/pdf	Multipart attached representation. A PDF to be shown together with the document to be signed

Request parameters (for application/json)

These parameters apply when creating a document with Base64 document.

The parameters with an asterisk are mandatory. All others are optional.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Name*	<p>Name of the document.</p> <p>The name is displayed in the eSignatures WebPortal.</p> <p><b>Note:</b> do not add an extension to the name value.</p> <p><b>Important:</b> The name must not contain any special characters such as slash, backslash, question mark, percent, asterisk, colon, pipe, quote double quote, less than, greater than.</p> <p><b>Note:</b> when using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.</p> <p>Minimum length: 1</p> <p>Maximum length: 150</p>	String (GUID)
Language*	<p>Language to use in signature texts.</p> <p>This is also the language that will be used for legal notices when LegalNoticeCode is filled for an Actor.</p> <p>Currently supported: en, nl, de, fr, es, da, nb, sv, fi, lv and pl.</p>	String (enum)
IsOptional	<p>Determines whether the document is optional.</p> <p>Default value is false.</p> <p>When set to true, signers may choose not to sign the document.</p>	Boolean
ExternalReference	<p>External reference to this resource given by a calling application.</p> <p>This parameter is not used by eSignatures itself.</p> <p>Maximum length: 256</p>	String
Elements	<p>The elements that will be added to the document.</p> <p>An element is an item that is placed on a document and may be assigned to an actor. For instance, a signing field that can be assigned to a signer actor.</p> <p>The elements parameters are identical to the ones in the <a href="#">Create element</a> call and are described there.</p>	Array of objects
ProofCorrelationId	Identifier to correlate this document to other resources in the proofs system.	String (GUID)
DocumentOptions*	Contains a document's base64 data and media type.	Object
RepresentationOptions	Contains a document's base64 data and media type.	Object

## DocumentOptions

Contains a document's base64 data and media type.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
TargetType	<p>targetType to which the document will be converted.</p> <p>Possible values are <b>application/pdf</b> and <b>application/xml</b></p>	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PdfOptions	This only applies when application/pdf is set as targetType. It defines optional pdf parameters.	Object
Base64data	base64 data of the document you will be sending.  Minimum length: 1  Maximum length: 200000000	String (base64)
ContentType*	contentType of the document you will be sending. Supported values are: application/pdf, application/xml, application/msword, application/vnd.openxmlformats-officedocument.wordprocessing.document, text/plain	String (mediaType)

#### PdfOptions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
TargetFormat	Defines if an extra conversion to needs to be done before signing. Supported values are: pdf, pdfa1a, pdfa2a.  <b>Notes:</b> This parameter 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.  <b>Important:</b> When using <b>itsme</b> as signing method, it is <b>mandatory</b> to use <b>pdfa1a</b> or <b>pdfa2a</b> as TargetFormat. Note however that Connective does not perform any checks whether this TargetFormat has been selected.  <b>Important:</b> Uploading PDF/A documents to which you add form fields using eSignatures API v4 breaks the "/A" part of the PDF. As a result, the documents end up a regular PDF. As a workaround, you can set the <b>TargetFormat</b> parameter in the POST document call to pdfa1a or pdfa2a (depending on the source format). The breaking will still happen but after all form fields have been filled or saved, the document will be converted to the set target format, resulting in a PDF/A document again.	string (enum)
PdfErrorhandling	How to deal with PDFs containing minor flaws. See section 4 for more info. Values: -Ignore -DetectWarn -DetectFail -DetectFixWarn -DetectFixFail	string (enum)

#### RepresentationOptions

Contains a document's base64 data and media type.

These parameters only apply when application/xml is set as ContentType.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Base64data	base64 data of the document you will be sending.  Minimum length: 1  Maximum length: 200000000	String (base64)
ContentType*	contentType of the document you will be sending. Supported value: application/pdf	String (mediaType)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Unique identifier of the document.	String (GUID)
Packageld	Unique identifier of the package.	String (GUID)
Name	Name of the document.  Minimum length: 1  Maximum length: 150	String (GUID)
CreationDate	Date and time when the document was created according to the server. Format is ISO 8601 date-time. E.g. 2020-01-23T12:34:00.000Z	String (date-time)
IsOptional	Whether the document is optional.	Boolean
MediaType	MediaType of the document within the package. Possible values: application/pdf or application/xml	String (enum)
Status	Status of the document within the package. The possible values are draft, pending, inProgress, ending, finished, rejected, revoked, expired, failed.	String (enum)
Elements	Details of each of the elements within the document. In the current version, an element is always a signing field. The elements parameters are identical to those in the <a href="#">Create element</a> call and are described there.	Array of objects
ExternalReference	External reference to this resource given by a calling application.  This parameter is not used by eSignatures itself.  Maximum length: 256	String

## Example request



```

{
  "Name": "string",
  "Language": "en",
  "IsOptional": false,
  "ExternalReference": "string",
  "Elements": [
    {
      "Type": "signingField",
      "ExternalReference": "myManualField",
      "SigningMethods": [
        "manual"
      ],
      "Location": {
        "Page": 1,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      }
    },
    {
      "Type": "signingField",
      "ExternalReference": "myBeidField",
      "FieldId": "signme",
      "SigningMethods": [
        "beid"
      ],
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      }
    }
  ],
  "ProofCorrelationId": "string",
  "DocumentOptions": {
    "targetType": "application/pdf",
    "pdfOptions": {
      "targetFormat": "pdfa1a"
    },
    "base64data": "string",
    "contentType": "application/pdf"
  }
}

```

Example response

```

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Name": "string",
  "CreationDate": "2020-08-05T13:18:19.465Z",
  "MediaType": "application/pdf",
  "Language": "en",
  "IsOptional": false,
  "Status": "draft",
  "Elements": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "ActorId": "00000000-0000-0000-0000-000000000000",
      "Type": "signingField",
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      },
      "Status": "pending",
      "ExternalReference": "string",
      "CompletedDate": "string",
      "UsedSigningMethod": "string",
      "SigningMethods": [
        "manual"
      ],
      "LegalNotice": {
        "Text": "Read and approved"
      }
    }
  ],
  "ExternalReference": "string"
}

```

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The document was created successfully.
<b>400 Bad request</b>	Request validation failed. Document could not be created due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.
<b>409 Conflict</b>	Document could not be created.

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldIsMissing

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	Document.NameLengthIncorrect
<b>400</b>	Document.UnsupportedLanguage
<b>400</b>	Request.UnsupportedValue
<b>400</b>	Base64Data.InvalidLength
<b>400</b>	Document.InvalidTargetFileType
<b>400</b>	PdfErrorHandling.InvalidType
<b>400</b>	Document.DataInvalid
<b>400</b>	Request.OneOfFieldsIsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	SigningType.Invalid
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	FormField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	FormField.InvalidHeightCoordinate
<b>400</b>	Document.DataInvalid
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>409</b>	Package.ApiVersionMismatch
<b>409</b>	User.NotFound
<b>409</b>	Document.CorrelationIdAlreadyExists
<b>409</b>	Document.Xml.UploadNotWellFormed
<b>409</b>	Package.PackageAcceptsDocumentType
<b>409</b>	Document.Xml.AlreadySignedNotAllowed

## 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.

# Get documents

## Description

This call retrieves all documents that currently exist within a specified package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/documents/

## HTTP Method

GET

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Packageld*	Unique identifier of the package whose documents must be retrieved.	String (GUID)

## Query parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Status	Gets items that have a specific status. Supported status values are: draft, pending, inProgress, ending, finished, rejected, revoked, expired, failed.	String (enum)

## Response parameters

The Response parameter are identical to the ones of the [Add document](#) call and are described there.

## Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Name": "string",
    "CreationDate": "2020-08-05T13:17:55.038Z",
    "MediaType": "application/pdf",
    "Language": "en",
    "IsOptional": false,
    "Status": "draft",
    "Elements": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "ActorId": "00000000-0000-0000-0000-000000000000",
        "Type": "signingField",
        "Location": {
          "Page": 2,
          "Top": 200,
          "Left": 200
        },
        "Dimensions": {
          "Width": 200,
          "Height": 200
        },
        "Status": "pending",
        "ExternalReference": "string",
        "CompletedDate": "string",
        "UsedSigningMethod": "string",
        "SigningMethods": [
          "manual"
        ],
        "LegalNotice": {
          "Text": "Read and approved"
        }
      }
    ],
    "ExternalReference": "string"
  }
]
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All documents are successfully retrieved.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect

# Get document by ID

## Description

This call allows you to retrieve a document by its ID.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/documents/{documentId}

## HTTP Method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
DocumentId*	Unique identifier of the document you want to retrieve.	String (GUID)
Packageld*	Unique identifier of the package to which the document belongs.	String (GUID)

## Response parameters

The Response parameter are identical to the ones of the [Add document](#) call and are described there.

## Example response

```

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Name": "string",
  "CreationDate": "2020-08-05T13:20:20.986Z",
  "MediaType": "application/pdf",
  "Language": "en",
  "IsOptional": false,
  "Status": "draft",
  "Elements": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "ActorId": "00000000-0000-0000-0000-000000000000",
      "Type": "signingField",
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      },
      "Status": "pending",
      "ExternalReference": "string",
      "CompletedDate": "string",
      "UsedSigningMethod": "string",
      "SigningMethods": [
        "manual"
      ],
      "LegalNotice": {
        "Text": "Read and approved"
      }
    }
  ],
  "ExternalReference": "string"
}

```

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Document is successfully retrieved.
<b>400 Bad request</b>	Request failed validation.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.



# Get document by orderIndex

## Description

This call allows you retrieve a document within a specified package based on its OrderIndex.

The OrderIndex defines the order in which documents should be sorted within a package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/documents/{orderIndex}

## HTTP Method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the target package.	String (GUID)
OrderIndex*	Order of the document within the package. Value must be positive.	Integer

## Example response

```

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Name": "string",
  "CreationDate": "2020-08-05T13:20:49.492Z",
  "MediaType": "application/pdf",
  "Language": "en",
  "IsOptional": false,
  "Status": "draft",
  "Elements": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "ActorId": "00000000-0000-0000-0000-000000000000",
      "Type": "signingField",
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      },
      "Status": "pending",
      "ExternalReference": "string",
      "CompletedDate": "string",
      "UsedSigningMethod": "string",
      "SigningMethods": [
        "manual"
      ],
      "LegalNotice": {
        "Text": "Read and approved"
      }
    }
  ],
  "ExternalReference": "string"
}

```

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Document is successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect

# Download document

## Description

This call download a specified, signed, document from a specified package as PDF or XML file.

Note that you can only download documents from fully signed packages.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageId}/download/{documentId}](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/download/{documentId})

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	The unique identifier of the package of which you want to download a signed document.	String (GUID)
DocumentId*	The unique identifier of the signed document you want to download.	String (GUID)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The document is successfully downloaded.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Specified package /document could not be found.
<b>409 Conflict</b>	The specified package has not been fully signed.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

# Delete document by ID

## Description

This call allows you to delete a specified document from a package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/documents/{documentId}

## HTTP Method

Delete

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
DocumentId*	Unique identifier of the document that must be deleted.	String (GUID)
Packageld*	Unique identifier of the package to which the document belongs	String (GUID)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Document was successfully deleted.
<b>404 Not found</b>	Target package or document could not be found.
<b>409 Conflict</b>	Document could not be deleted.

## Error codes

TBD

## 7. Elements

The **Elements** collection contains all resources and methods for creating and managing elements.

### What is an element?

An element is an item that is placed on a document and may be assigned to an actor.

An element cannot exist on its own, it is always part of a document.

Currently, three types of elements can be added to a document:

- SigningField elements
- TextBoxField elements
- CheckBoxField elements

**SigningField** elements determine where on the document a signature must be placed. SigningFields are always mandatory.

**TextBoxField** elements determine where on the document a textbox field will be placed. A textbox may contain a default, prefilled value, which can be modified afterwards by the end user. A textbox may be mandatory or optional.

**CheckBoxField** elements determine where on the document a checkbox field will be placed. A checkbox may be already checked by default, but can still be modified by the end user. Like textboxes, checkboxes may also be mandatory or optional.

### Element methods

- Create element
- Get elements
- Get element by ID
- Delete element by ID

# CreateSigningField element

## Description

The POST element call adds an element to an existing document.

Currently, three types of elements are supported:

- SigningField element
- CheckBoxField element
- TextBoxField element

This sections covers the **SigningField** element parameters.

A SigningField element can be created in three different ways:

- Using specific dimensions and a specific location
- Using a marker
- Using a field identifier

**Important:** A single document must not contain more than 30 SigningField elements.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents/{documentId}/elements

## HTTP Method

POST

## MIME Type

application/json

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the document belongs.	String (GUID)
DocumentId*	Unique identifier of the package to which the element must be added.	String (GUID)

## Request parameters

These are the parameters to create a SigningField element.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Type*	Type of element that must be created. In this case SigningField element.	String (enum)
ExternalReference	External reference to this resource given by a calling application.  This parameter is not used by eSignatures itself.  Maximum length: 256	String
SigningMethods	The SigningMethods that may be used to signing this signing field. Note: When using a single SigningMethod, it must also be placed in an array.	Array of strings

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
LegalNotice	Legal notice that will be added to the signing field.	Object

## LegalNotice

A legal notice is a text the signer must retype before they are able to place their signature. Note that legal notices are case-sensitive.

When defining a DefaultLegalNotice, you can choose from two DefaultLegalNotice types: **name** or **text**.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Text	Use the text parameter to use a custom legal notice, i.e. one that is not configured in the Config Index. The legal notice should be written in the same language as the documents of the package.  Maximum length: 255	String
Name	The name of the legal notice that will be added to the signing field. To correctly use this parameter, you need to know the name of the different legal notices that are configured in the Config Index. E.g. LegalNotice1. The value of the legal notices is also set in the Config Index. Note that the language in which the legal notice is displayed depends on the language of the document.  Maximum length: 20	String (enum)

## Element with specific dimensions and specific location

### Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page of the document on which the element is placed. The paging works as follows: '1' is the first page of the PDF, '2' the second, and so on. '0' must not be used as page value. To count backwards from the last page, use negative integers: '-1' for the last page, '-2' for the second to last, and so on.	Integer
Top*	How far from the top edge of the document the element is placed.  Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed.  Minimum value is 0.	Number (float)

### Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element in pixels.  Minimum value is 112.  Maximum value is 999999.	Number (float)
Height*	Height of the element in pixels.  Minimum value is 70.  Maximum value is 999999	Number (float)

## Element with marker

Use this when a marker has been placed on the documents you will be uploading.

**Attention:** For performance reasons it's recommended to do this at document upload time.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Marker	A marker is a piece of text inside the document where an element must be. It contains location and dimensions of the element. Pattern: <code>#[a-zA-Z]+(?:\d*.)?\d+</code> .	String

See [Detecting markers](#) for more info.

## Element with field identifier

The **FieldId** parameter must only be used for documents that already contain input fields (created in Adobe Acrobat DC for instance), and you want to convert those input fields into signing fields.

**Attention:** For performance reasons it's recommended to do this at document upload time. It may happen that after a document conversion the field with given identifier doesn't exist anymore.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
FieldId	The name of an input field on the document. Note that the FieldId must be unique per document.	String

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Element ID	String (GUID)
ActorId	Actor to which the element is linked.	String (GUID)
Type	Type of element that was created. In this case SigningField.	String (enum)
Location	Location of the element within the document.	Object
Dimensions	Dimensions of the element.	Object
Status	The status of the document element. Possible values: Pending, InProgress, Rejected, Failed, Finished, Refused.	String (enum)
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
CompletedDate	Date and time on which this element was completed.	String (date-time)
UsedSigningMethod	SigningMethod that was used to sign this SigningField.	String
SigningMethods	The SigningMethods that may be used to sign this signing field.	Array of strings



<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
LegalNotice	Legal notice that will be added to the signing field.	Object

#### Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page within the document the element is on. Minimum value is 0.	Integer
Top*	How far from the top edge of the document the element is placed. Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed. Minimum value is 0.	Number (float)

#### Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element. Minimum value is 112. Maximum value is 999999.	Number (float)
Height*	Height of the element. Minimum value is 70. Maximum value is 999999	Number (float)

#### LegalNotice

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Text	Text of the legal notice.	String

#### Example request

```
{
  "Type": "SigningField",
  "SigningMethods": [
    "manual"
  ],
  "LegalNotice": {
    "Text": "read and approved"
  },
  "Location": {
    "Page": 2,
    "Top": 200,
    "Left": 200
  },
  "Dimensions": {
    "Width": 200,
    "Height": 200
  }
}
```

#### Example response

```

{
  "Type": "SigningField",
  "SigningMethods": [
    "manual"
  ],
  "LegalNotice": {
    "Text": "read and approved"
  },
  "Location": {
    "Page": 2,
    "Top": 200,
    "Left": 200
  },
  "Dimensions": {
    "Width": 200,
    "Height": 200
  }
}

```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The element was created successfully.
<b>400 Bad request</b>	Request validation failed. Element could not be created due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.
<b>409 Conflict</b>	Element could not be created.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldIsMissing
<b>400</b>	Request.OneOfFieldsIsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	SigningType.Invalid
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	Element.FieldNotUnique
<b>401</b>	Authentication failed, username or password incorrect

<u>HTTP CODE</u>	<u>CODE</u>
409	Element.ExternalReferenceNotUnique
409	Element.UsedFieldId
409	Package.InvalidStatus
409	Document.NotFoundInPackage
409	SigningField.InvalidPage
409	LegalNotice.NotFound
409	SigningField.MarkerOrFieldIdNotFound
409	SigningField.MultipleMarkersFound
409	SigningField.MarkerAlreadySigned
409	SigningField.InvalidWidthMarker

# CreateTextBoxField element

## Description

The POST element call adds an element to an existing document.

Currently, three types of elements are supported:

- SigningField element
- CheckBoxField element
- TextBoxField element

This sections covers the **TextBoxField** element parameters.

A TextBoxField element can be created in three different ways:

- Using specific dimensions and a specific location
- Using a marker
- Using a field identifier

Note that the three ways of creating a TextBoxField element can be combined within a single POST element call.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents/{documentId}/elements

## HTTP Method

POST

## MIME Type

application/json

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package to which the document belongs.	String (GUID)
DocumentId*	Unique identifier of the package to which the element must be added.	String (GUID)

## Request parameters

These are the parameters to create a TextBoxField element.

**Important:** The parameters **Name**, **IsMultiline** and **CharLimit** must *only* be used to create TextBoxField elements using **element location** and **element dimensions**. They must *not* be used when creating TextBoxField elements using markers, or when specifying existing form fields (created in Adobe for instance) by means of the **FieldId** parameter.

Note however, that it is supported to create different types of TextBoxField elements within a single POST element call.

PARAMETER	CONTENT / DESCRIPTION	TYPE
Type*	Type of element that must be created. In this case TextBoxField.	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.  Maximum length: 256	String
Name	Name of the form field. Note that the name must be unique per document.	String
ToolTipLabel	The value that will be shown as tooltip when a user hovers over the form field with the cursor. <b>Note:</b> If you're sending a document that already contains a prefilled form field with a tooltip, and enter a ToolTipLabel parameter, the value of this parameter will overwrite the original value. To avoid overwritten the original value, the ToolTipLabel parameter must be null.	String
IsRequired	Determines whether the TextBoxField is mandatory or optional. When set to true, the text field is mandatory. The default value is false. <b>Note:</b> If you're sending a document that already contains a prefilled form field, make sure the IsRequired parameter is null. Otherwise the original value will be overwritten.	Boolean
DefaultValue	Default value prefilled in the TextBoxField.  <b>Note:</b> If you're sending a document that already contains a prefilled form field and enter a DefaultValue parameter, the value of this parameter will overwrite the original value. To avoid overwritten the original value, the DefaultValue parameter must be null.	String
IsMultiLine	Determines whether the TextBoxField has multiple lines. Default value is false. Wen set to true, the TextBoxField will be scrollable.	Boolean
CharLimit	Determines the maximum number of characters that may be entered in the TextBoxField.	Integer

Element with specific dimensions and specific location

Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page of the document on which the element is placed. The paging works as follows: '1' is the first page of the PDF, '2' the second, and so on. '0' must not be used as page value. To count backwards from the last page, use negative integers: '-1' for the last page, '-2' for the second to last, and so on.	Integer
Top*	How far from the top edge of the document the element is placed.  Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed.  Minimum value is 0.	Number (float)

Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element in pixels.  Minimum value is 10.  Maximum value is 999999.	Number (float)
Height*	Height of the element in pixels.  Minimum value is 10.  Maximum value is 999999	Number (float)

### Element with marker

Use this when a marker has been placed on the documents you will be uploading.

**Attention:** For performance reasons it's recommended to do this at document upload time.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Marker	A marker is a piece of text inside the document where an element must be. It contains location and dimensions of the element. Pattern: @[a-zA-Z]+(?:\d*.)?\d+	String

See [Detecting markers](#) for more info.

### Element with field identifier

The **FieldId** parameter must only be used documents that already contain input fields (created in Adobe Acrobat Pro for instance), and you want to show those input fields as textbox fields.

**Attention:** For performance reasons it's recommended to do this at document upload time. It may happen that after a document conversion the field with given identifier doesn't exist anymore.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
FieldId	The name of an input field on the document. Note that the FieldId must be unique per document.	String

### Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Element ID	String (GUID)
ActorId	Actor to which the element is linked.	String (GUID)
Type	Type of element that was created. In this case TextBoxField.	String (enum)
Location	Location of the element within the document.	Object
Dimensions	Dimensions of the element.	Object

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
Status	The status of the document element. The possible values are: Pending, InProgress, Rejected, Failed, Finished, Refused.	String (enum)
CompletedDate	Date and time on which this element was completed.	String (date-time)
Name	Name of the form field.	String
TooltipLabel	The value that will be shown as tooltip when a user hovers over the form field with the cursor.	String
IsRequired	Whether the TextFieldBox is mandatory or optional.	Boolean
DefaultValue	Default value prefilled in the TextBoxField.	String
IsMultiLine	Whether the TextBoxField has multiple lines.	Boolean
CharLimit	Maximum number of characters that may be entered in the TextBoxField.	Integer
Value	Value of the completed TextBoxField.	String

#### Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page within the document the element is on. Minimum value is 0.	Integer
Top*	How far from the top edge of the document the element is placed. Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed. Minimum value is 0.	Number (float)

#### Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element. Minimum value is 10. Maximum value is 999999.	Number (float)
Height*	Height of the element. Minimum value is 10. Maximum value is 999999	Number (float)

#### Example request

```
{
  "type": "textBoxField",
  "name": "Textbox1",
  "ToolTipLabel": "Enter text here",
  "IsRequired": false,
  "DefaultValue": "Hello",
  "IsMultiline": true,
  "CharLimit": 100,
  "location": {
    "page": "1",
    "left": "100",
    "top": "50"
  },
  "dimensions": {
    "width": "150",
    "height": "50"
  },
  "documentIndex": 0
},
```

## Example response

```
{
  "Type": "textBoxfield",
  "DefaultValue": "Hello",
  "IsMultiline": true,
  "CharLimit": 100,
  "Value": null,
  "Name": "Textbox1",
  "ToolTipLabel": "Enter text here",
  "IsRequired": false,
  "Id": "54e6d471-17ad-44e0-9798-b27c04da13d4",
  "ActorId": "b2954765-0a50-4015-9287-abb2629dee0f",
  "Location": {
    "Page": 1,
    "Top": 50.0,
    "Left": 100.0
  },
  "Dimensions": {
    "Width": 150.0,
    "Height": 50.0
  },
  "Status": "Pending",
  "ExternalReference": null,
  "CompletedDate": null
},
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The element was created successfully.
<b>400 Bad request</b>	Request validation failed. Element could not be created due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.
<b>409 Conflict</b>	Element could not be created.



## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
400	Request.RequiredFieldsIsMissing
400	Request.OneOfFieldsIsMissing
400	Request.FieldMaxLength
400	Request.FieldMinimumValue
400	Request.FieldMaxLength
400	FormField.ShouldBeEmpty:{field}
400	TextBoxField.InvalidCharLimit
400	Element.FieldNotUnique
401	Authentication failed, username or password incorrect
409	Element.FieldIdNotUnique
409	Element.ExternalReferenceNotUnique
409	Element.UsedFieldId
409	FormField.NameNotUnique
409	FormField.InvalidPage
409	FormField.InvalidWidthCoordinate
409	FormField.InvalidHeightCoordinate
409	FormField.InvalidWidthMarker
409	FormField.InvalidHeightMarker
409	FormField.MarkerOrFieldIdNotFound
409	FormField.MultipleMarkersFound
409	Package.InvalidStatus
409	Document.NotFoundInPackage

# CreateCheckBoxField element

## Description

The POST element call adds an element to an existing document.

Currently, three types of elements are supported:

- SigningField element
- CheckBoxField element
- TextBoxField element

This sections covers the **CheckBoxField** element parameters.

A CheckBoxField element can be created in three different ways:

- Using specific dimensions and a specific location
- Using a marker
- Using a field identifier

Note that the three ways of creating a CheckBoxField element can be combined within a single POST element call.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents/{documentId}/elements

## HTTP Method

POST

## MIME Type

application/json

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package to which the document belongs.	String (GUID)
DocumentId*	Unique identifier of the package to which the element must be added.	String (GUID)

## Request parameters

These are the parameters to create a CheckBoxField element.

**Important:** The **Name** parameter must *only* be used to create CheckBoxField elements using **element location** and **element dimensions**. It must *not* be used when creating CheckBoxField elements using markers, or when specifying existing form fields (created in Adobe for instance) by means of the **FieldId** parameter.

Note however, that it is supported to create different types of CheckBoxField elements within a single POST element call.

PARAMETER	CONTENT / DESCRIPTION	TYPE
Type*	Type of element that must be created. In this case CheckBoxField.	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.  Maximum length: 256	String
Name	Name of the form field. Note that the name must be unique per document.	String
ToolTipLabel	The value that will be shown as tooltip when a user hovers over the form field with the cursor. <b>Note:</b> If you're sending a document that already contains a prefilled form field with a tooltip, and enter a ToolTipLabel parameter, the value of this parameter will overwrite the original value. To avoid overwritten the original value, the ToolTipLabel parameter must be null.	String
IsRequired	Determines whether the checkbox is mandatory or optional. When set to true, the checkbox is mandatory. The default value is false. <b>Note:</b> If you're sending documents that already contain a prefilled form field, make sure the IsRequired parameter is null. Otherwise the original value will be overwritten.	Boolean
DefaultValue	Determines whether the checkbox is checked or unchecked by default. When set to true, the checkbox is checked. The default value is false. <b>Note:</b> If you're sending documents that already contain a prefilled form field, make sure the DefaultValue parameter is set to false. Otherwise the original value will be overwritten.	Boolean

Element with specific dimensions and specific location

Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page of the document on which the element is placed. The paging works as follows: '1' is the first page of the PDF, '2' the second, and so on. '0' must not be used as page value. To count backwards from the last page, use negative integers: '-1' for the last page, '-2' for the second to last, and so on.	Integer
Top*	How far from the top edge of the document the element is placed.  Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed.  Minimum value is 0.	Number (float)

Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element in pixels.  Minimum value is 10.  Maximum value is 999999.	Number (float)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Height*	Height of the element in pixels.  Minimum value is 10.  Maximum value is 999999	Number (float)

### Element with marker

Use this when a marker has been placed on the documents you will be uploading.

**Attention:** For performance reasons it's recommended to do this at document upload time.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Marker	A marker is a piece of text inside the document where an element must be. It contains location and dimensions of the element. Pattern: @[a-zA-Z]+(?:\d*.)?\d+.	String

See [Detecting markers](#) for more info.

### Element with field identifier

The **FieldId** parameter must only be used documents that already contain input fields (created in Adobe Acrobat Pro for instance), and you want to show those input fields as checkbox fields.

**Attention:** For performance reasons it's recommended to do this at document upload time. It may happen that after a document conversion the field with given identifier doesn't exist anymore.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
FieldId	The name of an input field on the document. Note that the FieldId must be unique per document.	String

### Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Element ID	String (GUID)
ActorId	Actor to which the element is linked.	String (GUID)
Type	Type of element that was created. In this case CheckBoxField.	String (enum)
Location	Location of the element within the document.	Object
Dimensions	Dimensions of the element.	Object
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
Status	The status of the document element. The possible values are: Pending, InProgress, Rejected, Failed, Finished, Refused.	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
CompletedDate	Date and time on which this element was completed.	String (date-time)
Name	Name of the form field.	String
TooltipLabel	The value that will be shown as tooltip when a user hovers over the form field with the cursor.	String
IsRequired	Whether the checkbox is mandatory or optional.	Boolean
DefaultValue	Whether the checkbox is by default checked or unchecked.	Boolean
Checked	The value of the completed checkbox field.	Boolean

Location

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Page*	The page within the document the element is on. Minimum value is 0.	Integer
Top*	How far from the top edge of the document the element is placed. Minimum value is 0.	Number (float)
Left*	How far from the left edge of the document the element is placed. Minimum value is 0.	Number (float)

Dimensions

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Width*	Width of the element. Minimum value is 10. Maximum value is 999999.	Number (float)
Height*	Height of the element. Minimum value is 10. Maximum value is 999999	Number (float)

Example request

```
{
  "type": "checkboxField",
  "name": "Checkbox10",
  "ToolTipLabel": "Optional",
  "IsRequired": false,
  "DefaultValue": false,
  "location": {
    "page": "1",
    "left": "300",
    "top": "230"
  },
  "dimensions": {
    "width": "12",
    "height": "12"
  },
  "documentIndex": 0
},
```

### Example response

```
{
  "Type": "checkboxfield",
  "DefaultValue": false,
  "Checked": false,
  "Name": "Checkbox10",
  "ToolTipLabel": "Optional",
  "IsRequired": false,
  "Id": "c4577ad0-fcca-43b3-bb39-465c6e0a27f5",
  "ActorId": "b2954765-0a50-4015-9287-abb2629dee0f",
  "Location": {
    "Page": 1,
    "Top": 230.0,
    "Left": 300.0
  },
  "Dimensions": {
    "Width": 12.0,
    "Height": 12.0
  },
  "Status": "Pending",
  "ExternalReference": null,
  "CompletedDate": null
},
```

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The element was created successfully.
<b>400 Bad request</b>	Request validation failed. Element could not be created due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.
<b>409 Conflict</b>	Element could not be created.

### Error codes

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	Request.RequiredFieldIsMissing
<b>400</b>	Request.OneOfFieldsIsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	Request.FieldMaxLength
<b>400</b>	FormField.ShouldBeEmpty:{field}
<b>400</b>	Element.FieldNotUnique
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Element.FieldIdNotUnique
<b>409</b>	Element.ExternalReferenceNotUnique
<b>409</b>	Element.UsedFieldId
<b>409</b>	FormField.NameNotUnique
<b>409</b>	FormField.InvalidPage
<b>409</b>	FormField.InvalidWidthCoordinate
<b>409</b>	FormField.InvalidHeightCoordinate
<b>409</b>	FormField.InvalidWidthMarker
<b>409</b>	FormField.InvalidHeightMarker
<b>409</b>	FormField.MarkerOrFieldIdNotFound
<b>409</b>	FormField.MultipleMarkersFound
<b>409</b>	Package.InvalidStatus
<b>409</b>	Document.NotFoundInPackage

# Get elements

## Description

This call retrieves all elements within a specified document.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents/{documentId}/elements

## HTTP Method

GET

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose elements you want to retrieve.	String (GUID)
DocumentId*	Unique identifier of the document whose elements you want to retrieve.	String (GUID)

## Response parameters

The response parameters are identical to those of the [Create element](#) call and are described there.

## Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "ActorId": "00000000-0000-0000-0000-000000000000",
    "Type": "SigningField",
    "Location": {
      "Page": 1,
      "Top": 200,
      "Left": 200
    },
    "Dimensions": {
      "Width": 200,
      "Height": 200
    }
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "SigningField",
    "Location": {
      "Page": 2,
      "Top": 250,
      "Left": 250
    },
    "Dimensions": {
      "Width": 230,
      "Height": 230
    }
  }
]
```

## Response codes



<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All elements are successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document id could not be found.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound
<b>409</b>	Document.NotFoundInPackage

# Get element by ID

## Description

This call allows you to retrieve an element by its ID.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/documents/{documentId}/elements/{elementId}

## HTTP Method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package to which the document belongs.	String (GUID)
DocumentId*	Unique identifier of the document to which the element belongs.	String (GUID)
ElementId*	Unique identifier of the element you want to retrieve.	String (GUID)

## Example response

{no example available}
------------------------

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Element is successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or document could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound
<b>409</b>	Document.NotFoundInPackage

# Delete element by ID

## Description

This call allows you to delete a specified element from a document.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/documents/{documentId}/elements/{elementId}

## HTTP Method

DELETE

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the document belongs.	String (GUID)
DocumentId*	Unique identifier of the document to which the element belongs.	String (GUID)
ElementId*	Unique identifier of the element that must be deleted.	String (GUID)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Element was successfully deleted.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package, document or element could not be found.
<b>409 Conflict</b>	Element could not be deleted.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus
<b>404</b>	Document.NotFound
<b>404</b>	Location.NotFound

## 8. Stakeholders

The **Stakeholders** collection contains all resources and methods for creating and managing stakeholders.

### What is a stakeholder?

A stakeholder is an entity that has an interest in a package. A single stakeholder may contain multiple actors, i.e. actions the stakeholders must take. For instance, fill out forms, approve, sign or receive.

A stakeholder may be a single person, or a group of people. When a group of people is defined, any member of the group will be able to approve or sign on behalf of the entire group.

### Stakeholder methods

- Add stakeholder to package
- Get stakeholders
- Get stakeholder by ID
- Delete stakeholder by ID

# Create stakeholder

## Description

The POST stakeholders call creates a stakeholder and adds it to a specified package.

A stakeholder is an entity that has an interest in a package.

Three types of stakeholders can be created:

- a **person** stakeholder
- a **group** stakeholder
- a **contactGroup** stakeholder

A person stakeholder is a single person. Only that person will be able to take action on the package.

A group stakeholder is a group of people who can interact with the package. The members of the group must be specified within the POST stakeholder call. When a group of people is defined, any member of the group will be able to approve or sign on behalf of the entire group. Each person of the group will receive a unique URL to approve/sign their document. **Attention:** as soon as one member of the group has taken action, the others no longer can.

A contactGroup stakeholder can be used when a contact group has been created in the eSignatures WebPortal. This type functions in the same way as a group stakeholder. The only difference is that you simply pass the contactGroup code obtained in the WebPortal instead of defining each member in the call.

**Note:** the stakeholder type contactGroup only exists when the package is in draft status. When the status changes to pending, the stakeholder type contactGroup is converted to **group**. The contactGroupCode value remains.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/stakeholders

## HTTP method

## POST

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the stakeholder will be added.	string (GUID)

## Request parameters

## Actor parameters

The Actor parameters are identical to those of the [Create actor](#) call and are described there.

## Create Person stakeholder

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Type*	Type of stakeholder that must be created. Supported values: person, group and contactGroup.	String (enum)
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.  Maximum length: 256	String

PARAMETER	CONTENT / DESCRIPTION	TYPE
Language*	UI language of this stakeholder, expressed in a 2 letter ISO 639-1 code. Currently supported: en, nl, de, fr, es, da, nb, sv, fi, lv and pl.	String (enum)
FirstName	First name of the stakeholder Maximum length: 150	String
LastName*	Last name of the stakeholder Maximum length: 150	String
EmailAddress*	Email address	String (email)
BirthDate*	Date of birth in format: YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the Config Index might make this parameter required.	String (date)
PhoneNumber	Phone number to receive an SMS OTP. <b>Note:</b> 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". <b>Important:</b> never use spaces in the phone number format.	String
AdditionalProperties	<p>In this array you can pass the additional properties that have been configured in the <b>Contact Properties Settings</b> of the Config Index. Additional properties may contain any additional stakeholder information the administrator deemed necessary. For instance, the stakeholder's maiden name, their nationality, etc.</p> <p>By default two additional properties are configured in every environment: <b>BeID</b> and <b>BeLawyer</b>. In the <b>BeID</b> property you can pass the stakeholder's national security number and in the <b>BeLawyer</b> property you can pass their lawyerID.</p> <p>As of eSignatures 6.2.1, mandated signing rules can be configured in the Config Index and applied to any additional property. When a mandated signing rule has been applied, the additional property to which it is applied becomes mandatory. Their value will be checked against the external data extracted from the signing certificate or returned by the signing service. If the data matches, the stakeholder will be mandated to sign.</p> <p><b>Tip:</b> To know which additional properties are mandatory for which SigningMethod, do a Get SigningMethods call. To know which additional properties are supported but not mandatory, you need to check the Config Index or contact your administrator.</p>	Array of strings

## Create Group stakeholder

Note that iDIN signing currently doesn't support Group stakeholders.

PARAMETER	CONTENT / DESCRIPTION	TYPE
Type*	Type of stakeholder that must be created.	String (enum)
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
GroupName*	Name of the group Maximum length: 128	String
Members*	Members that must be added to the group. The parameters are identical to those of a Person Stakeholder, except for type.	Array of objects

## Create Contact Group stakeholder

PARAMETER	CONTENT / DESCRIPTION	TYPE
Type*	Type of stakeholder that must be created.	String (enum)
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
ContactGroupCode*	Code that was generated when creating a contact group in the eSignatures WebPortal.	String

## Response parameters

### Person stakeholder

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Unique identifier of the stakeholder.	String (GUID)
Packageld	Unique identifier of the package to which the stakeholder is added.	String (GUID)
Type	Type of stakeholder that is created. Supported values: person, group and contactGroup.	String (enum)
Actors	Actors that are added to the stakeholder. Their parameters are identical to those of the <a href="#">Create actor</a> call and are described there.	Array of objects
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.  Maximum length: 256	String
Language	UI language of this stakeholder, expressed in a 2 letter ISO 639-1 code.	String (enum)
FirstName	First name of the stakeholder Maximum length: 150	String
LastName	Last name of the stakeholder Maximum length: 150	String
EmailAddress	Email address	String (email)
PhoneNumber	Phone number to receive an SMS OTP.	String
BirthDate	Date of birth in format: YYYY-MM-DD	String (date)
AdditionalProperties	The additional properties that have been passed for the stakeholder. E.g. "Beld": "12345678900" or "Nationality": "Belgian"	Array of strings

## Create Group stakeholder

PARAMETER	CONTENT / DESCRIPTION	TYPE
Id	Unique identifier of the stakeholder.	String (GUID)

PARAMETER	CONTENT / DESCRIPTION	TYPE
Packageld	Unique identifier of the package to which the stakeholder is added.	String (GUID)
Type	Type of stakeholder that must be created.	String (enum)
Actors	Actors that are added to the stakeholder. Their parameters are identical to those of the <a href="#">Create actor</a> call and are described there.	Array of objects
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
GroupName	Name of the group	String
ContactGroupCode	The contactGroupCode that was used to create this group.	String
Members	Members that are added to the group. The parameters are identical to those of a Person Stakeholder, except for type.	Array of objects

### Create Contact Group stakeholder

PARAMETER	CONTENT / DESCRIPTION	TYPE
Id	Unique identifier of the stakeholder.	String (GUID)
Packageld	Unique identifier of the package to which the stakeholder is added.	String (GUID)
Type	Type of stakeholder that must be created.	String (enum)
Actors	Actors that are added to the stakeholder. Their parameters are identical to those of the <a href="#">Create actor</a> call and are described there.	Array of objects
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself. Maximum length: 256	String
ContactGroupCode	An eSignatures contactGroup identifier.	String
Members	Members that are added to the group. The parameters are identical to those of a Person Stakeholder, except for type.	Array of objects

### Example request



```

{
  "Type": "group",
  "GroupName": "Great mates",
  "Actors": [
    {
      "Type": "signer",
      "Elements": [
        {
          "Type": "signingField",
          "Location": {
            "Page": 2,
            "Top": 200,
            "Left": 200
          },
          "Dimensions": {
            "Width": 200,
            "Height": 200
          }
        }
      ]
    }
  ],
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14",
      "AdditionalProperties": {
        "BeId": "12345678900",
        "BeLawyer": "b459d74c-1f90-4d63-9f4a-6de9cead8c5e"
      }
    }
  ]
}

```

Example response

```

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Type": "group",
  "Actors": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "Type": "signer",
      "Status": "available",
      "MemberLinks": [
        {
          "Email": "zu@li.test",
          "Link": "https://dothething.test"
        },
        {
          "Email": "john@doe.test",
          "Link": "https://alsocompletelyouraction.test"
        }
      ]
    }
  ],
  "GroupName": "string",
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
      "AdditionalProperties": {
        "BeId": "12345678900",
        "BeLawyer": "b459d74c-1f90-4d63-9f4a-6de9cead8c5e"
      }
    }
  ]
}

```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The stakeholder was created successfully.
<b>400 Bad request</b>	Request validation failed. The stakeholder could not be created due to invalid parameters in the request.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.
<b>409 Conflict</b>	Stakeholder could not be created.

Error codes

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	Stakeholder.UnsupportedLanguage
<b>400</b>	Stakeholder.EmailAddressInvalid
<b>400</b>	Stakeholder.BirthDayInvalid
<b>400</b>	Stakeholder.BirthDayInFuture
<b>400</b>	Stakeholder.InvalidPhoneNumber
<b>400</b>	GroupMember.RequiredFieldsMissing
<b>400</b>	GroupMember.UnsupportedLanguage
<b>400</b>	GroupMember.EmailAddressInvalid
<b>400</b>	GroupMember.BirthDayInvalid
<b>400</b>	GroupMember.BirthDayInFuture
<b>400</b>	GroupMember.InvalidPhoneNumber
<b>400</b>	Request.OneOfFieldsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	SigningType.Invalid
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	Url.Invalid
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>409</b>	ContactGroup.NotFound
<b>409</b>	Package.ContainsEmptyContactGroup

<u>HTTP CODE</u>	<u>CODE</u>
409	Stakeholder.TypeInvalid

# Get stakeholders

## Description

This call retrieves all stakeholders that currently exist within a specified package.

The results are ordered by date on which the stakeholders were added to the package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/stakeholders

## HTTP method

GET

## Query parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package whose stakeholders you want to retrieve.	String (GUID)

## Response parameters

The Response parameters per package are identical to the ones of the [Create stakeholder](#) call and are described there.

## Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "person",
    "Actors": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "available",
        "Links": [
          "https://dothething.test"
        ]
      },
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "unavailable"
      },
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "skipped"
      }
    ],
    "ExternalReference": "string",
    "Language": "en",
    "FirstName": "string",
    "LastName": "string",
    "EmailAddress": "hello@world.test",
    "BirthDate": "2020-01-22"
  },
  {
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "group",
    "Actors": [
```

```

    ACTORS : [
    {
        "Id": "00000000-0000-0000-0000-000000000000",
        "Type": "signer",
        "Status": "available",
        "MemberLinks": [
            {
                "Email": "zu@li.test",
                "Link": "https://dothething.test"
            },
            {
                "Email": "john@doe.test",
                "Link": "https://alsocompleteyouraction.test"
            }
        ]
    }
],
"GroupName": "Group123",
"Members": [
    {
        "Language": "en",
        "FirstName": "John",
        "LastName": "Doe",
        "EmailAddress": "john@doe.test",
        "BirthDate": "1990-01-16"
    },
    {
        "Language": "en",
        "FirstName": "Zu",
        "LastName": "Li",
        "EmailAddress": "zu@li.test",
        "BirthDate": "1991-03-14"
    }
]
},
{
    "Id": "00000000-0000-0000-0000-000000000000",
    "PackageId": "00000000-0000-0000-0000-000000000000",
    "Type": "contactGroup",
    "ContactGroupCode": "00002",
    "Actors": [
        {
            "Id": "00000000-0000-0000-0000-000000000000",
            "Type": "signer",
            "Status": "available",
            "MemberLinks": [
                {
                    "Email": "john@doe.test",
                    "Link": "https://alsocompleteyouraction.test"
                },
                {
                    "Email": "zu@li.test",
                    "Link": "https://dothething.test"
                }
            ]
        }
    ]
},
"ExternalReference": "string",
"Members": [
    {
        "Language": "en",
        "FirstName": "John",
        "LastName": "Doe",
        "EmailAddress": "john@doe.test",
        "BirthDate": "1990-01-16"
    },

```

```
{
  "Language": "en",
  "FirstName": "Zu",
  "LastName": "Li",
  "EmailAddress": "zu@li.test",
  "BirthDate": "1991-03-14"
}
]
```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All stakeholders are successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound

# Get stakeholder by ID

## Description

This call allows you to retrieve a stakeholder by its ID.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the stakeholder belongs.	String (GUID)
StakeholderId*	Unique identifier of the stakeholder you want to retrieve.	String (GUID)

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/stakeholders/{stakeholderId}

## HTTP method

GET

## Response parameters

The Response parameter are identical to the ones of the [Create stakeholder](#) call and are described there.

## Example response



```

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "PackageId": "00000000-0000-0000-0000-000000000000",
  "Type": "group",
  "Actors": [
    {
      "Id": "00000000-0000-0000-0000-000000000000",
      "Type": "signer",
      "Status": "available",
      "MemberLinks": [
        {
          "Email": "zu@li.test",
          "Link": "https://dothething.test"
        },
        {
          "Email": "john@doe.test",
          "Link": "https://alsocompleteyouraction.test"
        }
      ]
    }
  ],
  "GroupName": "Group123",
  "Members": [
    {
      "Language": "en",
      "FirstName": "John",
      "LastName": "Doe",
      "EmailAddress": "john@doe.test",
      "BirthDate": "1990-01-16"
    },
    {
      "Language": "en",
      "FirstName": "Zu",
      "LastName": "Li",
      "EmailAddress": "zu@li.test",
      "BirthDate": "1991-03-14"
    }
  ]
}

```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Stakeholder is successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or stakeholder could not be found.

Error codes

HTTP CODE	CODE
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound
<b>404</b>	Stakeholder.NotFoundInPackage

# Delete stakeholder by ID

## Description

This call allows you to delete a specified stakeholder from a package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/stakeholders/{stakeholderId}

## HTTP method

DELETE

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package to which the stakeholder belongs.	String (GUID)
StakeholderId*	Unique identifier of the stakeholder you want to delete.	String (GUID)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
204 No content	Stakeholder was successfully deleted.
401 Unauthorized	Caller is unauthorized to perform this operation.
404 Not found	Target package or stakeholder could not be found.
409 Conflict	Stakeholder could not be deleted.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
401	Authentication failed, username or password incorrect
404	Package.NotFound
409	Package.InvalidStatus
404	Stakeholder.NotFound
409	Stakeholder.CannotRemoveUndecided

# Actors

The **Actors** collection contains all resources and methods for creating and managing actors.

## What is an actor?

An actor is a single action a stakeholder needs to take on a package.

There are four types of actors: **formfillers**, **approvers**, **signers** and **receivers**.

## Actor methods

- Create actor
- Get actors
- Get actor by ID
- Delete actor by ID

# Create actor

## Description

The POST actors call creates an actor and adds it to the stakeholder you specify.

An actor is a single action that a stakeholder must do on a package.

The different actor types are:

- **FormFiller**

A FormFiller must fill in TextBoxFields and/or CheckBoxFields on a document. FormFiller actors do not sign or approve documents.

- **Approver** An Approver must approve the document before it is sent to any signer. Note that the approval may occur before or after the filling of forms.
- **Signer** A Signer must sign the document.
- **Receiver** A Receiver receives a a copy of the document when it has been signed by all signers. A Receiver does not take action on the document.

## Important:

- All Approvers must be added to the first step of the process. This is because approvers must first approve a package before it is sent to any signers. All receivers must be added to the last step of the process, since a package is only sent to receivers once it has been signed by all signers.
- If you do not define the process steps manually, all **FormFiller** actors are also added to the first step, together with the Approvers. As a result, the approval action may happen before, during or after the filling of forms.

**Attention:** All other process steps may only contain one type of actor. It's not supported to mix actor types within an array.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/stakeholders/{stakeholderId}/actors

## HTTP Method

POST

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the actor must be added.	String (GUID)
StakeholderId*	Unique identifier of the stakeholder to which the actor must be added.	String (GUID)

## Request parameters

The Request parameters are the same for all four types of actors, with the exception of **elements**. SigningField elements can only be used for **Signers**, since the other actor types do not sign documents. CheckBoxField elements and TextBoxField elements can only be used for **FormFillers**, since that is the only actor type that fills out forms.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Type*	The type of action that must be done on the package. The possible values are: <b>FormFiller</b> , <b>Approver</b> , <b>Signer</b> and <b>Receiver</b> .	String (enum)

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
SuppressNotifications	This parameter determines whether notifications are sent to the actor, informing them about the package. By default its value is false, which means notifications are sent.	Boolean
Elements	The elements to which the actor will be linked. Currently, three element types are supported: SigningField, TextBoxField and CheckBoxField. This parameter does not apply to ReceiverActors, since this actor type does not take action on documents. The elements parameters are identical to those of the <a href="#">Create element</a> call and are described there.	Array of objects
RedirectUrl	Url to which the stakeholder is redirected after completing this action. See the <b>RedirectUrl</b> section below for more details.	String (url)
RedirectType	<p>This parameter defines when exactly signer actors are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>- The RedirectType parameter must <i>only</i> be used for actors of the type <b>Signer</b>. It is not allowed when the actor is of the type <b>FormFiller</b>, <b>Approver</b> or <b>Receiver</b>.</li> <li>- The RedirectType parameter is forbidden if no <b>RedirectUrl</b> has been set.</li> </ul> <p>Possible values: <b>AfterSession</b> (default), <b>AfterCompletion</b>, <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, actors are redirected <b>AfterSession</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For actors to be redirected immediately, enter <b>Immediately</b> as value.</p> <p>For actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value. To make sure the final actor is only redirected after the entire package is completed, enter <b>AfterCompletion</b> as value.</p> <p><b>Note:</b> when <b>AfterCompletion</b> is set for multiple actors, it will only be applied to the final actor. For the other actors, the default <b>AfterSession</b> will be used.</p>	String (enum)
BackButtonUrl	URL to which the end user is sent after pressing back button. This parameter does not apply to receivers.	String (url)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Id	Unique identifier of the actor.	String (GUID)
Type	The type of action that must be done on the package. The possible values are: <b>FormFiller</b> , <b>Approver</b> , <b>Signer</b> and <b>Receiver</b> .	String (enum)
Status	Status of an action. Available statuses are: Draft, Waiting, Available, InProgress, Failed, Finished, Rejected, Skipped.	String (enum)
SuppressNotifications	This parameter determines whether notifications are sent to the actor, informing them about the package. By default its value is false, which means notifications are sent.	Boolean
Result	Result of a completed action. This parameter does not apply to receivers.	Object

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Elements	The elements to which the actor will be linked. This parameter only applies to signers.	Array of objects
RedirectUrl	Url to which the stakeholder is redirected after completing this action. This parameter does not apply to receivers. See the <b>RedirectUrl Details</b> section below for more information.	String (url)
RedirectType	<p>This parameter defines when exactly actors of the type <b>Signer</b> are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p>Possible values: <b>AfterSession</b> (default), <b>AfterCompletion</b>, <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, signer actors are redirected <b>AfterSession</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For signer actors to be redirected immediately, enter <b>Immediately</b> as value.</p> <p>For signer actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value. To make sure the final actor is only redirected after the entire package is completed, enter <b>AfterCompletion</b> as value.</p> <p><b>Note:</b> when <b>AfterCompletion</b> is set for multiple signer actors, it will only be applied to the final actor. For the other signer actors, the default <b>AfterSession</b> will be used.</p>	String (enum)
BackButtonUrl	URL to which the end user is sent after pressing back button. This parameter does not apply to receivers.	String (url)
OneOf	links memberLinks	Object

## Result

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
CompletedBy	Email address.	String (email)
VerifiedName	The actor's verified name as mentioned on the signing certificate or signing service.	String
CompletedDate	Date when the action was completed.	String (date-time)
SigningMethod	<p>Signing method the actor used to sign the document.</p> <p>This parameter only applies to signers.</p>	String (enum)
RejectReason	Reason why a document was rejected.	String

## links / memberLinks

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Links	Links to interact with the package for this person stakeholder.	String (url)
MemberLinks	Link to interact with the package for this group stakeholder.	String (url)

memberLinks

PARAMETER	CONTENT / DESCRIPTION	TYPE
Email	Email address of this member of the group stakeholder.	String (email)
Link	Link to interact with this member	String (url)

### Example request

```
{
  "Type": "signer",
  "SuppressNotifications": false,
  "Elements": [
    {
      "Id": "5a5365e1-3780-47c7-9657-a1496be6ea5e"
    },
    {
      "Type": "signingField",
      "DocumentId": "c05866bc-6b5a-4b5b-a4c1-b76895469f6f",
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      }
    }
  ]
}
```

### Example response

(no example available)

### Response codes

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>
<b>201 OK</b>	Actor was created
<b>400 Bad request</b>	Request failed validation
<b>401 Unauthorized</b>	Caller is unauthorized to do this operation
<b>404 Not found</b>	Target package or stakeholder could not be found
<b>409 Conflict</b>	Actor could not be added to the stakeholder

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldIsMissing
<b>400</b>	Request.OneOfFieldsIsMissing

<u>HTTP CODE</u>	<u>CODE</u>
400	Request.FieldMaxLength
400	SigningType.Invalid
400	Request.FieldMinimumValue
400	SigningField.InvalidWidthCoordinate
400	SigningField.InvalidHeightCoordinate
400	FormField.InvalidWidthCoordinate
400	FormField.InvalidWidthCoordinate
400	Actor.TypeCantHaveFormFields
400	Actor.InvalidOrder
400	Url.Invalid
401	Authentication failed, username or password incorrect
409	Package.InvalidStatus
409	Stakeholder.NotFound
409	Stakeholder.TypeInvalid
409	Stakeholder.NotFoundInPackage
409	Actor.TypeInvalid

### RedirectUrl Details

The **RedirectUrl** is used to redirect the end user to the originating web application pointed to by that URL. The redirect occurs immediately after the user has signed or rejected.

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 Create actor call and is appended with the following query parameters:

- SessionID (unique identifier of the package signing session, i.e. ActorId)
- ExternalReference (as found in the ExternalReference parameter of the Create Stakeholder call)
- Status (SIGNED, REJECTED or INVALIDTOKEN)
- PackageExternalReference (as found in the **ExternalPackageReference** parameter of the Create 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 RedirectUrl will be:

[https://myserver.example.org/rental/services/testredirect?\\*\\*SessionID\\*\\*=5a2aff04-3cfa-4278-9480-64ac39f74734&\\*\\*ExternalReference\\*\\*=user1@example.org&\\*\\*Status\\*\\*=REJECTED&\\*\\*PackageExternalReference\\*\\*=dossier-3592](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 RedirectUrl because the redirection happens *client-side*! Either establish a second secure channel by means of the CallbackUrl, or verify through session state that the end user returned from the correct session id. Afterwards a call to Get Package Status 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. By default, the signer has to wait until all documents are signed before being redirected. Since eSignatures 5.4, it is possible to configure when the redirect happens (See the RedirectType parameter info above).

# Get actors

## Description

This call retrieves all actors of a specified stakeholder.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/stakeholders/{stakeholderId}/actors

## HTTP method

GET

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package whose actors you want to retrieve.	String (GUID)
StakeholderId*	Unique identifier of the stakeholder whose actors you want to retrieve.	String (GUID)

## Response parameters

The Response parameters are identical to the ones of the [Create actor](#) call and are described there.

## Example response

```
[
  {
    "Type": "approver",
    "SuppressNotifications": false,
    "Status": "finished",
    "RedirectUrl": "https://redirecttothis.test",
    "Result": {
      "CompletedBy": {
        "Email": "john@doe.test"
      },
      "CompletedDate": "2020-01-24T09:24:13+0000"
    }
  },
  {
    "Type": "signer",
    "SuppressNotifications": false,
    "Status": "available",
    "Elements": [
      {
        "Id": "00000000-0000-0000-0000-000000000000",
        "ActorId": "00000000-0000-0000-0000-000000000000",
        "Type": "signingField",
        "Location": {
          "Page": 2,
          "Top": 200,
          "Left": 200
        },
        "Dimensions": {
          "Width": 200,
          "Height": 200
        },
        "ExternalReference": "my_signingfield",
        "SigningMethods": [
          "manual"
        ],
        "LegalNotice": {
          "Text": "Read and approved"
        }
      }
    ],
    "Links": [
      "https://sign.test/package/1234"
    ],
    "RedirectUrl": "https://redirecttothis.test"
  }
]
```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	All actors are successfully retrieved.
401 Unauthorized	Caller is unauthorized to perform this operation.
404 Not found	Target package or stakeholder could not be found.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>

<u>HTTP CODE</u>	<u>CODE</u>
401	Authentication failed, username or password incorrect
404	Package.NotFound
404	Stakeholder.NotFoundInPackage

# Get actor by ID

## Description

This call allows you to retrieve an actor by its ID.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/stakeholders/{stakeholderId}/actors/{actorId}

## HTTP method

GET

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the actor belongs.	String (GUID)
StakeholderId*	Unique identifier of the package to which the actor belongs.	String (GUID)
ActorId*	Unique identifier of the actor you want to retrieve.	String (GUID)

## Response parameters

The Response parameter are identical to the ones of the [Create actor](#) call and are described there.

## Example response

```
{
  "Type": "signer",
  "Status": "available",
  "Elements": {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "signingField",
    "Location": {
      "Page": 2,
      "Top": 250,
      "Left": 250
    },
    "Dimensions": {
      "Width": 230,
      "Height": 230
    }
  }
}
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	Actor is successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package, stakeholder or actor could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
401	Authentication failed, username or password incorrect
404	Package.NotFound
404	Stakeholder.NotFoundInPackage
404	Actor.NotFound

# Delete actor by ID

## Description

This call allows you to delete a specified actor from a package.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/stakeholders/{stakeholderId}/actors/{actorId}

## HTTP Method

Delete

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package to which the stakeholder belongs.	String (GUID)
StakeholderId*	Unique identifier of the stakeholder to which the actor belongs.	String (GUID)
ActorId*	Unique identifier of the actor you want to delete.	String (GUID)

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Actor was successfully deleted.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package, stakeholder or actor could not be found.
<b>409 Conflict</b>	Actor could not be deleted.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus
<b>404</b>	Stakeholder.NotFound
<b>404</b>	Actor.NotFound
<b>409</b>	Actor.CannotRemoveUndecided

## 10. Process

The **Process** collection contains all resources and methods to set up a package's approval/signing process. Its allows you to set up parallel, sequential and complex signing flows.

### How does a process work?

A package's process defines which stakeholder will do their action at which moment in time.

A process may consist of multiple process steps. A process step is a set of actions that may be done in parallel. For instance, approve in process step 1, fill out forms in step 2, sign in step 3 and receive in step 4. The approval step may occur before or after the filling of forms, as long as they are both prior to signing. Receivers must be placed in the last step, since they receive the signed documents after the signers have completed their tasks.

### Process methods

- Add process step
- Add an action to a process step
- Update process steps
- Get all process steps
- Get process step actions by StepIndex
- Get current process step actions
- Delete a process step and all its actions



# Add process step

## Description

This call adds a process step to a package's process. A process step consists of an array of actors.

Note that all approvers must be grouped in a process step that precedes the signing step. The same goes for all FormFillers. The form filling step may precede the approval step, but may also succeed it. All receivers must be grouped in the last step. It's not supported to mix different actor types within the same process step.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process

## HTTP method

POST

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the process step must be added.	String (GUID)

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
StakeholderId*	Unique identifier of the stakeholder.	String (GUID)
OneOf	CreateSignerActor CreateApproverActor CreateFormFillerActor CreateReceiverActor	Object

The Request parameters are the same for all four types of actors, with the exception of **elements**. SigningField elements can only be used for **Signers**, since the other actor types do not sign documents. CheckBoxField elements and TextBoxField elements can only be used for **FormFillers**, since that is the only actor type that fills out forms.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Type*	The type of action that must be done on the package. The possible values are: <b>FormFiller</b> , <b>Approver</b> , <b>Signer</b> and <b>Receiver</b> .	String (enum)
ExternalReference	External reference to this resource given by a calling application. This parameter is not used by eSignatures itself.  Maximum length: 256	String
SuppressNotifications	This parameter determines whether notifications are sent to the actor, informing them about the package. By default its value is false, which means notifications are sent.	Boolean
Elements	The elements to which the actor will be linked. The elements parameters are identical to those of the <a href="#">Create element</a> call and are described there.	Array of objects
RedirectUrl	Url to which the stakeholder is redirected after completing this action	String (url)

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
StakeholderId	The stakeholder's unique identifier.	String (GUID)
Type	The type of action that must be done on the package. The possible values are: <b>approver</b> , <b>formfiller</b> , <b>signer</b> and <b>receiver</b> .	String (enum)
Status	Status of an action. Available statuses are: draft, waiting, available, inProgress, failed, finished, rejected, kipped.	String (enum)
SuppressNotifications	This parameter determines whether notifications are sent to the actor, informing them about the package. By default its value is false, which means notifications are sent.	Boolean
Result	Result of a completed action. This parameter does not apply to receivers.	Object
Elements	The elements to which the actor will be linked. This parameter only applies to signers and formfillers.	Array of objects
RedirectUrl	Url to which the stakeholder is redirected after completing this action. This parameter does not apply to receivers. See the <b>RedirectUrl Details</b> section below for more information.	String (url)
RedirectType	<p>This parameter defines when exactly signer actors are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>-The RedirectType parameter must <i>only</i> be used for actors of the type <b>Signer</b>. It is not allowed when the actor is of the type <b>FormFiller</b>, <b>Approver</b> or <b>Receiver</b>.</li> <li>- The RedirectType parameter is forbidden if no <b>RedirectUrl</b> has been set.</li> </ul> <p>Possible values: <b>AfterSession</b> (default), <b>AfterCompletion</b>, <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, actors are redirected <b>AfterSession</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For actors to be redirected immediately, enter <b>Immediately</b> as value.</p> <p>For actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value. To make sure the final actor is only redirected after the entire package is completed, enter <b>AfterCompletion</b> as value.</p> <p><b>Note:</b> when <b>AfterCompletion</b> is set for multiple actors, it will only be applied to the final actor. For the other actors, the default <b>AfterSession</b> will be used.</p>	String (enum)
BackButtonUrl	URL to which the end user is sent after pressing back button. This parameter does not apply to receivers.	String (url)
OneOf	links memberLinks	Object

## Result

<b><u>PARAMETER</u></b>	<b><u>CONTENT / DESCRIPTION</u></b>	<b><u>TYPE</u></b>
CompletedBy	Email address.	String (email)
VerifiedName	The actor's verified name as mentioned on the signing certificate or signing service.	String
CompletedDate	Date when the action was completed.	String (date-time)
Identity	Identity data returned when signer used eID or OpenID Connect to sign.	TBD
SigningMethod	Signing method the actor used to sign the document.  Only applies to signers.	String (enum)
RejectReason	Reason why a document was rejected.	String

## Links / MemberLinks

<b>PARAMETER</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
Links	Links to interact with the package for this person stakeholder.	String (url)
MemberLinks	Link to interact with the package for this group stakeholder.	Array of objects

## MemberLinks

<b>PARAMETER</b>	<b>CONTENT / DESCRIPTION</b>	<b>TYPE</b>
Email	Email address of this member of the group stakeholder.	String (email)
Link	Link to interact with this member	String (url)

## Example request

```
[
  {
    "StakeholderId": "00000000-0000-0000-0000-000000000000",
    "Type": "signer",
    "SuppressNotifications": false,
    "Elements": [
      {
        "Type": "SigningField",
        "Location": {
          "Page": 2,
          "Top": 200,
          "Left": 200
        },
        "Dimensions": {
          "Width": 200,
          "Height": 200
        },
        "SigningMethods": [
          "manual",
          "beid"
        ]
      }
    ]
  },
  {
    "StakeholderId": "00000000-0000-0000-0000-000000000001",
    "Type": "signer",
    "Elements": [
      {
        "Id": "00000000-0000-0000-0000-000000000000"
      }
    ]
  }
]
```

### Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000003",
    "StakeholderId": "00000000-0000-0000-0000-000000000003",
    "Type": "signer",
    "Status": "available",
    "Links": [
      "https://completeyouraction.test"
    ]
  },
  {
    "Id": "00000000-0000-0000-0000-000000000002",
    "StakeholderId": "00000000-0000-0000-0000-000000000002",
    "Type": "signer",
    "Status": "available",
    "MemberLinks": [
      {
        "Email": "john@doe.test",
        "Link": "https://alsocompleteyouraction.test"
      },
      {
        "Email": "zu@li.test",
        "Link": "https://dothething.test"
      }
    ]
  }
]
```

## Response codes

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>
<b>201 Created</b>	Process step was successfully created.
<b>401 Unauthorized</b>	Caller is unauthorized to do this operation.
<b>404 Not found</b>	Target package could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldsMissing
<b>400</b>	Request.OneOfFieldsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	SigningType.Invalid
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	FormField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	FormField.InvalidHeightCoordinate
<b>400</b>	Url.Invalid
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>409</b>	ProcessStep.MixedTypes
<b>404</b>	Stakeholder.NotFound
<b>409</b>	Document.DataMissing
<b>404</b>	Actor.NotFound
<b>404</b>	SigningField.NotFound

## RedirectUrl Details

The **RedirectUrl** is used to redirect the end user to the originating web application pointed to by that URL. The redirect occurs immediately after the user has signed or rejected.

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 Create actor call and is appended with the following query parameters:

- SessionID (unique identifier of the package signing session, i.e. ActorId)
- ExternalReference (as found in the ExternalReference parameter of the Create Stakeholder call)
- Status (SIGNED, REJECTED or INVALIDTOKEN)
- PackageExternalReference (as found in the **ExternalPackageReference** parameter of the Create 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 RedirectUrl will be:

[https://myserver.example.org/rental/services/testredirect?\\*\\*SessionID\\*\\*=5a2aff04-3cfa-4278-9480-64ac39f74734&\\*\\*ExternalReference\\*\\*=user1@example.org&\\*\\*Status\\*\\*=REJECTED&\\*\\*PackageExternalReference\\*\\*=dossier-3592](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 RedirectUrl because the redirection happens *client-side*! Either establish a second secure channel by means of the CallbackUrl, or verify through session state that the end user returned from the correct session id. Afterwards a call to Get Package Status 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. By default, the signer has to wait until all documents are signed before being redirected. Since eSignatures 5.4, it is possible to configure when the redirect happens (See the RedirectType parameter info above).

# Add process step action

## Description

This call adds an actor to a process step, by means of StepIndex. The StepIndex indicates at which step of the process the action must be done.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process/{stepIndex}

## HTTP method

POST

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package to which the process step must be added.	String (GUID)
StepIndex*	Step at which the action must be done.	Integer

## Request parameters

The request parameters are identical to those of the [Add process step](#) call and are described there.

## Response parameters

The response parameters are identical to those of the [Add process step](#) call and are described there.

## Example request

```
{
  "Type": "signer",
  "SuppressNotifications": false,
  "Elements": [
    {
      "Id": "5a5365e1-3780-47c7-9657-a1496be6ea5e"
    },
    {
      "Type": "signingField",
      "DocumentId": "c05866bc-6b5a-4b5b-a4c1-b76895469f6f",
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      }
    }
  ]
}
```

## Example response

```

{
  "Type": "signer",
  "Status": "available",
  "Elements": {
    "Id": "00000000-0000-0000-0000-000000000000",
    "Type": "signingField",
    "Location": {
      "Page": 2,
      "Top": 250,
      "Left": 250
    },
    "Dimensions": {
      "Width": 230,
      "Height": 230
    }
  }
}

```

### Response codes

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>
<b>201 Created</b>	Actor was successfully added to the process step
<b>401 Unauthorized</b>	Caller is unauthorized to do this operation
<b>404 Not found</b>	Target package or process step could not be found
<b>409 Conflict</b>	Actor could not be added to the process step

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	LegalNotice.NameAndTextUsedSimultaneously
<b>400</b>	Request.RequiredFieldsMissing
<b>400</b>	Request.OneOfFieldsMissing
<b>400</b>	Request.FieldMaxLength
<b>400</b>	SigningType.Invalid
<b>400</b>	Request.FieldMinimumValue
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	FormField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	FormField.InvalidHeightCoordinate
<b>400</b>	Url.Invalid



<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	Actor.InvalidOrder
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>409</b>	Document.DataMissing
<b>404</b>	Actor.NotFound
<b>404</b>	SigningField.NotFound
<b>404</b>	FormField.NotFound
<b>409</b>	Package.InvalidStatus
<b>409</b>	ProcessStep.MixedTypes
<b>404</b>	Stakeholder.NotFound

# Update process steps

## Description

This call overwrites all actors in a package using a 2D array, where every set of actors are actions that can be done in parallel.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process

## HTTP method

PUT

## Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose actors must be updated.	String (GUID)

## Request parameters

The request parameters are identical to those of the [Add process step](#) call and are described there.

## Example request

```
[
  [
    {
      "Type": "approver",
      "StakeholderId": "00000000-0000-0000-0000-000000000000"
    }
  ],
  [
    {
      "Type": "signer",
      "StakeholderId": "00000000-0000-0000-0000-000000000000",
      "Elements": [
        {
          "Id": "00000000-0000-0000-0000-000000000000"
        },
        {
          "Type": "signingField",
          "Location": {
            "Page": 2,
            "Top": 200,
            "Left": 200,
            "Dimensions": {
              "Width": 200,
              "Height": 200
            }
          }
        }
      ]
    }
  ]
],
[
  {
    "Type": "receiver",
    "StakeholderId": "00000000-0000-0000-0000-000000000000"
  }
]
]
```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Process steps were updated successfully.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.

# Get all process steps

## Description

This call retrieves all actions that must be done on a specified package.

The actions are displayed in a 2D array where every set of actors are actions that can be done in parallel.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package whose process steps you want to retrieve.	String (GUID)

## Response parameters

The Response parameters are identical to the ones of the [Add process step](#) call and are described there.

## Example response

```
[
  [
    {
      "Id": "00000000-0000-0000-0000-000000000001",
      "StakeholderId": "00000000-0000-0000-0000-000000000001",
      "Type": "approver"
    }
  ],
  [
    {
      "Id": "00000000-0000-0000-0000-000000000003",
      "StakeholderId": "00000000-0000-0000-0000-000000000003",
      "Type": "ormFiller",
      "Elements": [
        {
          "Type": "CheckBoxField",
          "Name": "MyCheckBox1",
          "ToolTipLabel": "Check me!",
          "IsRequired": true,
          "DefaultValue": true,
          "Location": {
            "Page": 3,
            "Top": 200,
            "Left": 200
          },
          "Dimensions": {
            "Width": 10,
            "Height": 10
          }
        }
      ]
    }
  ]
],
[
  {
    "Id": "00000000-0000-0000-0000-000000000003",
    "StakeholderId": "00000000-0000-0000-0000-000000000003",
    "Type": "signer",
    "Elements": [
      {
        "Type": "Text",
        "Text": "Text"
      }
    ]
  }
]
```

```

        "Type": "signingField",
        "ExternalReference": "myManualField",
        "SigningMethods": [
            "manual"
        ],
        "Location": {
            "Page": 1,
            "Top": 200,
            "Left": 200
        },
        "Dimensions": {
            "Width": 200,
            "Height": 200
        }
    }
}
],
{
    "Id": "00000000-0000-0000-0000-000000000002",
    "StakeholderId": "00000000-0000-0000-0000-000000000002",
    "Type": "signer",
    "Elements": [
        {
            "Type": "signingField",
            "SigningMethods": [
                "beid"
            ],
            "Location": {
                "Page": 2,
                "Top": 200,
                "Left": 200
            },
            "Dimensions": {
                "Width": 200,
                "Height": 200
            }
        }
    ]
}
],
[
    {
        "Id": "00000000-0000-0000-0000-000000000004",
        "StakeholderId": "00000000-0000-0000-0000-000000000004",
        "Type": "signer",
        "Elements": [
            {
                "Type": "signingField",
                "ExternalReference": "myManualField",
                "SigningMethods": [
                    "itsme"
                ],
                "Location": {
                    "Page": 1,
                    "Top": 400,
                    "Left": 400
                },
                "Dimensions": {
                    "Width": 200,
                    "Height": 200
                }
            }
        ]
    }
]
}
],
[

```

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000005",
    "StakeholderId": "00000000-0000-0000-0000-000000000005",
    "Type": "receiver"
  }
]
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All process steps are successfully retrieved.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package or process step could not be found.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound

# Get process step actions

## Description

This call gets a particular set of actions that can be completed in parallel from any specific process step

The process step of which the actions must be retrieved is specified by the StepIndex parameter.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process/{stepIndex}

## HTTP method

GET

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	Unique identifier of the package wo which the process step belongs.	String (GUID)
StepIndex*	Index of the process step that must be retrieved.	Integer

## Response parameters

The Response parameters are identical to the ones of the [Add process step action](#) call and are described there.

## Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000003",
    "StakeholderId": "00000000-0000-0000-0000-000000000003",
    "Type": "signer",
    "Status": "available",
    "Links": [
      "https://completeyouraction.test"
    ]
  },
  {
    "Id": "00000000-0000-0000-0000-000000000002",
    "StakeholderId": "00000000-0000-0000-0000-000000000002",
    "Type": "signer",
    "Status": "available",
    "MemberLinks": [
      {
        "Email": "john@doe.test",
        "Link": "https://alsocompleteyouraction.test"
      },
      {
        "Email": "zu@li.test",
        "Link": "https://dothething.test"
      }
    ]
  }
]
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All process steps are successfully retrieved.

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package could not be found.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound



# Get process step actions of current process step

## Description

This call gets all actions that can be completed in parallel of the current process step, i.e. the process step that is currently in progress.

In the example below, you'll see that the actions of the signing step are retrieved.

## URL

https://[servername]:[port]/esig/webportalapi/v4//packages/{packageld}/process/current

## HTTP method

GET

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Packageld*	Unique identifier of the package to which the process step belongs.	String (GUID)

## Response parameters

The Response parameters are identical to the ones of the [Add process step action](#) call and are described there.

## Example response

```
[
  {
    "Id": "00000000-0000-0000-0000-000000000003",
    "StakeholderId": "00000000-0000-0000-0000-000000000003",
    "Type": "signer",
    "Status": "available",
    "Links": [
      "https://completeyouraction.test"
    ]
  },
  {
    "Id": "00000000-0000-0000-0000-000000000002",
    "StakeholderId": "00000000-0000-0000-0000-000000000002",
    "Type": "signer",
    "Status": "available",
    "MemberLinks": [
      {
        "Email": "john@doe.test",
        "Link": "https://alsocompleteyouraction.test"
      },
      {
        "Email": "zu@li.test",
        "Link": "https://dothething.test"
      }
    ]
  }
]
```

## Response codes

RESPONSE STATUS CODE	DESCRIPTION
200 OK	All actions of the current process steps are successfully retrieved.

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Package or process step could not be found.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>404</b>	Package.NotFound

## Delete process step and its actions

This call allows you to delete a process step and its actions from a package.

### URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/process/{stepIndex}

### HTTP Method

DELETE

### Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
PackageId*	Unique identifier of the package whose process step must be deleted.	String (GUID)
StepIndex*	Index of the process step that must be deleted.	Integer

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	Process step was successfully deleted.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	Target package or process step could not be found.
<b>409 Conflict</b>	Process step could not be deleted.

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Actor.NotFound
<b>404</b>	Package.NotFound

# 11. Configuration

- [Get Document Groups](#)
- [Get Version](#)
- [Get Themes](#)
- [Get Contact Groups](#)
- [Get Enabled Languages](#)

# Get DocumentGroups

## 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.

## URL

https://[servername]:[port]/esig/webportalapi/v4/documentgroups

## HTTP Method

GET

## MIME Type

application/json

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>DocumentGroups</b>	List of all document groups	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>DocumentGroups (array of objects)</b>	Document group details	Array of objects
<b>Name</b>	Name or description of the document group	String
<b>Code</b>	A unique value identifying each document group	String

## Example response

```
{
  "DocumentGroups" : [{
    "Name" : "My Documents",
    "Code" : "00001"
  }, {
    "Name" : "HR",
    "Code" : "00002"
  }, ]
}
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The documentgroups gets returned successfully.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No error messages possible

# Get Version

## Description

This call checks the version and build number of eSignatures.

## URL

https://[servername]:[port]/esig/webportalapi/v4/version

## HTTP Method

GET

## MIME Type

application/json

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ProductVersion</b>	Number of the version	String
<b>FileVersion</b>	Number of the file (build)	String

## Example response

```
{
  "ProductVersion": "5.5.0",
  "FileVersion": "5.5.0.48326.02"
}
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The version and build number get returned successfully.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>None</b>	No error messages possible

# Get Themes

## Description

This call retrieves all the Themes and their code that are currently created in the Configuration Index.

The theme code can be entered as Request parameter when creating Packages and Instant Packages. The packages in question will then have the requested theming.

## URL

https://[servername]:[port]/esig/webportalapi/v4/themes

## HTTP Method

GET

## MIME Type

application/json

## Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
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

## Response parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
Items	Optional	List of themes	Array of objects
<u>PARAMETER</u>		<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
Items (array of objects)		Array, 0..*	Array of objects
Name		Name of the theme	String
ThemeCode		Unique id of the theme	String

## Example response



```
{
  "ContinuationToken": "1",
  "Items": [
    {
      "Name": "Connective Theme",
      "Code": "0000"
    },
    {
      "Name": "System Theme",
      "Code": "00001"
    },
    {
      "Name": "My theme",
      "Code": "00003"
    },
  ],
  "MaxQuantity": 20,
  "Total": 3
}
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The themes list gets returned successfully.
400 Bad request	A request parameter was invalid

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

# Get ContactGroups

## Description

This call retrieves all the contact groups and their code that are currently created in the eSignatures WebPortal.

The ContactGroup code can be entered as Request parameter when doing a Create package call. Any member of the requested contact group will be able to sign the package for the entire group.

## URL

https://[servername]:[port]/esig/webportalapi/v4/contactgroups

## HTTP Method

GET

## MIME Type

application/json

## Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
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

## Response parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
Items	Optional	List of contact groups	Array of objects
<u>PARAMETER</u>		<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
Items (array of objects)		Array, 0..*	Array of objects
Name		Name of the contact group	String
ContactGroupCode		Unique id of the contact group	String

## Example response

```
{
  "ContinuationToken": "1",
  "Items": [
    {
      "Code": "00001",
      "Name": "Personal"
    },
    {
      "Code": "00002",
      "Name": "SharedContacts"
    },
    {
      "Code": "00003",
      "Name": "OldGroup"
    }
  ],
  "MaxQuantity": 20,
  "Total": 3
}
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The contact groups list gets returned successfully.
400 Bad request	A request parameter was invalid.

## Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

# Get Enabled Languages

## Description

This call retrieves the list of languages that have been enabled in the eSignatures Configuration Index and are thus available in the eSignatures Portal.

This call is useful for integrators who need to know which languages are currently enabled in a given eSignatures installation.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/languages](https://[servername]:[port]/esig/webportalapi/v4/languages)

## HTTP Method

GET

## MIME Type

application/json

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>EnabledLanguages</b>	List of enabled languages	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>EnabledLanguages (array of objects)</b>	Array, 1..*	
<b>DisplayName</b>	DisplayName of the language as defined in the Config Index.	String
<b>NativeName</b>	NativeName of the language as defined in the Config Index. The NativeName is the name of the language in its actual language. E.g. français for French.	String
<b>IsoCultureCode</b>	ISO 639-1 language code of the language.	String

## Example response

```
[
  {
    "EnabledLanguages": [
      {
        "DisplayName": "Dutch",
        "NativeName": "Nederlands",
        "IsoCultureCode": "nl"
      }
    ]
  }
]
```

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The EnabledLanguages list gets returned successfully.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

## 8. Audit Proofs

In previous versions of eSignatures, the Audit proofs feature had a significant impact on the eSignatures database. As of eSignatures 5.3, the Audit proof files are stored in the repository. This reduces the load on the database but obviously increases the load on the repository. The bigger the documents, the more space will be used. The impact grows exponentially with bigger documents. So, make sure the repository has sufficient free space. How much space is needed really depends on the size of your documents and on the retention time in the repository. Consider the following rule of thumb when using Audit proofs: The Audit proofs will have a size of 1.5x of the original document. If the **Intermediate States Saved** setting is enabled in the Configuration Index, this size increase is added for every signature placed on the document.

## 12.1 Limitations

The Audit proof is available for documents and packages created in API v3, API v4 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:** In previous versions of eSignatures, the Audit proofs feature had a *significant* impact on the eSignatures database. Since eSignatures 5.4, the Audit proof files are stored on the storage drive of the server that hosts the data (configurable in the **File Storage Settings** of the Configuration Index). This reduces the load on the database but obviously increases the load on the storage drive. The bigger the documents, the more space will be used. The impact grows exponentially with bigger documents. So, make sure the storage has sufficient free space. How much space is needed largely depends on the size of your documents and on the retention time in the repository. Consider the following rule of thumb when using Audit proofs: The Audit proofs will have a size of 1.5x of the original document. If the Intermediate States Saved setting is enabled in the Configuration Index, this size increase is added for every signature placed on the document.

**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.

# Get Package or Document Audit Proofs Xml

## 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.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageId}/auditproof/download](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/auditproof/download)

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageId}/auditproof/download/{documentId}](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/auditproof/download/{documentId})

## HTTP Method

GET

## MIME Type

application/xml

## 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.

## Response codes

<b><u>RESPONSE STATUS CODE</u></b>	<b><u>DESCRIPTION</u></b>
<b>200 OK</b>	The Audit proofs xml is returned.
<b>404 Not Found</b>	The documentation/package id could not be found.
<b>409 Conflict</b>	The package is not fully signed.



## 12.3 Add Proof from External Source

### 12.3.1 Description

This call allows API users to add extra proofs from an external source 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).

### 12.3.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageId}/auditproof/proofs](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageId}/auditproof/proofs)

### 12.3.3 HTTP Method

POST

### 12.3.4 MIME Type

application/json

### 12.3.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Content</b>	Mandatory	The actual content of the proof.	Base64 string
<b>LocationId</b>	Mandatory	Location of the signature for which the proof content was generated.	Guid
<b>Name</b>	Mandatory	Name of the proof.	String
<b>Type</b>	Mandatory	A machine-readable "type" key. Can be freely chosen.	String
<b>Description</b>	Optional	Brief human-readable description of the proof.	String
<b>IpAddress</b>	Optional	IP address of the end user of the external source for which the proof was added.	String

### 12.3.6 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"
}
```

### 12.3.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	The proof was created for the requested location.
<b>400 Bad Request</b>	The request contained parameters which could not be accepted.
<b>404 Not Found</b>	The documentId or the LocationId could not be found.
<b>409 Conflict</b>	The proof could not be added due to some other reason.

## 12.4 Get Package or Document Correlation Audit Proofs Xml

### 12.4.1 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.

### 12.4.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packagecorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/esig/webportalapi/v4/packagecorrelations/{correlationId}/auditproof/download)

[https://\[servername\]:\[port\]/esig/webportalapi/v4/documentcorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/esig/webportalapi/v4/documentcorrelations/{correlationId}/auditproof/download)

### 12.4.3 HTTP Method

GET

### 12.4.4 MIME Type

application/xml

### 12.4.5 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.

### 12.4.6 Response codes

<b><u>RESPONSE STATUS CODE</u></b>	<b><u>DESCRIPTION</u></b>
<b>200 OK</b>	The Audit proofs xml is returned.
<b>404 Not Found</b>	The document/package id could not be found.
<b>409 Conflict</b>	The package is not fully signed.

# 12.5 Get Audit Proof Info

## Description

This calls allows to you retrieve the audit proof info of packages.

Currently, you can retrieve the audit proof info of DELETED packages.

You can narrow the search for specific audit proof info by adding the optional request parameters listed below.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/auditproofinfo](https://[servername]:[port]/esig/webportalapi/v4/packages/auditproofinfo)

## HTTP Method

GET

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Status*	Status of the package. Currently, you can only retrieve the audit proof info of packages that have the status DELETED.	String (enum)
Packageld	Unique identifier of the package whose audit proof info you want to retrieve.	String
CorrelationId	Identifier that correlates this package to other resources in the proofs system.	String (GUID)
CreatedBefore	Gets only the audit proof info of packages created before this date. ISO 8601 date format.	String (date-time)
CreatedAfter	Gets only the audit proof info of packages created after this date. ISO 8601 date format.	String (date-time)
DeletedBefore	Gets only the audit proof info of the packages deleted before this date. ISO 8601 date format.	String (date-time)
DeletedAfter	Gets only the audit proof info of the packages deleted after this date. ISO 8601 date format.	String (date-time)
ExpiresBefore	Gets only the audit proof info of the packages that expire before this date. ISO 8601 date format.	String (date-time)
ExpiresAfter	Gets only the audit proof info of the packages that expire after this date. ISO 8601 date format.	String (date-time)
SortField	Field on which the result is sorted.	String
SortOrder	Determines how the result is sorted. Available values are asc, desc. Default value: desc	String (enum)

<b><u>PARAMETER</u></b>	<b><u>CONTENT / DESCRIPTION</u></b>	<b><u>TYPE</u></b>
PageSize	Maximum number of records in the response  Default value is 20.	Integer
PageNumber	The page to retrieve.  Default value is 0.  Example: when you set the PageNumber to 0 and the PageSize is 20, the first 20 packages are retrieved. When you set the PageNumber to 1 and the PageSize is 20, the next 20 packages are retrieved.  Note: when the value you pass is higher than the actual number of pages, an empty list will be returned.	

## Response parameters

<b><u>PARAMETER</u></b>	<b><u>CONTENT / DESCRIPTION</u></b>	<b><u>TYPE</u></b>
Packageld	Unique identifier of the package.	String (GUID)
PackageName	Name of the package.	String
CorrelationId	Identifier that correlates this package to other resources in the proofs system.	
PackageCreationDate	Date and time when the package was created according to the server.  Format is ISO 8601 date-time. E.g. 2020-01-23T12:34:00.000Z	String (date-time)
PackageDeletionDate	Date and time when the package was deleted according to the server.  Format is ISO 8601 date-time. E.g. 2020-01-23T12:34:00.000Z	String (date-time)
DelayedDeletionDate	Date and time when the audit proofs of the package were deleted according to the server.  Format is ISO 8601 date-time. E.g. 2020-01-23T12:34:00.000Z	String (date-time)
Documents	The documents that the retrieved package contains.	Array of objects
<b><u>PARAMETER</u></b>	<b><u>CONTENT / DESCRIPTION</u></b>	<b><u>TYPE</u></b>
Name	Name of the document.  Minimum length: 1  Maximum length: 150	String (GUID)
DocumentId	Unique identifier of the document.	String (GUID)
CorrelationId	Identifier that correlates this document to other resources in the proofs system.	String (GUID)

## Example response

```
{
  "PageSize": 20,
  "Total": 1,
  "Items": [
    {
      "PackageId": "00000000-0000-0000-0000-000000000000",
      "PackageName": "example package",
      "CorrelationId": "string",
      "PackageCreationDate": "2020-09-04T07:21:28.991Z",
      "PackageDeletionDate": "2020-09-04T07:21:28.991Z",
      "DelayedDeletionDate": "2020-09-04T07:21:28.991Z",
      "Documents": [
        {
          "Name": "string",
          "DocumentId": "00000000-0000-0000-0000-000000000000",
          "CorrelationId": "string"
        }
      ]
    }
  ]
}
```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All audit proof info is successfully retrieved.
<b>400 Bad request</b>	The parameter <b>Status</b> does not support value 'xyz'. Supported value is 'Deleted' only.
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect

## 12.6 Delete Audit Proof Info

### Description

Audit proofs are no longer automatically deleted as soon as the corresponding package is deleted in the Document Portal or via the API.

The time (in days) after which they are deleted can now be configured on environment level in the Configuration Index, or specified per package through the **DelayedDeletionDate** parameter in the Delete Package by ID call.

If you want to delete audit proofs earlier than specified, you can also do a manual Delete audit proof info call and specify of which package the audit proof info must be deleted.

### URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/{packageld}/auditproof](https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/auditproof)

### HTTP Method

DELETE

### Template parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
Packageld*	Unique identifier of the package whose audit proofs must be deleted.	String (GUID)

### Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	All audit proofs are deleted successfully
<b>401 Unauthorized</b>	Caller is unauthorized to perform this operation.
<b>404 Not found</b>	No audit proofs available for this package
<b>409 Conflict</b>	Could not perform operation on package with id {id}

### Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect
<b>409</b>	Package.InvalidStatus
<b>404</b>	Package.NotFound

## 12.7 Audit Proofs XML Format

The XSD below describes the structure of the XML that will be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="qualified">
  <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:attribute type="xs:string"
name="ipAddress" use="optional" />
                                        </xs:complexType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            <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:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute type="xs:string" name="documentCorrelationId" use="optional" />
    <xs:attribute type="xs:string" name="documentId" use="optional" />
  </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:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</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>
```



## 13. Audit trails

Prior eSignatures versions already supported the Audit Proof, an XML that can be downloaded through the API that keeps track of all actions taken on a package.

With eSignatures 6.1.0, we added the notion of the Audit Trail. The Audit Trail is a pdf that can be downloaded from the Document Portal or through the API. The Audit Trail contains the actions taken for that package and can be downloaded when the package is in a final state.

# Get Audit Trail

## Description

This call allows you to retrieve the Audit Trail of the specified package in a specified language.

## URL

https://[servername]:[port]/esig/webportalapi/v4/packages/{packageld}/audittrail/{culture}

## HTTP Method

GET

## MIME Type

application/pdf

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Packageld*	The unique identifier of the package whose Audit Trail you want to retrieve.	String (GUID)
Culture*	ISO 639-1 language code of the language.	String

## Response parameters

The API returns the Audit Trail in pdf format.

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The Audit Trail pdf is returned.
404 Not Found	The package could not be found.
409 Conflict	The package is not in a final state.

# Verify Audit Trail

## Description

This call allows you to verify if a specified package's Audit Trail is valid.

## URL

[https://\[servername\]:\[port\]/esig/webportalapi/v4/packages/packages/{packageId}/audittrail/verify](https://[servername]:[port]/esig/webportalapi/v4/packages/packages/{packageId}/audittrail/verify)

## HTTP Method

GET

## Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
PackageId*	The unique identifier of the package whose Audit Trail you want to verify.	String (GUID)

## Example response

true
------

## Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	A Boolean indicating if the Audit Trail is valid.
404 Not Found	The package could not be found.

## 14. SigningMethods

The SigningMethods collection currently contains one single call, to get the configured SigningMethods.

# Get signing methods

## Description

This call retrieves all signing methods that are currently enabled in the eSignatures Configuration Index and are thus available to the users.

This call is useful for integrators who need to know which methods are currently enabled in a given eSignatures installation, otherwise that list of signing methods would need to be duplicated in the integrator's configuration database as well.

Note that this API v4 call retrieves both the **SigningTypes** that are configured in the deprecated **Signing Options** settings group, and the **SigningMethods** and the **Signing Behavior** they belong to, configured in the new **Signing Options** settings group. The new SigningMethods are formatted as follows: SigningBehavior:SigningMethod. E.g. SmartCard:BeID.

## URL

https://[servername]:[port]/esig/webportalapi/v4/signingmethods

## HTTP Method

GET

## MIME Type

application/json

## Request parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
isActive	Determines whether the call must retrieve all active or inactive SigningMethods.	Boolean

## Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ConfiguredSigningMethod	List of configured SigningMethods	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
isActive	Whether the SigningMethod can be used to create new signing fields.	Boolean
Name	The name of the SigningMethod as configured in the Configuration Index.	String
DisplayNames	The names of the SigningMethod in the different languages as they will be shown to a signer.	String
RequiredProperties	The properties that are required to be passed in the Create stakeholders call. When RequiredProperties are returned for a SigningMethod, it means mandated signing rules have been applied to that SigningMethod. To check whether a signer is mandated to signing during a particular signing session, eSignatures compares the data passed in the RequiredProperties to the external data retrieved from the signing certificate or returned by the signing service.	Array or objects

## Example response

```
[
  {
    "IsActive": true,
    "Name": "BeId",
    "DisplayNames": {
```

```

        "de": "BeID",
        "en": "BeID",
        "es": "BeID",
        "fr": "BeID",
        "nl": "BeID"
    },
    "RequiredProperties": [
        "BirthDate",
        "LastName"
    ]
},
{
    "IsActive": true,
    "Name": "BeLawyer",
    "DisplayNames": {
        "de": "BeLawyer",
        "en": "BeLawyer",
        "es": "BeLawyer",
        "fr": "BeLawyer",
        "nl": "BeLawyer"
    },
    "RequiredProperties": [
        "BeLawyer"
    ]
},
{
    "IsActive": true,
    "Name": "Manual",
    "DisplayNames": {
        "de": "Manuell",
        "en": "Manual",
        "es": "Manual",
        "fr": "Manuscrite",
        "nl": "Handmatig"
    },
    "RequiredProperties": []
},
{
    "IsActive": true,
    "Name": "Manual:Manual",
    "DisplayNames": {
        "de": "Manuell",
        "en": "Manual",
        "es": "Manual",
        "fr": "Manuscrite",
        "nl": "Handmatig"
    },
    "RequiredProperties": []
},
{
    "IsActive": true,
    "Name": "ManualBeId",
    "DisplayNames": {
        "de": "BeIDManual",
        "en": "BeIDManual",
        "es": "BeIDManual",
        "fr": "BeIDManual",
        "nl": "BeIDManual"
    },
    "RequiredProperties": [
        "BeId"
    ]
},
{
    "IsActive": true,
    "Name": "OpenIdConnect:Google".

```

```

    name : "openidconnect-google",
    "DisplayNames": {
      "de": "Google",
      "en": "Google",
      "es": "Google",
      "fr": "Google",
      "nl": "Google"
    },
    "RequiredProperties": []
  },
  {
    "IsActive": true,
    "Name": "SmartCard:BeID",
    "DisplayNames": {
      "de": "BeID",
      "en": "BeID",
      "es": "BeID",
      "fr": "BeID",
      "nl": "BeID"
    },
    "RequiredProperties": [
      "BirthDate",
      "LastName"
    ]
  },
  {
    "IsActive": true,
    "Name": "SmartCard:BeIDManual",
    "DisplayNames": {
      "de": "BeIDManual",
      "en": "BeIDManual",
      "es": "BeIDManual",
      "fr": "BeIDManual",
      "nl": "BeIDManual"
    },
    "RequiredProperties": [
      "BeId"
    ]
  },
  {
    "IsActive": true,
    "Name": "SmartCard:BeIDMatchID",
    "DisplayNames": {
      "de": "BeID2",
      "en": "BeID2",
      "es": "BeID2",
      "fr": "BeID2",
      "nl": "BeID2"
    },
    "RequiredProperties": []
  },
  {
    "IsActive": true,
    "Name": "SmartCard:BeLawyer",
    "DisplayNames": {
      "de": "BeLawyer",
      "en": "BeLawyer",
      "es": "BeLawyer",
      "fr": "BeLawyer",
      "nl": "BeLawyer"
    },
    "RequiredProperties": [
      "BeLawyer"
    ]
  }
]

```

Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The SigningMethods list is successfully returned.
<b>401 Unauthorized</b>	Caller is unauthorized to do this operation.

Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>401</b>	Authentication failed, username or password incorrect



## 15. PDF Signing Locations

SigningFields can be added to a PDF in 3 different manners:

- [Via coordinates, specifying a specific location and specific dimensions](#)
- [Via Text Markers](#)
- [Via FieldId reference](#)

**Note:** all fields should at least be 112 points wide and 70 points 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 points x 120 points. Using the absolute minimum values may lead to round-off errors during conversions.

## 16. Error code descriptions

The following list describes all error codes in more detail. The codes may contain one or more placeholders which will be discussed in each section.

### 16.1 Actor

#### **Actor.NotFound**

The actor with the provided actor id could not be found in the database.

#### **Actor.TypeInvalid**

The actor type can only contain one of four possible values: "FormFiller", Approver", "Signer" or "Receiver".

#### **Actor.InvalidOrder**

Since approvers must first approve a package before it is sent to any signers, all approvers must be added before all signers in the signing flow.

Since receivers only receive a package after it has been signed by all signers, all receivers must be added to the last step of the signing flow.

If you do not define the process steps manually, all **FormFiller** actors are also added to the first step, together with the Approvers. As a result, the approval action may happen before, during or after the filling of forms.

### 16.2 ContactGroup

#### **ContactGroup.NotFound**

The contactGroup with the provided contactGroup code could not be found.

The contactGroup code is generated automatically when you create a contact group in the eSignatures WebPortal. When the contactGroup code cannot be found, this means the contact group no longer exists in the WebPortal, or never existed.

#### **ContactGroup.CountryCodeDisabled**

None of the contact group members have a valid phone number country code that can be used with {signingMethod} signing method.

#### **ContactGroup.MemberCountryCodeDisabled:{phoneNumber}**

The country prefix of phone number [{phoneNumber}] is not enabled in the {signingMethod} settings.

### 16.3 Document

#### **Document.DataInvalid**

The document data is not in base64 format.

#### **Document.CorrelationIdAlreadyExists**

Since the provided CorrelationId already exists and it must be unique, the document cannot be uploaded.

The CorrelationId is the identifier to correlate the package to other resources in the proofs system.

#### **Document.Xml.NotEnabled**

XML signing has not been enabled in your eSignatures environment. XML documents can therefore not be uploaded.

### **Document.Xml.UploadNotWellFormed**

The provided .xml document is not well-formed and contains some irregularities, and can therefore not be uploaded.

### **Document.Xml.AlreadySignedNotAllowed**

The provided .xml document already contains a digital signature and can therefore not be signed.

### **Document.DataMissing**

Base64 string data is missing.

### **Document.InvalidTargetFileType**

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**

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.NotFoundInPackage**

The document with the provided document id could not be found in the package.

### **Document.NotFound**

The document with the provided document id could not be found.

### **Document.NameLengthIncorrect**

The name of the given document was longer than 150 characters.

### **Document.UnsupportedLanguage**

The given document language is not supported.

### **Document.InvalidTargetFileType**

The provided targetType is not supported.

The supported values are **application/pdf** and **application/xml**.

## **16.4 DocumentGroup**

### **DocumentGroup.NotFoundWithCode**

The documentGroup with the provided code could not be found.

When creating a documentGroup in the eSignatures WebPortal, a corresponding documentGroup code is generated automatically. When a documentGroup code cannot be found, this means the documentGroup no longer exists, or never existed.

## **16.5 FormFields**

### **FormField.InvalidPage**

One of the form 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.

#### **FormField.MarkerOrFieldIdNotFound**

The marker or field id for the given form field was not found or did not confirm to requirements. Consequently, the form field was not created.

#### **FormField.NotFound**

The field id for the given form field was not found.

#### **FormField.ShouldbeEmpty**

The given form field is not empty.

#### **FormField.NameNotUnique**

Each form field within a document must have a unique name.

#### **FormField.InvalidWidthMarker**

The given marker id contains a width which is smaller than 10 points.

#### **FormField.InvalidHeightMarker**

The given marker id contains a height which is smaller than 10 points.

#### **FormField.InvalidWidthCoordinate**

The given marker id contains a width which is smaller than 10 points.

#### **FormField.InvalidHeightCoordinate**

The given marker id contains a height which is smaller than 10 points.

## **16.6 GroupMember**

#### **GroupMember.RequiredFieldsMissing**

One of the required fields for this member of a group stakeholder is missing.

The placeholder includes the name of the missing field. E.g. FirstName, EmailAddress.

#### **GroupMember.UnsupportedLanguage**

The given language for this member of a group stakeholder is not supported.

#### **GroupMember.EmailAddressInvalid**

The provided email address for this member of a stakeholder group is invalid.

#### **GroupMember.BirthDayInvalid**

Parsing the BirthDate from string to date format was unsuccessful.

#### **GroupMember.BirthDayInFuture**

The provided date for this member of a group stakeholder cannot be a valid birthday because it is in the future.

### **GroupMember.InvalidPhoneNumber**

The provided phone number for this member of a group stakeholder is not a valid mobile phone number.

## **16.7 LegalNotice**

### **LegalNotice.NameAndTextUsedSimultaneously**

When defining a defaultLegalNotice, you can choose from two defaultLegalNotice types: **name** or **text**.

Both types cannot be used at the same time.

## **16.8 Package**

### **Package.PackageAcceptsDocumentType**

A package accepts only one document type: PDF or XML. It's not supported to combine PDF and XML within a package.

### **Package.NotFound**

The package with the provided package id could not be found.

### **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** The operation on the specified package could not be performed because the package has an invalid status.

## **16.9 Pdf**

### **Pdf.UploadDoesNotComplyToSpec**

Uploaded or converted document doesn't comply to the pdf specification.

## **16.10 PdfErrorHandling**

### **PdfErrorHandling.InvalidType**

Invalid value for pdf error handling method.

## **16.11 PersonGroup**

### **PersonGroup.CountryCodeDisabled**

None of the person group members have a valid phone number country code that can be used with {signingMethod} signing method.

### **PersonGroup.MemberCountryCodeDisabled:{phoneNumber}**

The country prefix of phone number [{phoneNumber}] is not enabled in the {signingMethod} settings.

### **PersonGroup.MissingPhoneNumbers**

None of the person group persons have a phone number provided to use with signing method [Smsotp]

### **PersonGroup.SomeMissingPhoneNumbers**

Some of the group members don't have a phone number required for [SmsOTP] signing method.

### **PersonGroup.MissingProperties**

None of the group members have properties required to use with signing method [{signingMethod}].

### **PersonGroup.MemberMissingProperties**

Group member [{emailAddress}] has missing properties [{String.Join(",", properties)}] required to use with signing method [{signingMethod}].

## **16.12 ProcessStep**

### **ProcessStep.MixedTypes**

A process step may only contain one type of actor: formfiller, approver, signer or receiver. It is not supported to mix actor types within a process step.

## **16.13 Request**

### **Request.RequiredFieldsMissing**

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.

### **Request.OneOfFieldsIsMissing**

The request could not be completed because none of the expected fields is passed.

### **Request.FieldMaxLength**

The request could not be completed because the field value exceeds the maximum length.

### **Request.FieldMinimumValue**

The request could not be completed because the field value does not contain the minimum number of characters.

### **Request.UnsupportedValue**

The request could not be completed because the field value contains characters that are not supported.

## **16.14 SigningField**

### **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.

### **SigningField.MarkerOrFieldIdNotFound**

The marker or field id for the given signature agent was not found or did not confirm to requirements. Consequently, the signing field was not created.

### **SigningField.NotFound**

The field id for the given signature agent was not found.

### **SigningField.MarkerAlreadySigned**

The given marker id already contains a signature, and could therefore not be signed.

#### **SigningField.InvalidWidthMarker**

The given marker id contains a width which is smaller than 112 points.

#### **SigningField.InvalidHeightMarker**

The given marker id contains a height which is smaller than 70 points.

## 16.15 SigningMethod

#### **SigningMethod.Invalid**

The passed SigningMethod parameter is not a valid signing method.

## 16.16 Stakeholder

#### **Stakeholder.UnsupportedLanguage**

The provided language is not supported. The message might contain the currently supported values.

#### **Stakeholder.BirthdayInFuture**

The provided date cannot be a valid birthday because it is in the future.

#### **Stakeholder.BirthdayInvalid**

Parsing the BirthDate from string to date format was unsuccessful.

#### **Stakeholder.EmailAddressInvalid**

The provided email address is invalid.

#### **Stakeholder.InvalidPhonenumber**

The provided phone number is not a valid mobile phone number.

#### **Stakeholder.CountryCodeDisabled::{phoneNumber}**

The country prefix of phonenummer [{phoneNumber}] is not enabled in the {signingMethod} settings.

#### **Stakeholder.TypeInvalid**

The provided stakeholder type is not supported. The supported values are: **Person**, **Group** and **ContactGroup**.

#### **Stakeholder.NotFound**

The provided stakeholder could not be found.

#### **Stakeholder.NotFoundinPackage**

The provided stakeholder could not be found within the specified package.

## 16.17 Theme

#### **Theme.NotFoundWithCode**

When you create a theme in the Configuration Index, a theme code is generated automatically.

When a theme with the provided theme code cannot be found, this means the theme no longer exists, or never existed.

## 16.18 User

### **User.NotFound**

The user with the provided email address could not be found.

## 16.19 URL

### **Url.Invalid**

The provided url does not comply to the specifications.



# API v3 Technical Documentation

This section documents the technical specifications of the Connective eSignatures API version 3.

**Important:** with the introduction of API version 4, API v3 has become deprecated. This means no new functionality will be added and the v3 routes will be removed once v5 will be released.

Although API v3 routes can still be used in combination with v4 routes, it is strongly recommended to use API v4 as starting point of any new integration or update.

## Revisions

<u>DATE</u>	<u>OWNER</u>	<u>TOPIC</u>
2020-01-13	DGI	Document creation
2020-01-20	DGI	Updated error codes
2020-01-27	DGI	Corrections
2020-01-29	DGI	Update to 5.5.1 + Error code description correction
2020-02-07	DGI	Correction in callback details
2020-02-17	DGI	Update to version 5.5.2
2020-03-06	CJ	Update to version 5.5.3
2020-04-22	DGI	Added note on package/document name
2020-05-05	DGI	Update to version 5.5.4
2020-06-17	DGI	Added Get Enabled Languages call + correction about itsme signing policy.
2020-08-27	DGI	Correction in base URL

# 1. Introduction

This document describes the technical specifications of Connective eSignatures 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.

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]/esig/webportalapi/v3/`**

### **Important:**

When upgrading to eSignatures 6.3.x, you need to add **/esig** to your API endpoint for the API calls, as shown in the default URL above.

### **Important note about MTLS:**

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 must be changed through configuration at installation time should have been communicated to you by email or sms.

**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 by default limited to one-time use (unless the administrator disabled the **IsOneTimeUrlEnabled** setting in the Config Index, which is not recommended). 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.

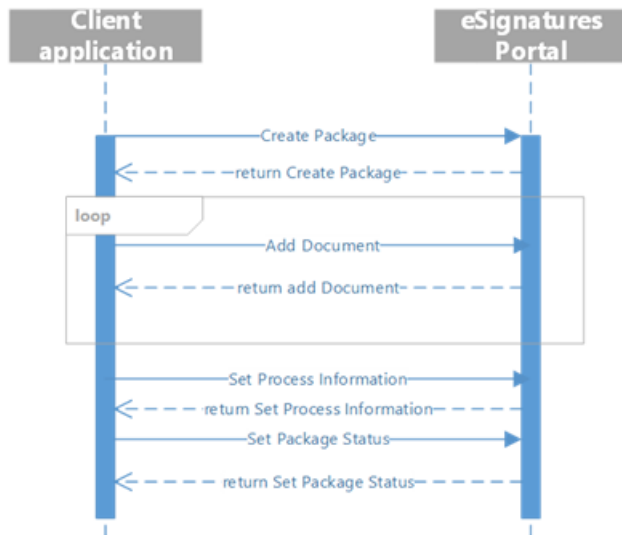
When end users open a download URL they already opened, they will see a warning. On the same screen they can ask for an email to be sent to the original signer or receiver email address (if available) which will then contain a new URL.

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

Since 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:

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Errors (array of objects)</b>	List of the errors	Array
<b>ErrorCode</b>	Error code with variable information	String
<b>Message</b>	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 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://\[servername\]:\[port\]/esig/webportalapi/v3/packages](https://[servername]:[port]/esig/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	<p>Package name, seen in the eSignatures Portal and when downloading as zip file.</p> <p><b>Note:</b> do not add an extension to the PackageName value.</p> <p><b>Important:</b> Pay attention when choosing a package name.</p> <p>A package file name must contain between 1 and 150 characters. Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (*), colon (:), pipe ( ), quote ("), double quote ("), less than (&lt;), greater than (&gt;). Note however, that is list is not exhaustive.</p> <p>Don't use characters that are HTML-sensitive such as ampersand (&amp;) or apostrophe (').</p> <p><i>Note*</i>: when using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.</p>	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

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>CorrelationId</b>	Optional	<p>Id that indicates which packages are correlated.</p> <p>The CorrelationId can be used in a GET call to retrieve the Audit proof file (a signed .xml file) about all correlated packages.</p> <p>See section <a href="#">7. Audit proofs</a> for more information about the audit proofs and the corresponding calls.</p> <p><b>Important:</b> 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.</p>	String
<b>DocumentGroupCode</b>	Optional	The 'Code' which identifies a document group in which the package should be shown. Default is "00001" to signify "My Documents". See section <a href="#">6.1: Get DocumentGroups</a> .	String
<b>ThemeCode</b>	Optional	Theme that must be applied to the package.	String
<b>DownloadUnsignedFiles</b>	Optional	<p>This parameter determines whether packages can be downloaded from the WYSIWYS before approving/signing.</p> <p>Enter 'true' if you want actors to be able to download the package before approving/signing. This way they can print it and read it on paper for instance.</p> <p>Enter 'false' to hide the download icon and prevent actors to be able to download packages from the WYSIWYS.</p> <p>When no value is entered, this parameter takes its value from the Config Index setting <b>IsDownloadUnsignedFilesEnabled</b> under <b>Customization Settings</b>.</p>	Boolean
<b>ReassignEnabled</b>	Optional	<p>This parameter determines whether packages can be reassigned from the WYSIWYS to another approver/signer.</p> <p>Enter 'true' if you want actors to be able to reassign the package.</p> <p>Enter 'false' to hide the reassign button and prevent actors to be able to reassign packages from the WYSIWYS.</p> <p>When no value is entered, this parameter takes its value from the Config Index setting <b>IsReassignEnabled</b> under <b>Customization Settings</b>.</p>	Boolean
<b>ActionUrlExpirationPeriodInDays</b>	Optional	<p>This parameter determines after how many days the action URLs must expire when they are not used.</p> <p>When no value is entered, this parameter takes its value from the Config Index setting <b>IsActionUrlExpirationEnabled</b> under <b>Customization Settings</b>.</p>	Integer
<b>ExpiryTimestamp</b>	Optional	<p>The date and time when this package expires and can no longer be approved/signed. Documents in packages all use the value given here.</p> <p>Format is ISO 8601 date-time.</p> <p>E.g. 2018-01-23T12:34:00.000Z</p>	String with Date+Time +Offset
<b>ExternalPackage Reference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ExternalPackageData</b>	Optional	This field is reserved for future use. It can be used for customer-specific extensions to integrate with third-party services, such as Debit Card signing. It is not part of a standard eSignatures installation and should not be used in calls.	String
<b>F2FRedirectUrl</b>	Optional	<p>URL to which the end user is redirected after all fields have been signed with 'face to face' signing, or when all fields have been rejected. The redirect occurs immediately after signing or rejected. This field must be a valid absolute url.</p> <p><b>Attention:</b> 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 <b>Set Process Information</b> call. See section <a href="#">5.4 Set Process Information</a> &gt; <b>5.4.14: Redirect URL Details for more info.</b></p> <p><b>Note:</b> 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
<b>NotificationCallbackUrl</b>	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

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageId</b>	Unique identifier of the package.	String
<b>CreationTimestamp</b>	<p>Date and time when the package was created according to the server.</p> <p>Format is ISO 8601 date-time.</p> <p>E.g. 2018-01-23T12:34:00.000Z</p>	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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The package was created successfully. The URL for the new package is available in the Location header.

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>400 Bad request</b>	The package could not be created due to invalid parameters in the request.
<b>409 Conflict</b>	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

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldIsMissing
<b>400</b>	Package.ExpiryTimestampInvalid
<b>400</b>	Package.ExpiryTimestampInPast
<b>400</b>	Url.Invalid
<b>409</b>	DocumentGroup.NotFoundWithCode
<b>409</b>	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.

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 text/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.

**Note:** in case a Callback error should occur, eSignatures now by default retries to do the callback 3 times, during 3 retry cycles. System administrators may customize this retry mechanism in the Worker config file. See **Connective - eSignatures 5.5.0 - Installation Documentation - Limited Public – OP** for more information.

#### 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:

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>packageId</b>	Unique identifier of the package	String
<b>actorId</b>	Identifier of the actor for which the notification was triggered	String
<b>language</b>	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
<b>notificationType</b>	Which kind of notification was requested	Integer
<b>notificationTypeKey</b>	Which kind of notification was requested	String

A callback may currently happen in the following cases:

(Note that the number in the left column is the id.)

<u>NOTIFICATIONTYPEKEY</u>	<u>NOTIFICATIONTYPE</u>	<u>USE CASE</u>
<b>SendSignerUrl</b>	<b>0</b>	The single-use signing link has been used already and the end user requested a new link.

<u>NOTIFICATIONTYPEKEY</u>	<u>NOTIFICATIONTYPE</u>	<u>USE CASE</u>
<b>SendDownloadUrl</b>	<b>1</b>	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.

**Note:** in case a Callback error should occur, eSignatures now by default retries to do the callback 3 times, during 3 retry cycles. System administrators may customize this retry mechanism in the Worker config file. See **Connective - eSignatures 5.5.0 - Installation Documentation - Limited Public – OP** for more information.

## 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]/esig/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:

<u>REQUEST ENTITY</u>	<u>CONTENT TYPE</u>	<u>DESCRIPTION</u>
<b>Data</b>	<b>application/json;charset=utf8</b>	Json request
<b>Document</b>	<b>application/octet-stream</b>	Document which needs to be signed
<b>Representation</b>	<b>application/octet-stream</b>	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

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package	String

## 5.2.7 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Document</b>	Conditional	Attached document in base64 encoded format. <i>Required</i> unless Multipart format is used.	String
<b>DocumentLanguage</b>	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
<b>DocumentName</b>	Required	Name of the document to be shown in the eSignatures Portal.  <b>Note:</b> do not add an extension to the <b>DocumentName</b> value.  <b>Important:</b> Pay attention when choosing a document name. A document file name must contain between 1 and 150 characters. Don't use forbidden file name characters such as slash (/), backslash (\), question mark (?), percent (%), asterisk (*), colon (:), pipe ( ), quote ("), double quote ("), less than (<), greater than (>). <i>Note however, that is list is not exhaustive.</i> <i>Don't use characters that are HTML-sensitive such as ampersand (&amp;) or apostrophe (').</i> <i>Note*:</i> when using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.	String



<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningFields</b>	Required	See section 5.2.7.1 below.	Array of objects
<b>CorrelationId</b>	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><b>Important:</b> the CorrelationId value must be unique within the same Package.</p> <p><b>Important:</b> The Audit proofs setting must be enabled in the eSignatures Configuration Index to use this parameter.</p>	String
<b>DocumentType</b>	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
<b>ExternalDocumentReference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String
<b>PdfErrorHandling</b>	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
<b>Representation</b>	Optional Forbidden for PDF signing	Attached representation document in base64 format. This must be PDF data.	String
<b>RepresentationType</b>	Conditional	Type of the representation document. Must be present when <b>Representation</b> is filled. Only "application/pdf" is supported.	String
<b>TargetType</b>	Optional	<p>The TargetType defines if an extra conversion to PDF/A needs to be done before signing. Values: Pdf PdfA1A PdfA2A</p> <p><b>Notes:</b> 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 <b>itsme</b> as signing method, it is <b>mandatory</b> to use <b>PdfA1A</b> or <b>PdfA2A</b> 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

**Important:** A single document must not contain more than 30 signing fields.

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 points wide and 70 points high. **Note:** it is recommended to use slightly higher values than the minimum ones. E.g. 75 points x 120 points. 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.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningFields(array of objects)</b>	Required (1-n)	One or more signing locations in the document	Array of objects
<b>PageNumber</b>	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 <a href="#">section 10.1</a> .	Integer
<b>Width</b>	Conditional	Width Minimum value is 112. It is recommended to use slightly higher values than the minimum ones.	String
<b>Height</b>	Conditional	Height Minimum value is 70. It is recommended to use slightly higher values than the minimum ones.	String
<b>Left</b>	Conditional	Position from the left of the page. Minimum value is 1. Negative values are not supported.	String
<b>Top</b>	Conditional	Position from top of the page. Minimum value is 1. Negative values are not supported.	String
<b>Label</b>	Conditional	Text string which identifies this location for later use	String
<b>MarkerOrFieldId</b>	Conditional	Unique reference to a signing field, text marker or textfield. See <a href="#">section 10.2</a> for more details.	String

#### 5.2.8 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>DocumentId</b>	Unique id for the document	String
<b>CreationTimestamp</b>	Date and time the flow was created. Format: YYYY-MM-DDThh:mm:ss+zz:zz	String
<b>Locations</b>	See table below	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Locations (array of objects)</b>	Represents a possible location for a signature	Array of objects
<b>Id</b>	Unique id for this location	String
<b>Label</b>	Detected or specified label	String
<b>PageNumber</b>	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" : "120",
      "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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The document was correctly added to the package. The response contains more information about the locations where a signer can place a signature.
<b>400 Bad Request</b>	The request contained parameters which could not be accepted.
<b>404 Not Found</b>	The package id could not be found in the database.
<b>409 Conflict</b>	When certain document conversions are forbidden, when the input document has issues, or when marker ids are not matched.

#### 5.2.12 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldsMissing
<b>400</b>	Document.InvalidTargetFileType
<b>400</b>	Document.NameInvalidLength
<b>400</b>	SigningField.MarkerAndCoordinatesCannotBeMixed
<b>400</b>	SigningField.MarkerNotUnique
<b>400</b>	SigningField.InvalidWidthCoordinate
<b>400</b>	SigningField.InvalidHeightCoordinate
<b>400</b>	SigningField.InvalidPage
<b>400</b>	Document.PasswordProtected
<b>400</b>	Pdf.UploadDoesNotComplyToSpec
<b>400</b>	PdfErrorHandling.InvalidType
<b>400</b>	Document.UnsupportedLanguage
<b>409</b>	Package.ApiVersionMismatch
<b>409</b>	Package.InvalidStatus
<b>409</b>	User.NotFound
<b>409</b>	Document.InvalidSourceFileType
<b>409</b>	Document.InvalidTargetFileType
<b>409</b>	SigningField.InvalidWidthMarker

<u>HTTP CODE</u>	<u>CODE</u>
409	SigningField.InvalidPage
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]/esig/webportalapi/v3/packages/{id}/locations`

### 5.3.3 HTTP Method

GET

### 5.3.4 MIME Type

Not applicable

### 5.3.5 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package. Use the value you receive as <b>Packageld</b> parameter in <a href="#">5.1.6</a> .	String

### 5.3.6 Response parameters

The root level has currently only one parameter: 'Documents'.

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Documents</b>	Zero or more document objects	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Documents (<u>array of objects</u>)</b>	Zero or more document objects	Array of objects
<b>DocumentId</b>	Unique id for the document	String
<b>ExternalDocumentReference</b>	External reference for identification. Make sure to use a unique value.	String
<b>Locations</b>	List of detected or created signature locations	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Documents --&gt; Locations (<u>object</u>)</b>	Possible locations for signatures.	Object
<b>Id</b>	Unique id for this location, to be used when referring to it later.	String
<b>Label</b>	Detected or specified label.	String
<b>PageNumber</b>	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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The signing locations from the package are returned successfully.
<b>404 Not Found</b>	The package Id given does not exist.
<b>409 Conflict</b>	The package with the specified id was made with an old version of the api.

### 5.3.9 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound
<b>409</b>	Package.ApiVersionMismatch

## 5.4 Set Process Information

### 5.4.1 Description

This webservice method updates the persons 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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/process](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/process)

### 5.4.3 HTTP Method

PUT

### 5.4.4 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package	String

### 5.4.5 Request parameters

The root level has currently only one parameter: 'Stakeholders'.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholder (array of objects)</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>EmailAddress</b>	Required	Email address	String
<b>FirstName</b>	Required	First name	String
<b>Language</b>	Required	UI language of this stakeholder. Currently supported: en, nl, de, fr, es.	String
<b>LastName</b>	Required	Last name	String
<b>BirthDate</b>	Conditional	Date of birth in format: YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the API or configuration might make this required. See section 5.4.12 below.	String
<b>ExternalStakeholderReference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String



<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>

#### 5.4.5.1 Actor

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders --&gt; Actors (array of objects)</b>	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Signer Receiver <b>Note:</b> actor of type Signer will become a receiver as well	String
<b>OrderIndex</b>	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 <b>OrderIndex</b> 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 <b>OrderIndex</b> value to multiple actors who may sign in parallel and assign a different <b>OrderIndex</b> value to actors who must sign before or after the parallel signing.	Int
<b>LocationIds</b>	Required for Signer	The location ids where a signature must be placed by this person.	Array of strings
<b>SigningTypes</b>	Required for Signer	One or more signing type info objects. See section 8.	Array of objects
<b>Phonenumber</b>	Optional Only Signer	Phone number to receive an SMS OTP. <b>Note:</b> 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". <b>Important:</b> never use spaces in the phone number format.	String
<b>RedirectURL</b>	Optional Only Signer Required if <b>IsBackButtonEnabled</b> is set to false in the Configuration Index.	Url to which the end user is redirected after signing, or rejecting. This field must be a valid absolute url. See section 5.4.14 Redirect Details below.	String

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>RedirectType</b>	Conditional  Only signer  Forbidden if no <b>RedirectUrl</b> has been set.	<p>This parameter defines when exactly actors are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p>Possible values: <b>AfterSession</b> (default), <b>AfterCompletion</b>, <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, actors are redirected <b>AfterSession</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For actors to be redirected immediately, enter <b>Immediately</b> as value.</p> <p>For actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value. To make sure the final actor is only redirected after the entire package is completed, enter <b>AfterCompletion</b> as value.</p> <p><b>Note:</b> when <b>AfterCompletion</b> is set for multiple actors, it will only be applied to the final actor. For the other actors, the default <b>AfterSession</b> will be used. Note that this parameter does not apply to <b>F2FRedirectUrl</b>.</p>	String
<b>SendNotifications</b>	Optional Only Signer	<p>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 '<b>true</b>' or '<b>false</b>' (the default is '<b>false</b>').</p> <p>This parameter is set per actor.</p>	Boolean
<b>UserRoles</b>	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
<b>LegalNoticeCode</b>	Optional Only Signer Forbidden for XML signing	<p>LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3</p> <p>The 3 values will point to the 3 legal notices built into the application. These can be altered in the Configuration Index.</p> <p>The language in which the legal notice is displayed, depends on the <b>DocumentLanguage</b>.</p>	String
<b>LegalNoticeText</b>	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.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders --&gt;</b> <b>Actors --&gt;</b> <b>SigningTypes (array of objects)</b>	Required (1-n)	This object specifies the signing type and its related properties.	Array of objects
<b>SigningType</b>	Required	The signing type. See <a href="#">section 9</a> .	String
<b>CommitmentTypes</b>	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
<b>MandatedSignerValidation</b>	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
<b>MandatedSignerIds</b>	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See section 5.4.12. <b>Required</b> when <b>MandatedSignerValidation</b> is of type <b>MatchId</b> .	Array of strings
<b>MatchLevel</b>	Conditional	Can only be passed when <b>NameAndBirthDate</b> is used as <b>MandatedSignerValidation</b> .  This parameter determines how accurately the actor's FirstName and LastName must match the data retrieved from the signing certificate or signing service.  <b>Important:</b> do not add the percentage sign to the value. A value between 90 and 95 usually produces good results.  See section 5.4.12 Mandated Signer Validation for more information.	Integer between 50 and 100
<b>SignaturePolicyId</b>	Optional	This parameter should only be used if you want a different signature policy than the default one set by set by Connective in the Configuration Index.  If clients want to use a custom Signature Policy, they need to log a service request at Connective. See section 5.4.13. for more information.  <b>Important:</b> if you want to combine legal notices and signature policies, the setting <b>CombineLegalNoticeAndSigningPolicy</b> must be enabled in the Configuration Index.	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",
      "ExternalStakeholderReference": "Employee",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 1,

```

```

        "LocationIds": [
            "68f35693-5530-4770-b8d8-76284719e524"
        ],
        "SigningTypes": [
            {
                "SigningType": "BeId",
                "MandatedSignerValidation": "Disabled"
            }
        ],
        "SendNotifications": true
    }
]
},
{
    "FirstName": "Jane",
    "LastName": "Jefferson",
    "EmailAddress": "jane.jefferson@mycompany.org",
    "Language": "en",
    "ExternalStakeholderReference": "CEO",
    "Actors": [
        {
            "Type": "Signer",
            "OrderIndex": 2,
            "LocationIds": [
                "76284719e524-5530-4770-b8d8-68f35693"
            ],
            "SigningTypes": [
                {
                    "SigningType": "BeId",
                    "MandatedSignerValidation": "Disabled"
                }
            ],
            "SendNotifications": true
        }
    ]
},
{
    "FirstName": "Bob",
    "LastName": "Smith",
    "EmailAddress": "bob.smith@mycompany.org",
    "Language": "en",
    "ExternalStakeholderReference": "HR Officer",
    "Actors": [
        {
            "Type": "Signer",
            "OrderIndex": 2,
            "LocationIds": [
                "55304486e524-5530-4770-b8d8-68f35748"
            ],
            "SigningTypes": [
                {
                    "SigningType": "BeId",
                    "MandatedSignerValidation": "Disabled"
                }
            ],
            "SendNotifications": true
        }
    ]
}
]
}

```

#### 5.4.8 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>None</b>	Empty object	

#### 5.4.9 Example response

```
{ }
```

#### 5.4.10 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The process information has successfully been added to the package.
<b>400 Bad Request</b>	There are invalid parameters in the request.
<b>404 Not Found</b>	When an unknown package id is passed.
<b>409 Conflict</b>	Some parameters conflict with the data found in the database or configuration.

#### 5.4.11 Error codes

<u>HHTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldIsMissing
<b>400</b>	Stakeholder.BirthDayInFuture
<b>400</b>	Stakeholder.UnsupportedLanguage
<b>400</b>	SigningField.InvalidPage
<b>400</b>	SigningType.Invalid
<b>400</b>	Stakeholder.EmailAddressInvalid
<b>400</b>	SignaturePolicy.MissingUserRole
<b>400</b>	SignaturePolicy.MissingCommitmentType
<b>400</b>	Signer.ParameterCannotBeUsedWithoutSignaturePolicy
<b>400</b>	Actor.TypeInvalid
<b>400</b>	Actor.MissingPhoneNumber
<b>409</b>	MandatedSigner.BirthDateMissing
<b>409</b>	MandatedSigner.MandatedSignerIdMissing
<b>409</b>	SignaturePolicy.NotFound
<b>409</b>	CommitmentType.NotAllowed

<u>HTTP CODE</u>	<u>CODE</u>
409	PhoneNumber.Invalid
409	Location.NotFound
409	SignaturePolicy.ShouldBeSameForActor
409	CommitmentType.ShouldBeSameForActor
409	SigningType.Disabled

#### 5.4.12 Mandated signer validation

eSignatures can do an extra identity check to verify if an end user is mandated to sign during a particular signing session. This check is called **Mandated Signer Validation**.

##### How does mandated signer validation work?

Some signing methods use a signing certificate (BeID, BeLawyer and Itsme) or signing service (iDIN) that contains data directly tied to the end user. This data may be their first and last name, but also their national security number or lawyerID.

To validate if an end user is mandated to sign during a particular session, eSignatures compares the data the API user enters in the Set Process Information call to the data stored in the signing certificate, or to the data returned by the signing service (in case of iDIN). If the data matches, the end user is mandated to sign. If it doesn't, the end user receives a message they are not mandated.

##### Types of mandated signer validation

The validation can be done in two ways: based on a unique identifier (**MatchId**) linked to a smart card or based on the end user's name and birthdate data stored in the signing certificate (**NameAndBirthDate**).

###### *MatchId*

The following signing methods support mandated signing based on **MatchId**:

- BeID
- BeLawyer

In case of BeID the **national security number** is used as unique ID. For BeLawyer, the **LawyerID** is used.

###### *NameAndBirthDate*

The following signing methods support mandated signing based on NameAndBirthDate:

- BeID
- Itsme
- iDIN

##### How to use mandated signer validation in the API?

###### *MatchId*

- Add "MatchId" as value of the MandatedSignerValidation parameter.
- Then add the required unique identifiers as value of the MandatedSignerIds parameter. Only the certificates that contain one of the allowed ids will be mandated to sign.

## Tip:

- The system admin can also choose to set MatchId as default validation type in the Configuration Index. This way, the API user doesn't need to enter the **MandatedSignerValidation** parameter, but only the **MandatedSignerIds** parameter.
- When MatchId is set as default in the Config Index, note that it can still be disabled for a single API request by passing "Disabled" as value of the **MandatedSignerValidation** parameter.

### *NameAndBirthDate*

- Add "NameAndBirthDate" as value of the **MandatedSignerValidation** parameter. **Important:** make sure the name and birth date exactly match the data on the signing certificate. How to do so is explained in **Which data must match?** below.
- The **MandatedSignerIds** parameter is not used in this case.

**Note:** the API also accepts numeric values ("0" = Disabled, "1" = NamedAndBirthDate, "2" = MatchId) but this is deprecated and will be removed in API v4 / eSignatures 6. It is recommended to use text ids.

## Which data must match?

In this section we provide an overview of the eSignatures data that is compared to the signing certificate data.

---

**Important:** since Connective does not issue the signing certificates used in the respective signing methods, we have zero control over the information they contain. In order to know which data to enter in the API request, the API user must check the data on the signing certificate to make sure the data matches.

---

For BeID signing based on MatchId:

- The id entered as **MandatedSignerIds** must match the national security number stored in the signing certificate of the eID card.

For BeLawyer signing based on MatchId:

- The id entered as **MandatedSignerIds** must match the LawyerID stored in the signing certificate of the BeLawyer card.

For BeID signing based on NameAndBirthDate:

- The **FirstName** value entered on Stakeholder level must match the "G=" value (Given Name) on the eID card.
- The **LastName** value must match the "SN=" value (SurName).
- The **BirthDate** must match the **Date of birth** value.

To check which data is stored on an eID card use the eID Viewer application.

For Itsme signing based on NameAndBirthDate:

- The **FirstName** value entered on Stakeholder level must match the First name(s) value in their Itsme app.
- The **LastName** value must match the Surname value.
- The **BirthDate** value must match the Date of Birth value.

To check which data is stored in an end user's itsme app, they must consult **My ID data** in their app.

For iDIN signing based on NameAndBirthDate:

- The **FirstName** value is not checked, since iDIN does not store this info.
- The **LastName** value must match the **consumer.legallastname** value.
- The **BirthDate** value must match the **consumer.dateofbirth** value.

**Attention:** specific LastName naming conventions apply with iDIN.



- It must be the legal last name of the iDIN consumer, without prefixes.
- It may contain a maximum number of 200 characters without numbers.
- It must not include the prefix of the first last name. Prefixes of later sections of the last name must be included.
- If the last name consists of multiple sections that are separated by a -, the – must not be led and followed by a space.
- Leading and trailing spaces are not allowed.

#### *Examples*

- For the name "Luana van Oranje-Nassau van Amsberg" the value should be "Oranje-Nassau van Amsberg".
- For the name "Jacques d'Ancona" the value should be "d'Ancona".
- For the name "Jacques d' Ancona" the value should be "Ancona"

### **Choice of signing with Mandated Signer Validation**

When you want to offer choice of signing combined with mandated signer validation, you need to make sure that all the signing methods you add on Actor level support the same Mandated Signer Validation type. For instance, if you want the end user to be able to choose between Itsme and BelD, you need to enter "NameAndBirthDate" as **MandatedSignerValidation** value, since that's the only type both signing methods support.

This restriction will be solved in a future version of eSignatures, making all kinds of combinations possible.

### **What to do in case the data doesn't match?**

For choice of signing with mandated signing to work, the data stored on the different signing certificates must be identical. Practice has shown that this is often not the case. Even when using a single signing method such as BelD, it may occur that the data stored in the signing certificate doesn't match the visual representation on the physical eID card, and therefore the wrong data is entered in the API call. This especially applies to names with special characters, or additional first names, etc.

In previous versions of eSignatures, the slightest mismatch in data would render the signer "not mandated to sign".

Since eSignatures 5.4 however, the system admin can configure a **MatchLevel** value in the Config Index when using **NameAndBirthDate** as **MandatedSignerValidation**. The **MatchLevel** parameter determines how accurately the FirstName and LastName entered in the Set Process Information or Instant Package Creation call must match the data retrieved from the signing certificate or signing service. The default value is set to 100%, which means the data must be completely identical. When you notice a signer constantly receives error messages stating they are not mandated to sign, you can lower the **MatchLevel** to for instance 90 or 95 in the Configuration Index. Or you can override the Config Index value by passing the **MatchLevel** parameter in the API call, in case you don't want to lower the accuracy on environment level.

This way you can configure the right balance so that only the required party is able to sign, while the identity check is flexible enough to go past minor inconsistencies.

**Note:** The Jaro-Winkler algorithm is used to calculate the accuracy. So, the accuracy is not based on the exact number of characters that match.

#### **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:

<u>TYPE OID</u>	<u>DESCRIPTION</u>
<b>1.2.840.113549.1.9.16.6.1</b>	Proof of origin
<b>1.2.840.113549.1.9.16.6.2</b>	Proof of receipt
<b>1.2.840.113549.1.9.16.6.3</b>	Proof of delivery
<b>1.2.840.113549.1.9.16.6.4</b>	Proof of sender
<b>1.2.840.113549.1.9.16.6.5</b>	Proof of approval
<b>1.2.840.113549.1.9.16.6.6</b>	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. The redirect occurs immediately after the user has signed or rejected.

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](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. By default, the signer has to wait until all documents are signed before being redirected. Since eSignatures 5.4, it is possible to configure when the redirect happens: 1) immediately as soon as the signing starts, 2) after a delay of 5 seconds, or 3) after completion of all documents.

## 5.5 Set Process Information (v3.1)

### 5.5.1 Description

This webservice method updates the people involved in the process, i.e. the stakeholders, and assigns them the steps that need to be taken.

A stakeholder can either be a person (default), a list of persons, or a contact group. The type of stakeholder is defined by the **Type** parameter. When you set the type to Person, or don't pass a type at all, the API call will function exactly as in previous versions.

If you want one of multiple persons to be able to approve or sign the package for the entire group, set the **Type** parameter to **PersonGroup**, pass a **PersonGroupName** and list the different persons in the **Persons** array. Each person of the group will receive a unique URL to approve/sign their document. **Attention:** as soon as one member of the group has taken action, the others no longer can.

If you don't want to list the different persons in each API call, you can also define a contact group in the eSignatures WebPortal. In that case you set the **Type** to **ContactGroup** and only need to pass the **ContactGroupCode** in the call.

#### Important notes:

- 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.
- All stakeholders must be specified each time this call is made, otherwise they will get deleted.
- 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.

### 5.5.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3.1/packages/{id}/process](https://[servername]:[port]/esig/webportalapi/v3.1/packages/{id}/process)

### 5.5.3 HTTP Method

PUT

### 5.5.4 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package	String

### 5.5.5 Request parameters

The root level has currently only one parameter: 'Stakeholders'.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects

#### 5.5.5.1 Parameters when Stakeholder Type is set to Person

The parameters below apply when the Stakeholder **Type** is set to **Person**, or not passed at all. The Set Process Information call will function exactly as in previous versions.

When the Stakeholder Type is set to another value, these parameters are forbidden.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholder (array of objects)</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>Type</b>	Optional	Stakeholder type: <b>Person</b>	String
<b>EmailAddress</b>	Required	Email address	String
<b>FirstName</b>	Required	First name	String
<b>Language</b>	Required	UI language of this stakeholder. Currently supported: en, nl, de, fr, es, da, nb, sv, fi, lv, pl.	String
<b>LastName</b>	Required	Last name	String
<b>BirthDate</b>	Conditional	Date of birth in format: YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the API or configuration might make this required. See section 5.4.12.	String
<b>Phonenumber</b>	Conditional	Phone number to receive an SMS OTP. <b>Note:</b> 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". <b>Important:</b> never use spaces in the phone number format.	String
<b>ExternalStakeholderReference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

#### 5.5.5.2 Parameters when Stakeholder Type is set to PersonGroup

The parameters below apply when the Stakeholder **Type** is set to **PersonGroup**. When the Type is set to another value, these parameters are forbidden.

When the Stakeholder Type is set to **PersonGroup** any of the persons listed in the **Persons** array is allowed to sign the package for the entire group. All persons listed in the Persons array will receive a unique URL to sign their document.

**Attention:** As soon as one person has signed the document, the others no longer can.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholder (array of objects)</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Stakeholder type: <b>PersonGroup</b>	String

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PersonGroupName</b>	Required	Name of the PersonGroup	String Max value: 128 characters
<b>Persons</b>	Required	This object provides information about all persons who are allowed to sign.	Array of objects
<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders --&gt; Persons (array of objects)</b>	Required if <b>Type</b> is set to <b>PersonGroup</b> . Otherwise forbidden.	This object provides information about all persons who are allowed to take action.	Array of objects
<b>EmailAddress</b>	Required	Email address.	String
<b>FirstName</b>	Required	First name.	String
<b>Language</b>	Required	UI language of this person. Currently supported: en, nl, de, fr, es, da, nb, sv, fi, lv, pl.	String
<b>LastName</b>	Required	Last name.	String
<b>BirthDate</b>	Conditional	Date of birth in format: YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the API or Configuration Index might make this required. See section 5.4.12.	String
<b>Phonenumber</b>	Conditional	Phone number to receive an SMS OTP. <b>Note:</b> 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". <b>Important:</b> never use spaces in the phone number format.	String
<b>ExternalReference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

#### 5.5.5.3 Parameters when Stakeholder Type is set to ContactGroup

The parameters below apply when the Stakeholder **Type** is set to **ContactGroup**. When the Type is set to another value, these parameters are forbidden.

Before you can use the **ContactGroup** type, you must first create a contact group in the eSignatures WebPortal and add the required contacts to it. Once you've created a contact group, a code is generated which must be passed as **ContactGroupCode**.

Any contacts that were added to the group will be allowed to the sign the package and will receive a unique signing URL.

**Attention:** As soon as one person has signed the document, the others no longer can.

<u>STAKEHOLDERS (ARRAY OF OBJECTS)</u>	<u>REQUIRED (1-N)</u>	<u>INFORMATION ABOUT THE PEOPLE WHO ARE INVOLVED WITH THIS PACKAGE</u>	<u>ARRAY OF OBJECTS</u>
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Stakeholder type: <b>ContactGroup</b>	String

<u>STAKEHOLDERS (ARRAY OF OBJECTS)</u>	<u>REQUIRED (1-N)</u>	<u>INFORMATION ABOUT THE PEOPLE WHO ARE INVOLVED WITH THIS PACKAGE</u>	<u>ARRAY OF OBJECTS</u>
<b>ContactGroupCode</b>	Required	Code that was generated when creating a contact group in the eSignatures WebPortal.	Integer

#### 5.5.5.4 Actor

##### Parameters when Actor Type is set to Approver

In eSignatures 5.5, a new actor type is available: Approver.

By adding an actor of the type "Approver", API users can set up an approval flow in which the package is first sent to one or more approvers, before being sent to any signers. Like signers, an approver can be of the Stakeholder Type **Person**, **PersonGroup** or **ContactGroup**. When using multiple approvers, the order in which they must approve is defined by the (mandatory) **OrderIndex** parameter. **Important:** the OrderIndex applied to approvers must be lower than the one applied to any signers.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders --&gt; Actors (array of objects)</b>	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Actor Type: <b>Approver</b>	String
<b>OrderIndex</b>	Required	<p>This number specifies in which order the actors need to execute their action. If this number is the same for all approver actors, then the order in which they approve doesn't matter.</p> <p>Incrementing numbers indicate a sequential flow: the actor with the lowest <b>OrderIndex</b> value must take action first, then the one with the second lowest value and so on.</p> <p>You can also design a complex flow: assign the same <b>OrderIndex</b> value to multiple actors who may approve in parallel and assign a different <b>OrderIndex</b> value to actors who must approve before or after the parallel approval.</p> <p><b>Important:</b> the OrderIndex applied to approvers must be lower to the one applied to signers.</p> <p><b>Note:</b> OrderIndex value is also used to sort the different approvers in the Document Details in the WebPortal. Approvers with the lowest OrderIndex value are listed first. If different approvers have the same OrderIndex value, they are listed alphabetically by last name, and then by first name.</p>	Int
<b>RedirectURL</b>	Optional Required if <b>IsBackButtonEnabled</b> is set to false in the Configuration Index.	Url to which the end user is redirected after approving, or rejecting. This field must be a valid absolute url. See section 5.4.14 Redirect Details below.	String
<b>SendNotifications</b>	Optional	eSignatures can send e-mails to all the people who are allowed to approve. Such notifications can be enabled or suppressed by setting this parameter as ' <b>true</b> ' or ' <b>false</b> ' (the default is 'false').	Boolean

##### Parameters when Actor Type is set to Signer

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders --&gt; Actors (array of objects)</b>	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Actor Type: <b>Signer</b> <b>Note:</b> actor of type Signer will become a receiver as well.	String
<b>OrderIndex</b>	Required	<p>This number specifies in which order actors need to execute their action.</p> <p>If this number is the same for all actors, then the order in which they sign doesn't matter. Incrementing numbers indicate a sequential signing flow: the actor with the lowest <b>OrderIndex</b> 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 <b>OrderIndex</b> value to multiple actors who may sign in parallel and assign a different <b>OrderIndex</b> value to actors who must sign before or after the parallel signing.</p> <p><b>Note:</b> OrderIndex value is also used to sort the different signers in the Face to face signing modal and in the Document Details in the WebPortal. Signers with the lowest OrderIndex value are listed first. If different signers have the same OrderIndex value, they are listed alphabetically by last name, and then by first name.</p>	Int
<b>Locations</b>	Required	The locations where a signature must be placed by this person.	Array of string
<b>SigningTypes</b>	Required	One or more signing type info objects. See section 9.	Array of objects
<b>Phonenumber</b>	Conditional	<p>Phone number to receive an SMS OTP.</p> <p><b>Note:</b> 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".</p> <p><b>Important:</b> never use spaces in the phone number format.</p>	String
<b>RedirectURL</b>	Optional Required if <b>IsBackButtonEnabled</b> is set to false in the Configuration Index.	<p>Url to which the end user is redirected after signing, or rejecting.</p> <p>This field must be a valid absolute url. See section 5.4.14 Redirect Details below.</p>	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>RedirectType</b>	Conditional  Forbidden if no <b>RedirectUrl</b> has been set.	<p>This parameter defines when exactly actors are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p>Possible values: <b>AfterSession</b> (default), <b>AfterCompletion</b>, <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, actors are redirected <b>AfterSession</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For actors to be redirected immediately, enter <b>Immediately</b> as value.</p> <p>For actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value. To make sure the final actor is only redirected after the entire package is completed, enter <b>AfterCompletion</b> as value.</p> <p><b>Note:</b> when <b>AfterCompletion</b> is set for multiple actors, it will only be applied to the final actor. For the other actors, the default <b>AfterSession</b> will be used. Note that this parameter does not apply to <b>F2FRedirectUrl</b>.</p>	String
<b>SendNotifications</b>	Optional	eSignatures can send e-mails to all the people who can sign. Such notifications can be enabled or suppressed by setting this parameter as ' <b>true</b> ' or ' <b>false</b> ' (the default is ' <b>false</b> ').	Boolean
<b>UserRoles</b>	Conditional 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
<b>LegalNoticeCode</b>	Optional Forbidden for XML signing	<p>LEGALNOTICE1 LEGALNOTICE2 LEGALNOTICE3</p> <p>The 3 values will point to the 3 legal notices built into the application. These can be altered in the Configuration Index.</p> <p>The language in which the legal notice is displayed depends on the <b>DocumentLanguage</b>.</p> <p><b>Note:</b> a <b>LegalNoticeCode</b> or <b>LegalNoticeText</b> set on <b>Location</b> level takes precedence over this value.</p>	String
<b>LegalNoticeText</b>	Optional Forbidden for XML signing.	<p>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.</p> <p><b>Note:</b> a <b>LegalNoticeCode</b> or <b>LegalNoticeText</b> set on Location level takes precedence over this value.</p>	String

#### Parameters when Actor Type is set to Receiver

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders --&gt; Actors (array of objects)</b>	Required (1-n)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Actor Type: <b>Receiver</b>	String
<b>SendNotifications</b>	Optional	eSignatures can send e-mails to all the people who should receive a copy of the signed package. Such notifications can be enabled or suppressed by setting this parameter as ' <b>true</b> ' or ' <b>false</b> ' (the default is ' <b>false</b> ').	Boolean



#### 5.5.5.5 Location

This object must only be used when Actor Type is set to Signer.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors =&gt; Locations (array of objects)</b>	Required (1 n)	This object specifies the locations where a signature must be placed.	Array of objects
<b>Id</b>	Required	The id of the location where a signature must be placed by this person.	String
<b>LegalNoticeCode</b>	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
<b>LegalNoticeText</b>	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

#### 5.5.5.6 Signing Type Specific Information

This object must only be used when Actor Type is set to Signer. It 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.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders --&gt; Actors --&gt; SigningTypes (array of objects)</b>	Required (1- n)	This object specifies the signing type and its related properties.	Array of objects
<b>SigningType</b>	Required	The signing type. See <a href="#">section 9</a> .	String
<b>CommitmentTypes</b>	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
<b>MandatedSignerValidation</b>	Optional	Type of validation to execute during eID or other smart card signing session. See section 5.4.12. Values: Disabled NameAndBirthDate MatchId	String

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>MandatedSignerIds</b>	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See section 5.4.12. <b>Required</b> when mandated signer validation is of type MatchId.	Array of strings
<b>MatchLevel</b>	Conditional	<p>Can only be passed when <b>NameAndBirthDate</b> is used as <b>MandatedSignerValidation</b>.</p> <p>This parameter determines how accurately the actor's FirstName and LastName must match the data retrieved from the signing certificate or signing service.</p> <p><b>Important:</b> do not add the percentage sign to the value. A value between 90 and 95 usually produces good results.</p> <p>See section 5.4.12 Mandated Signer Validation for more information.</p>	Integer between 50 and 100
<b>SignaturePolicyId</b>	Optional	<p>This parameter should only be used if you want a different signature policy than the default one set by set by Connective in the Configuration Index.</p> <p>If clients want to use a custom Signature Policy, they need to log a service request at Connective. See section 5.4.13. for more information.</p> <p><b>Important:</b> if you want to combine legal notices and signature policies, the setting <b>CombineLegalNoticeAndSigningPolicy</b> must be enabled in the Configuration Index.</p>	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"
            },
            {
              "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",
      "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"
    }
  ],
  "SendNotifications": true
}
]
},
{
  "FirstName": "Jane",
  "LastName": "Jefferson",
  "EmailAddress": "jane.jefferson@mycompany.org",
  "Language": "en",
  "ExternalStakeholderReference": "CEO",
  "Actors": [
    {
      "Type": "Signer",
      "OrderIndex": 2,
      "Locations": [
        {
          "Id": "76284719e524-5530-4770-b8d8-68f35693"
        }
      ],
      "SigningTypes": [
        {
          "SigningType": "BeId",
          "MandatedSignerValidation": "Disabled"
        }
      ],
      "SendNotifications": true
    }
  ]
},
{
  "FirstName": "Bob",
  "LastName": "Smith",
  "EmailAddress": "bob.smith@mycompany.org",
  "Language": "en",
  "ExternalStakeholderReference": "HR Officer",
  "Actors": [
    {
      "Type": "Signer",
      "OrderIndex": 2,
      "Locations": [
        {
          "Id": "55304486e524-5530-4770-b8d8-68f35748"
        }
      ],
      "SigningTypes": [
        {
          "SigningType": "BeId",
          "MandatedSignerValidation": "Disabled"
        }
      ],
      "SendNotifications": true
    }
  ]
}
]
}

```

### 5.5.8 Example request 3

In this example request, a PersonGroup containing 3 persons has been created. Any of the listed persons will be able to sign the package using either beid or manual signing.

```
{
  "Stakeholders": [
    {
      "Type": "PersonGroup",
      "PersonGroupName": "APIGroup",
      "Persons": [
        {
          "FirstName": "John",
          "LastName": "Smith",
          "EmailAddress": "John.Smith@email.com",
          "Language": "nl",
          "BirthDate": "1980-10-21",
        },
        {
          "FirstName": "Jane",
          "LastName": "Jefferson",
          "EmailAddress": "Jane.Jefferson@email.com",
          "Language": "nl",
          "BirthDate": "1985-05-05",
        },
        {
          "FirstName": "Joe",
          "LastName": "Doe",
          "EmailAddress": "Joe.Doe@email.com",
          "Language": "nl",
          "BirthDate": "1990-07-05",
        }
      ],
    },
  ],
  "Actors": [
    {
      "Type": "Signer",
      "OrderIndex": 1,
      "Locations": [
        {
          "Id": "68f35693-5530-4770-b8d8-76284719e524",
          "LegalNoticeCode": "LEGALNOTICE1"
        }
      ],
      "SigningTypes": [
        {
          "SigningType": "beid"
        },
        {
          "SigningType": "manual"
        }
      ],
      "SendNotifications": true
    }
  ]
}
```

### 5.5.9 Example request 4

This example is the same as Example request 1, with the difference that the package is first sent to approver Jane Jefferson before being sent signer John Doe.

If the package is approved, the signer will receive a signing invitation email. If the approver rejects the package, the signer won't

be notified, since there is no package to sign. The initiator who sent the package will be notified however.

```
{
  "Stakeholders": [
    {
      "FirstName": "Jane",
      "LastName": "Jefferson",
      "EmailAddress": "jane.jefferson@example.org",
      "BirthDate": "1980-09-23",
      "Language": "en",
      "Actors": [
        {
          "Type": "Approver",
          "OrderIndex": 1,
          "SendNotifications": true,
        }
      ]
    },
    {
      "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"
            },
            {
              "SigningType": "manual"
            }
          ],
          "SendNotifications": true
        }
      ]
    }
  ]
}
```

5.5.10 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
None	Empty object	

5.5.11 Example response

```
{ }
```

### 5.5.12 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The process information has successfully been added to the package.
<b>400 Bad Request</b>	There are invalid parameters in the request.
<b>404 Not Found</b>	When an unknown package id is passed.
<b>409 Conflict</b>	Some parameters conflict with the data found in the database or configuration.

### 5.5.13 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldsMissing
<b>400</b>	Request.ShouldBeEmpty:[Data field]
<b>400</b>	Stakeholder.BirthDayInFuture
<b>400</b>	Stakeholder.UnsupportedLanguage
<b>400</b>	SigningField.InvalidPage
<b>400</b>	SigningType.Invalid
<b>400</b>	Stakeholder.EmailAddressInvalid
<b>400</b>	SignaturePolicy.MissingUserRole
<b>400</b>	SignaturePolicy.MissingCommitmentType
<b>400</b>	Signer.ParameterCannotBeUsedWithoutSignaturePolicy
<b>400</b>	Actor.TypeInvalid
<b>400</b>	Actor.MissingPhoneNumber
<b>409</b>	MandatedSigner.BirthDateMissing
<b>409</b>	MandatedSigner.MandatedSignerIdMissing
<b>409</b>	SignaturePolicy.NotFound
<b>409</b>	CommitmentType.NotAllowed
<b>409</b>	PhoneNumber.Invalid
<b>409</b>	Location.NotFound
<b>409</b>	SignaturePolicy.ShouldBeSameForActor

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>409</b>	CommitmentType.ShouldBeSameForActor
<b>409</b>	SigningType.Disabled

#### 5.5.14 Mandated signer validation

These parameters haven't changed. See section 5.4.12.

#### 5.5.15 Signature Policies and Commitment Types

These parameters haven't changed. See section 5.4.13.

#### 5.5.16 Redirect URL Details

These parameters haven't changed. See section 5.4.14.



## 5.6 Set Package Status

### 5.6.1 Description

By means of the Set Package Status call, you can change the status of a package.

The available statuses are:

- Pending
- Revoked

#### Pending

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.

#### Revoked

When a package has the status "Pending" but you want to delete it, you must first change the status to "Revoked". This make the package 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 approve/sign the specified package.

### 5.6.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/status](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/status)

### 5.6.3 HTTP Method

PUT

### 5.6.4 MIME Type

application/json

### 5.6.5 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package	String

### 5.6.6 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Status</b>	Required	Pending Revoked	String

### 5.6.7 Example request

In this example we set the status of the package to "Pending", making it available for signing.

```
{
  "Status": "Pending"
}
```

### 5.6.8 Response parameters

The response is currently the same as that of the Get Package Status call. See [section 5.8](#).

### 5.6.9 Response codes

<b><u>RESPONSE STATUS CODE</u></b>	<b><u>DESCRIPTION</u></b>
<b>200 OK</b>	The package was successfully changed to the new status.
<b>400 Bad Request</b>	When invalid parameters are passed.
<b>404 Not Found</b>	When an unknown package id is passed.
<b>409 Conflict</b>	When the package doesn't contain any documents or the status doesn't allow revocation.

#### 5.6.10 Error codes

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	Request.RequiredFieldsMissing
<b>403</b>	User.PermissionDenied
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus
<b>409</b>	Package.ApiVersionMismatch
<b>409</b>	Package.ContainsNoDocuments
<b>409</b>	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 approval/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 must not contain more than 30 signing fields.
- 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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/instant](https://[servername]:[port]/esig/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.

<u>REQUEST ENTITY</u>	<u>CONTENT TYPE</u>	<u>DESCRIPTION</u>
<b>Data</b>	<b>application/json;charset=utf8</b>	Json request
<b>Document</b>	<b>application/octet-stream</b>	Document that needs to be signed
<b>Representation</b>	<b>application/octet-stream</b>	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

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Document</b>	Conditional	Attached document in base64 encoded format. <i>Required</i> unless Multipart format is used.	String
<b>DocumentLanguage</b>	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 <b>LegalNoticeCode</b> is filled.	String
<b>DocumentName</b>	Required	Name of the document and package to be shown in the eSignatures Portal. <b>Note:</b> do not add an extension to the <b>DocumentName</b> value.  <b>Important:</b> Pay attention when choosing a package name. A document file name must contain between 1 and 150 characters. Don't use forbidden file name characters such as slash (/), backslash (), question mark (?), percent (%), asterisk (), colon (:), pipe (), quote ('), double quote ("), less than (<), greater than (>). <i>Note however, that is list is not exhaustive.</i> <i>Don't use characters that are HTML-sensitive such as ampersand (&amp;) or apostrophe (').</i> <i>Note*:</i> when using itsme signing, only use characters that are supported by ISO 8859-15. This character set supports most usual characters, but some software-generated characters like curly apostrophes and long dashes are not supported.	String
<b>Initiator</b>	Required	Email address of a registered user	String
<b>Stakeholders</b>	Conditional	Information about the people who are involved in the process. 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

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>CallbackUrl</b>	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 <a href="#">5.1.11 Package Callback Details</a> .	String
<b>CorrelationId</b>	Optional	<p>Id that indicates which packages or documents are correlated.</p> <p>The CorrelationId can later be used to retrieve all related Audit proofs.</p> <p>See section 7 for more info. In this request the parameter will be used as packaged <i>and</i> document correlation id.</p> <p><b>Important:</b> the Audit proofs setting must be enabled in the Configuration Index to use this parameter.</p>	String
<b>DocumentGroupCode</b>	Optional	The 'Code' which identifies a document group in which the document should be shown. Default is "00001" to signify "My Documents". See <a href="#">section 6.1: Get DocumentGroups</a> .	String
<b>ThemeCode</b>	Optional	Theme that must be applied to the package.	String
<b>DownloadUnsignedFiles</b>	Optional	<p>This parameter determines whether packages can be downloaded from the WYSIWYS before signing.</p> <p>Enter 'true' if you want signers to be able to download the package before signing. This way they can print it and read it on paper for instance.</p> <p>Enter 'false' to hide the download icon and prevent signers to be able to download packages from the WYSIWYS.</p> <p>When no value is entered, this parameter takes it value from the Config Index setting <b>IsDownloadUnsignedFilesEnabled</b> under <b>Customization Settings</b>.</p>	Boolean
<b>ReassignEnabled</b>	Optional	<p>This parameter determines whether packages can be reassigned from the WYSIWYS to another approver/signer.</p> <p>Enter 'true' if you want actors to be able to reassign the package.</p> <p>Enter 'false' to hide the reassign button and prevent actors to be able to reassign packages from the WYSIWYS.</p> <p>When no value is entered, this parameter takes it value from the Config Index setting <b>IsReassignEnabled</b> under <b>Customization Settings</b>.</p>	Boolean
<b>ActionUrlExpirationPeriodInDays</b>	Optional	<p>This parameter determines after how many days the action URLs must expire when they are not used.</p> <p>When no value is entered, this parameter takes its value from the Config Index setting <b>IsActionUrlExpirationEnabled</b> under <b>Customization Settings</b>.</p>	Integer

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ExpiryTimestamp</b>	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
<b>ExternalDocumentReference</b>	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
<b>ExternalPackageReference</b>	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
<b>ExternalPackageData</b>	Optional	This field is reserved for future use. It can be used for customer-specific extensions to integrate with third-party services, such as Debit Card signing. It is not part of a standard eSignatures installation and should not be used in calls.	String
<b>F2FRedirectUrl</b>	Optional	Url to which the end user is redirected after all fields have been signed with 'face to face' signing, or when all fields have been rejected. The redirect occurs immediately after signing or rejecting. This field must be a valid absolute url.  <b>Attention:</b> 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 <b>Set Process Information</b> call. See section <a href="#">5.4.14: Redirect URL Details</a> for more info.  <b>Note:</b> 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
<b>NotificationCallbackUrl</b>	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 <a href="#">5.1.12 Notification Callback Details</a> .	String
<b>PdfErrorHandling</b>	Optional	How to deal with PDFs containing minor flaws. See section <a href="#">4</a> for more info. Values: Ignore DetectWarn DetectFail DetectFixWarn DetectFixFail	String

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Representation</b>	Optional  Only allowed for XML signing	Attached representation document in base64 format. This must be PDF data.	String
<b>RepresentationType</b>	Conditional	Type of the representation document, must be present when Representation is filled. Only "application/pdf" is supported.	String
<b>SigningTemplateCode</b>	Optional Forbidden for XML signing	The template configured in the portal will provide all necessary information.  The SigningTemplateCode can either be retrieved in the portal or via the Get signing templates call. See <a href="#">6.2 Getting Signing Templates (Paginated)</a> for more info.  When you use this parameter, further use of the Stakeholders parameter will be forbidden.	String
<b>TargetType</b>	Optional	The TargetType defines if an extra conversion to PDF/A needs to be done before signing. Values: Pdf PdfA1A PdfA2A <b>Notes</b> This will only work when the Document Conversion settings have been enabled in the Configuration Index. When using <b>itsme</b> as signing method, it is <b>mandatory</b> to use <b>PdfA1A</b> or <b>PdfA2A</b> as TargetType. Note however that Connective does not perform any checks whether this TargetType has been selected.	String

#### 5.7.6.1 Stakeholder information

The following parameters specify which stakeholder will sign or receive this package.

Since eSignatures 5.4.2, a stakeholder can either be a person (default), a list of persons, or a contact group. The type of stakeholder is defined by the **Type** parameter. When you set the type to Person, or don't pass a type at all, the API call will function exactly as in previous versions.

If you want one of multiple persons to be able to approve/sign the package for the entire group, set the **Type** parameter to **PersonGroup**, pass a **PersonGroupName** and list the different persons in the Persons array. Each person of the group will receive a unique URL to approve/sign their document. **Attention:** as soon as one member of the group has taken action, the others no longer can.

If you don't want to list the different persons in each API call, you can also define a contact group in the eSignatures WebPortal. In that case you set the **Type** to **ContactGroup** and only need to pass the ContactGroupCode in the call.

Parameters when Stakeholder Type is set to Person

The parameters below apply when the Stakeholder **Type** is set to **Person**, or not passed at all. The Set Process Information call will function exactly as in previous versions.

When the Stakeholder Type is set to another value, these parameters are forbidden.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders (array of objects)</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects
<b>Actors</b>	Required (1-n)	Array of actor objects. See below.	Array of objects
<b>Type</b>	Optional	Stakeholder type: <b>Person</b>	String
<b>EmailAddress</b>	Required	Email address of the signer.	String
<b>FirstName</b>	Required	First name	String
<b>Language</b>	Required	UI language of the stakeholder. Currently supported: en, nl, de, fr, es.	String
<b>LastName</b>	Required	Last name	String
<b>BirthDate</b>	Conditional	Date of birth in format YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the API or configuration might make this required. See section <a href="#">5.4.12</a> .	
<b>ExternalReference</b>	Optional	Reference given by the calling application, this parameter will not be used by the eSignatures Portal.	String

Parameters when Stakeholder Type is set to PersonGroup

The parameters below apply when the Stakeholder **Type** is set to **PersonGroup**. When the Type is set to another value, these parameters are forbidden.

When the Stakeholder Type is set to **PersonGroup** any of the persons listed in the **Persons** array is allowed to approve/sign the package for the entire group. All persons listed in the Persons array will receive a unique URL to approve/sign their document.

**Attention:** As soon as one person has taken action on the document, the others no longer can.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholder (array of objects)</b>	Required (1-n)	Information about the people who are involved in the process.	Array of objects
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Stakeholder type: <b>PersonGroup</b>	String
<b>PersonGroupName</b>	Required	Name of the PersonGroup	String Max value: 128 characters
<b>Persons</b>	Required	This object provides information about all persons who are allowed to sign.	Array of objects



PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders --&gt; Persons (array of objects)</b>	Required if <b>Type</b> is set to <b>PersonGroup</b> . Otherwise forbidden.	This object provides information about all persons who are allowed to sign.	Array of objects
<b>EmailAddress</b>	Required	Email address.	String
<b>FirstName</b>	Required	First name.	String
<b>Language</b>	Required	UI language of this person. Currently supported: en, nl, de, fr, es.	String
<b>LastName</b>	Required	Last name.	String
<b>BirthDate</b>	Conditional	Date of birth in format: YYYY-MM-DD <b>Note:</b> activating mandated signer validation in the API or Configuration Index might make this required. See section 5.4.12.	String
<b>ExternalReference</b>	Optional	Reference given by the calling application. This parameter will not be used by the eSignatures Portal.	String

Parameters when Stakeholder Type is set to ContactGroup

The parameters below apply when the Stakeholder **Type** is set to **ContactGroup**. When the Type is set to another value, these parameters are forbidden.

Before you can use the **ContactGroup** type, you must first create a contact group in the eSignatures WebPortal and add the required contacts to it. Once you've created a contact group, a code is generated which must be passed as **ContactGroupCode**.

Any contacts that were added to the group will be allowed to the sign the package and will receive a unique signing URL.

**Attention:** As soon as one person has signed the document, the others no longer can.

STAKEHOLDERS (ARRAY OF OBJECTS)	REQUIRED (1-N)	INFORMATION ABOUT THE PEOPLE WHO ARE INVOLVED WITH THIS PACKAGE	ARRAY OF OBJECTS
<b>Actors</b>	Required (1-n)	Array with more information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Stakeholder type: <b>ContactGroup</b>	String
<b>ContactGroupCode</b>	Required	Code that was generated when creating a contact group in the eSignatures WebPortal.	Integer

#### 5.7.6.2 Actor

##### Parameters when Actor Type is set to Approver

In eSignatures 5.5, a new actor type is available: Approver.

By adding an actor of the type "Approver", API users can set up an approval flow in which the package is first sent to one or more approvers, before being sent to any signers.

Like signers, an approver can be of the Stakeholder Type **Person**, **PersonGroup** or **ContactGroup**. When using multiple approvers, the order in which they must approve is defined by the (mandatory) **OrderIndex** parameter. **Important:** the OrderIndex applied to approvers must be lower than the one applied to signers.

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders =&gt; Actors (array of objects)</b>	Required (1-n-)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Actor Type: <b>Approver</b>	String
<b>OrderIndex</b>	Required	<p>This number specifies in which order the actors need to execute their action. If this number is the same for all approver actors, then the order in which they approve doesn't matter.</p> <p>Incrementing numbers indicate a sequential flow: the actor with the lowest <b>OrderIndex</b> value must take action first, then the one with the second lowest value and so on.</p> <p>You can also design a complex flow: assign the same <b>OrderIndex</b> value to multiple actors who may approve in parallel and assign a different OrderIndex value to actors who must approve before or after the parallel approval.</p> <p><b>Important:</b> the OrderIndex applied to approvers must be lower to the one applied to signers.</p> <p><b>Note:</b> OrderIndex value is also used to sort the different approvers the Document Details in the WebPortal. Approvers with the lowest OrderIndex value are listed first. If different approvers have the same OrderIndex value, they are listed alphabetically by last name, and then by first name.</p>	Boolean
<b>RedirectURL</b>	Optional Required if <b>IsBackButtonEnabled</b> is set to false in the Configuration Index.	Url to which the end user is redirected after approving, or rejecting. This field must be a valid absolute url. See section 5.4.14 Redirect Details below.	Strin
<b>SendNotifications</b>	Optional	eSignatures can send e-mails to all the people who are allowed to approve. Such notifications can be enabled or suppressed by setting this parameter as ' <b>true</b> ' or ' <b>false</b> ' (the default is 'false').	Boolean

#### Parameters when Actor Type is set to Signer

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors (array of objects)</b>	Required (1-n-)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	<p>Actor Type: Signer</p> <p><b>Note:</b> actor of type Signer will automatically become a receiver as well.</p>	String

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>OrderIndex</b>	Required	<p>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.</p> <p>Incrementing numbers indicate a sequential flow: the actor with the lowest OrderIndex value must take action first, then the one with the second lowest value and so on.</p> <p>You can also design a complex 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.</p> <p><b>Note:</b> OrderIndex value is also used to sort the different signers in the Face to face signing modal and in the Document Details in the WebPortal. Signers with the lowest OrderIndex value are listed first. If different signers have the same OrderIndex value, they are listed alphabetically by last name, and then by first name.</p>	Integer
<b>SigningFields</b>	Required	Define the locations where this actor should sign. See section 5.7.6.3 below.	Array of objects
<b>SigningTypes</b>	Required (1 -n) for Signer	One or more signing type info objects. See section 9.	Array of objects
<b>Phonenumber</b>	Conditional	<p>Phone number to receive an SMS OTP.</p> <p><b>Note:</b> 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".</p> <p><b>Important:</b> never use spaces in the phone number format.</p>	String
<b>RedirectURL</b>	Optional Required if <b>IsBackButtonEnabled</b> is set to false in the Configuration Index.	Url to which the end user is redirected after signing, or rejecting. This field must be a valid absolute url. See section 5.4.14 Redirect Details below.	String
<b>RedirectType</b>	Conditional  Forbidden if no <b>RedirectUrl</b> has been set.	<p>This parameter defines when exactly actors are redirected when using an asynchronous signing method and when a <b>RedirectUrl</b> has been defined.</p> <p>Note that this parameter does not apply to F2FRedirectUrl.</p> <p>Possible values: <b>AfterCompletion</b> (default), <b>AfterDelay</b>, <b>Immediately</b>.</p> <p>By default, actors are redirected <b>AfterCompletion</b>. This means they cannot close the signing session and have to wait for all documents to be signed before being redirected.</p> <p>For actors to be redirected immediately, enter <b>immediately</b> as value.</p> <p>For actors to be redirected after 5 seconds, enter <b>AfterDelay</b> as value.</p>	String
<b>SendNotifications</b>	Optional	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 ' <b>true</b> ' or ' <b>false</b> ' (the default is ' <b>false</b> ').	Boolean
<b>UserRoles</b>	Conditional	Role or function of the signer. <b>Note:</b> only allowed if signature policy id is present. See section 5.4.13 below.	Array of strings

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>LegalNoticeCode</b>	Optional 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.  <b>Important:</b> it is highly recommended not to combine legal notices and signature policies in API calls. This is not compliant with the ETSI standard. Should you still want to do this, to combine itsme signing with a legal notice for instance, the setting <b>CombineLegalNoticeAndSigningPolicy</b> must be enabled in the Configuration Index.	String
<b>LegalNoticeText</b>	Optional 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.  <b>Important:</b> it is highly recommended not to combine legal notices and signature policies in API calls. This is not compliant with the ETSI standard. Should you still want to do this, to combine itsme signing with a legal notice for instance, the setting <b>CombineLegalNoticeAndSigningPolicy</b> must be enabled in the Configuration Index.	String

#### Parameters when Actor Type is set to Receiver

PARAMETER	OCCURRENCE	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders =&gt; Actors (array of objects)</b>	Required (1-n-)	This object gives information about what the stakeholder must do.	Array of objects
<b>Type</b>	Required	Actor Type: <b>Receiver</b>	String
<b>SendNotifications</b>	Optional	eSignatures can send e-mails to all the people who should receive a copy of the signed package. Such notifications can be enabled or suppressed by setting this parameter as <b>'true'</b> or <b>'false'</b> (the default is 'false').	Boolean

#### 5.7.6.3 Signing Field Position

##### PDF Signing

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 points wide and 70 points high.

**Note:** it is recommended to use slightly higher values than the minimum ones. E.g. 75 points x 120 points. 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 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.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholder--&gt; Actors--&gt; SigningFields (array of objects)</b>	Required (1-n)	One or more signing locations in the document	Array of objects
<b>PageNumber</b>	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 <a href="#">10.1</a> .	Integer
<b>Width</b>	Conditional	Width Minimum value is 112. It is recommended to use slightly higher values than the minimum ones.	String
<b>Height</b>	Conditional	Height Minimum value is 70. It is recommended to use slightly higher values than the minimum ones.	String
<b>Left</b>	Conditional	Position from the left of the page. Minimum value is 1. Negative values are not supported.	String
<b>Top</b>	Conditional	Position from top of the page. Minimum value is 1. Negative values are not supported.	String
<b>MarkerOrFieldId</b>	Conditional	Unique reference to a signing field, text marker or textfield. See section <a href="#">10.2</a> 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.

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors =&gt; SigningTypes (array of objects)</b>	Required (1-n)	This object specifies the signing type and its related properties.	Array of objects

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningType</b>	Required	The signing type used in this actor's session. See <a href="#">section 9</a> .	String
<b>CommitmentTypes</b>	Conditional Forbidden for XML signing	One or more OIDs of commitment types. Can only be passed when signature policy is used. See <a href="#">section 5.4.13</a> .	Array of strings
<b>MandatedSignerValidation</b>	Optional	Type of validation to execute during eID other smart card, or itsme signing session. See <a href="#">section 5.4.12</a> . Values: Disabled NameAndBirthDate MatchId  Note that MatchId is not supported for itsme.	String
<b>MandatedSignerIds</b>	Conditional	Defines which eID or other smart cards are allowed to sign during this session. See <a href="#">section 5.4.12</a> . <b>Required</b> when mandated signer validation is of type MatchId.	Array of string
<b>MatchLevel</b>	Conditional	Can only be passed when <b>NameAndBirthDate</b> is used as <b>MandatedSignerValidation</b> .  This parameter determines how accurately the actor's FirstName and LastName must match the data retrieved from the signing certificate or signing service.  <b>Important:</b> do not add the percentage sign to the value. A value between 90 and 95 usually produces good results.  See <a href="#">section 5.4.12</a> Mandated Signer Validation for more information.	Integer between 50 and 100
<b>SignaturePolicyId</b>	Optional Forbidden for XML signing	Signing policy details for the signature. See <a href="#">section 5.4.13</a> .  <b>Important:</b> it is highly recommended not to combine legal notices and signature policies in API calls. This is not compliant with the ETSI standard. Should you still want to do this, to combine itsme signing with a legal notice for instance, the setting <b>CombineLegalNoticeAndSigningPolicy</b> must be enabled in the Configuration Index.	String

### 5.7.7 Example Request

#### 5.7.7.1 Regular Request

```

{
  "DocumentLanguage": "en",
  "DocumentName": "instant package",
  "Initiator": "john.smith@mail.com",
  "ExternalPackageReference": "reference",
  "Stakeholders": [
    {
      "FirstName": "John",
      "LastName": "Smith",
      "EmailAddress": "john.smith@mail.com",
      "BirthDate": "1974-04-12",
      "Language": "en",
      "Type": "Person",
      "ExternalStakeholderReference": "stakeref",
      "Actors": [
        {
          "Type": "Signer",
          "OrderIndex": 1,
          "SigningFields": [{// "Width": "110", "Height": "50", "PageNumber": 1, "Left": "391", "Top":
"741",
            "MarkerOrFieldId": "SIG01",
          },
        ],
        //"legalnoticetext": "custom[draft]",
        "SigningTypes":
        [
          {
            "SigningType": "manual",
            //"MandatedSignerValidation": "matchid",
            //"MandatedSignerIds": ["ee32ee8e-deed-4605-a2e5-
fbed3cbd82a", "c488d53fb75f466eaaf4f130426d407a", "94092339579"],
            //"MatchLevel": 80,
            //"CommitmentTypes": ["1.2.840.113549.1.9.16.6.5"],
            //"SignaturePolicyId": "1.3.6.1.4.1.35390.10.2.3.2.0"
          },
        ],
        "Phonenumber": "+32465232553",
        "SendNotifications": true,
      },
    ],
  "Document": ""
}

```

#### 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": "3bb6a2d1-35db-4a3a-a434-868c01bcca9d",
"PackageName": "instant package",
"Initiator": "john.smith@mail.com",
"ExpiryTimestamp": null,
"ExternalPackageReference": "reference",
"F2FSigningUrl": "https://yourFTFSignUrl",
"PackageStatus": "Pending",
"PackageDocuments": [
  {
    "DocumentId": "97cfc102-70c2-4c17-9fb7-74b29360d53f",
    "DocumentType": "application/pdf",
    "ExternalDocumentReference": null,
    "DocumentName": "instant package",
    "Locations": [
      {
        "Id": "e3ba68ac-e737-4e15-b61f-229034cf0797",
        "Label": "d28d483a-824d-43c9-93d1-026117d8ebaf",
        "PageNumber": 2
      }
    ]
  }
],
"Stakeholders": [
  {
    "Type": "Person",
    "EmailAddress": "john.smith@mail.com",
    "ContactGroupCode": null,
    "ExternalStakeholderReference": "stakeref",
    "StakeholderId": "7aa76f03-7981-44c1-8b91-598256edf260",
    "Actors": [
      {
        "Type": "Signer",
        "Reason": null,
        "CompletedBy": null,
        "CompletedTimestamp": null,
        "Locations": [
          {
            "Id": "e3ba68ac-e737-4e15-b61f-229034cf0797",
            "UsedSigningType": null
          }
        ],
        "ActorId": "db4a380f-fbd8-4bd7-b9e3-23507e986f66",
        "ActionUrl": "https://ActionUrl",
        "ActionUrls": [],
        "ActorStatus": "Available"
      }
    ],
    "PersonGroupName": null
  }
],
"CreationTimestamp": "2019-10-24T11:20:55Z",
"Warnings": []
}
```

5.7.9 Response parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
Packageld	Unique id of the package	String



<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageName</b>	Description for the Package shown to the eSignatures Portal user as file name.	String
<b>CreationTimestamp</b>	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
<b>Initiator</b>	Initiator field of the package as it was passed in at creation time.	String
<b>ExpiryTimestamp</b>	UTC formatted time at which the document expires. Can be null.	String
<b>ExternalPackageReference</b>	Returns the external reference id of the package as it was passed in at creation time.	String
<b>F2FSigningUrl</b>	Link to the package which allows to pick from all the signing session at once.	String
<b>PackageStatus</b>	Status of the package as a whole: Draft Pending Finished Rejected Revoked Expired Failed <b>Note:</b> a package has the status <b>Failed</b> when a background operation has failed and left a message on the Poison Queue.	String
<b>PackageDocuments</b>	Details for each of the documents in the package.	Array of objects
<b>Stakeholders</b>	Details for each of the people who will interact with the package.	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageDocuments (array of object)</b>	Details for each of the documents in the package. In case of an instant package there is always 1 document.	Array of objects
<b>DocumentId</b>	Unique id of the document	String
<b>ExternalDocumentReference</b>	Returns the external reference of this document as it was passed in through the Add document to package call.	String
<b>DocumentName</b>	Name of the document	String
<b>DocumentType</b>	Type of the document	String
<b>Locations</b>	See table below.	Array of objects

<u>PARAMETER</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
<b>Locations (array of objects)</b>	Represents a possible location for a signature	Array of objects
<b>Id</b>	Unique id for this location	String
<b>Label</b>	Value you entered as Request parameter	String
<b>PageNumber</b>	Number of the page on which the signature can be found	Int

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders (array of objects)</b>	Details for each of the persons which will interact with the package.	Array of objects
<b>Type</b>	Type of stakeholder: <b>Person</b> , <b>PersonGroup</b> or <b>ContactGroup</b> .	String
<b>PersonGroupName</b>	Name of the person group. Only returned if Type was set to PersonGroup in the request.	String
<b>ContactGroupCode</b>	Code of the contact group. Only returned if Type was set to ContactGroup in the request.	Integer
<b>EmailAddress</b>	Email address of the signer.	String
<b>ExternalStakeholderReference</b>	External reference identifying this person in the external system.	String
<b>StakeholderId</b>	Unique id	String
<b>Actors</b>	See below	Array

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors (object)</b>	Details of all steps to take.	Array of objects
<b>ActorId</b>	Unique id for this combination of action, stakeholder and document.	String
<b>ActionUrl</b>	<p>URL that this person can open to interact with the package.</p> <p>This parameter is only used when Stakeholder Type is set to Person, or not passed at all.</p> <p>When the Stakeholder Type is set to PersonGroup or ContactGroup, the ActionUrl will be null and the different Urls sent to the different persons are listed in the ActionsUrls array.</p> <p><b>Exception:</b> if the PersonGroup or ContactGroup only contains 1 person, the <b>ActionUrl</b> parameter will still be returned instead of the <b>ActionUrls</b>.</p>	String
<b>ActionUrls</b>	<p>Array of URLs that the different persons of the PersonGroup or ContactGroup can open to interact with the package.</p> <p>See the table below.</p>	String
<b>ActorStatus</b>	<p>Draft (package has status Draft) Inprogress (package is being signed) Available (ready for execution) Finished Rejected (signing cannot continue) Failed (signing has failed) Skipped (Initiator skipped the actor)</p>	String
<b>Type</b>	<p>Signer</p> <p>Receiver</p>	String
<b>CompletedBy</b>	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.	

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>CompletedTimestamp</b>	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	
<b>Reason</b>	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
<b>Locations</b>	See table below.	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Locations (array of objects)</b>	Represents a possible location for a signature.	Array of objects
<b>Id</b>	Unique id for this location.	String
<b>UsedSigningType</b>	Returns the signing type that was used to sign the document. See section 9 for an overview of the available signing types. If no signing type was used (i.e. if the document isn't signed yet), this parameter returns "null".	String

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors =&gt; ActionUrls (array of objects)</b>	Array of URLs that the different persons can open to interact with the package. The ActionURLs array is only used when the Stakeholder Type was set to PersonGroup or ContactGroup. Each person receives a unique URL only they can use. <b>Exception:</b> if the PersonGroup or ContactGroup only contains 1 person, the ActionUrl parameter will still be returned instead of the ActionUrls.	Array of objects
<b>EmailAddress</b>	Email address of the person.	String
<b>Url</b>	URL that this person can open to interact with the package.	String

#### 5.7.10 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>201 Created</b>	The package can be made ready for signing. The response contains more information about the locations where a signer can place a signature.
<b>400 Bad request</b>	The request contained parameters which could not be accepted.
<b>404 Not Found</b>	The package id could not be found in the database.
<b>409 Conflict</b>	When certain document conversions are forbidden, when the input document has issues, or when marker ids are not matched.

#### 5.7.11 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldsMissing

<u>HTTP CODE</u>	<u>CODE</u>
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
400	Package.ExpiryTimestampInvalid

<u>HTTP CODE</u>	<u>CODE</u>
400	Package.ExpiryTimestampInThePast
409	SigningTemplate.NotFoundWithCode
409	MandatedSigner.MandatedSignerIdMissing
409	MandatedSigner.BirthDateMissing
409	DocumentGroup.NotFoundWithCode
409	User.NotFound
409	SigningType.Disabled
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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/status](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/status)

### 5.8.3 HTTP Method

GET

### 5.8.4 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
id	Required	Unique id for the signing package	String

### 5.8.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageName</b>	Description for the Package shown to the eSignatures Portal user as file name.	String
<b>CreationTimestamp</b>	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
<b>Initiator</b>	Initiator field of the package as it was passed in at creation time.	String
<b>ExpiryTimestamp</b>	UTC formatted time at which the document expires. Can be null.	String
<b>ExternalPackageReference</b>	Returns the external reference id of the package as it was passed in at creation time.	String
<b>F2FSigningUrl</b>	Link to the package which allows to pick from all the signing session at once.	String
<b>PackageStatus</b>	Status of the package as a whole: Draft Pending Finished Rejected Revoked Expired Failed  <b>Note:</b> a package has the status <b>Failed</b> when a background operation has failed and left a message on the Poison Queue.	String
<b>PackageDocuments</b>	Details for each of the documents in the package.	Array of objects
<b>Stakeholders</b>	Details for each of the persons which will interact with the package.	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageDocuments</b> (array of <b>object</b> )	Details for each of the documents in the package	Array of objects
<b>DocumentId</b>	Unique id of the document	String
<b>ExternalDocumentReference</b>	Returns the external reference of this document as it was passed in through the Add document to package call.	String
<b>DocumentName</b>	Name of the document	String
<b>DocumentType</b>	Type of document within the package.  Possible values: application/pdf or application/xml	String

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders</b> (array of <b>objects</b> )	Details for each of the persons which will interact with the package.	Array of objects
<b>Type</b>	Type of stakeholder: <b>Person</b> , <b>PersonGroup</b> , or <b>ContactGroup</b> .	String
<b>PersonGroupName</b>	Name of the person group. Only returned if Type was set to PersonGroup in the request.	String
<b>ContactGroupCode</b>	Code of the contact group. Only returned if Type was set to ContactGroup in the request.	Integer
<b>EmailAddress</b>	Email address of the signer.	String
<b>ExternalStakeholderReference</b>	External reference identifying this person in the external system.  <b>Note:</b> when a package is reassigned, the <b>ExternalStakeholderReference</b> is transferred to the new assignee.	String
<b>StakeholderId</b>	Unique id  <b>Note:</b> when a package is reassigned, the <b>StakeholderId</b> is transferred to the new assignee in a standard use case. When using complex signing scenario in which signer 1 is also signer 3, a new <b>StakeholderId</b> will be created for the new assignee who will become signer 1.	String
<b>Actors</b>	See below	Array

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Stakeholders =&gt; Actors</b> (object)	Details of all steps to take.	Array of objects
<b>ActorId</b>	Unique id for this combination of action, stakeholder and document.	String

<u>PARAMETER</u>		<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ActionUrl</b>		<p>URL that this person can open to interact with the package.</p> <p>This parameter is only used when Stakeholder Type is set to Person, or not passed at all.</p> <p>When the Stakeholder Type is set to PersonGroup or ContactGroup, the ActionUrl will be null. Instead, the different Urls sent to the different persons are listed in the ActionsUrls array.</p> <p><b>Exception:</b> if the PersonGroup or ContactGroup only contains 1 person, the <b>ActionUrl</b> parameter will still be returned instead of the <b>ActionUrls</b>.</p>	String
<b>ActionUrls</b>		<p>Array of URLs that the different persons of the PersonGroup or ContactGroup can open to interact with the package.</p> <p>The ActionsUrls array is only used then the Stakeholder Type was set to PersonGroup or ContactGroup.</p> <p>Each person receives a unique URL only they can use. See the table below.</p>	Array of objects
<b>ActorStatus</b>		<p>Draft (package has status Draft)</p> <p>Inprogress (package is being signed)</p> <p>Available (ready for execution)</p> <p>Finished</p> <p>Rejected (signing cannot continue)</p> <p>Failed (signing has failed)</p> <p>Skipped (Initiator skipped the actor)</p>	String
<b>Type</b>		<p>Approver</p> <p>Signer</p> <p>Receiver</p>	String
<b>CompletedBy</b>		<p>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.</p> <p>Can be null, will never be present for a Receiver.</p>	
<b>CompletedTimestamp</b>		<p>Timestamp of the moment on which this action was completed.</p> <p>Format is ISO 8601 date-time.</p> <p>E.g. 2018-01-23T12:34:00.000Z</p> <p>Can be null, will never be present for a Receiver</p>	
<b>Reason</b>		<p>Returns the text entered by the person who changed the status of a package to a final state (e.g. a reject).</p> <p>Can be null, will never be present for a Receiver.</p>	String
<b>Locations</b>		See table below.	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>		<u>TYPE</u>
<b>Locations (array of objects)</b>	Represents a possible location for a signature.		Array of objects
<b>Id</b>	Unique id for this location		String



PARAMETER	CONTENT / DESCRIPTION	TYPE
<b>UsedSigningType</b>	Returns the signing type that was used to sign the document. See section 9 for an overview of the available signing types. If no signing type was used (i.e. if the document isn't signed yet), this parameter returns "null".	String
PARAMETER	CONTENT / DESCRIPTION	TYPE
<b>Stakeholders =&gt; Actors =&gt; ActionUrls (array of objects)</b>	<p>Array of URLs that the different persons can open to interact with the package.</p> <p>The ActionURLs array is only used when the Stakeholder Type was set to PersonGroup or ContactGroup.</p> <p>Each person receives a unique URL only they can use.</p> <p><b>Exception:</b> if the PersonGroup or ContactGroup only contains 1 person, the ActionUrl parameter will still be returned instead of the ActionUrls.</p>	Array of objects
<b>EmailAddress</b>	Email address of the person.	String
<b>Url</b>	URL that this person can open to interact with the package.	String

### 5.8.6 Example response

```
{
  "PackageName": "package-docu1.pdf",
  "Initiator": "signer@gmail.com",
  "ExpiryTimestamp": null,
  "ExternalPackageReference": "reference",
  "F2FSigningUrl": "http://myserver/signinit?packageSignId=6bc402eb-6cbd-423a-bf00-1157e8d68f37&f2f=True",
  "PackageStatus": "Pending",
  "PackageDocuments": [
    {
      "DocumentId": "dc2691d8-e3c0-470b-9715-e55b489ea493",
      "DocumentType": "application/pdf",
      "ExternalDocumentReference": null,
      "DocumentName": "docu1"
    }
  ],
  "Stakeholders": [
    {
      "Type": "PersonGroup",
      "EmailAddress": null,
      "ContactGroupCode": null,
      "ExternalStakeholderReference": "stakeref",
      "StakeholderId": "6b2cda0c-ab81-4984-9e16-159fe20d983f",
      "Actors": [
        {
          "Type": "Signer",
          "Reason": null,
          "CompletedBy": null,
          "CompletedTimestamp": null,
          "Locations": [
            {
              "Id": "24ab070a-29e4-496e-9b79-e66c0edcced7",
              "UsedSigningType": null
            }
          ],
          "ActorId": "2806f94d-2a45-4168-8667-cbd4ce4ce090",
          "ActionUrl": null,
          "ActionUrls": [
            {

```

```

        "EmailAddress": "john.smith@mail.com",
        "Url": "https://MyURL.com"
    },
    {
        "EmailAddress": "jane.jefferson@mail.com",
        "Url": "https://HerURL.com"
    }
],
"ActorStatus": "Available"
},
{
    "Type": "Receiver",
    "ActorId": "e3da1877-fac9-43c4-9949-bacef76718fa",
    "ActionUrl": null,
    "ActionUrls": [],
    "ActorStatus": ""
}
],
"PersonGroupName": "APIGroup"
},
{
    "Type": "Person",
    "EmailAddress": "signer@gmail.com",
    "ContactGroupCode": null,
    "ExternalStakeholderReference": "3",
    "StakeholderId": "490e242f-43c4-4bc0-a1b4-e2d67dc1fdac",
    "Actors": [
        {
            "Type": "Receiver",
            "ActorId": "0903c863-9197-4abf-93f4-694fddde9d99",
            "ActionUrl": null,
            "ActionUrls": [],
            "ActorStatus": ""
        }
    ],
    "PersonGroupName": null
}
],
"CreationTimestamp": "2019-10-23T13:34:12Z"
}

```

### 5.8.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The package was returned successfully.
<b>404 Not Found</b>	The package with the specified id could not be found.
<b>409 Conflict</b>	The package with the specified id was made with an old version of the api.

### 5.8.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound
<b>409</b>	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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/signers/{actorId}](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/signers/{actorId})

### 5.9.3 HTTP Method

PUT

### 5.9.4 MIME Type

application/json

### 5.9.5 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package.	String
<b>actorid</b>	Required	Unique id of the actor which needs updates.	String

### 5.9.6 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningTypes</b>	Required	Signing types to be used during the actor's signing session.	Array of string

### 5.9.7 Example request

```
{
  "SigningTypes": ["BeID"]
}
```

### 5.9.8 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>
<b>None</b>	Empty object

### 5.9.8 Example response

```
{ }
```

### 5.9.9 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The package was successfully changed to the new status.
<b>400 Bad Request</b>	When invalid parameters are passed in.

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>404 Not Found</b>	When an unknown package or actor id is passed in, or the actor is not a signer.
<b>409 Conflict</b>	When the signing types were never allowed in the first place or when the package status is wrong.

#### 5.9.10 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>400</b>	Request.RequiredFieldsMissing
<b>404</b>	Package.NotFound
<b>404</b>	Actor.NotFound
<b>409</b>	Package.InvalidStatus
<b>409</b>	Package.ApiVersionMismatch

## 5.10 Skip Signers

### 5.10.1 Description

A package that has been signed by at least one party, but still has some unsigned fields, that must be signed by others can now be finished by doing a Skip Signers call. This call skips all signers that haven't signed yet and sets their status to "Skipped", which results in the complete package being marked as "Finished".

This way, the package becomes available for download and it is no longer necessary to wait for all parties to sign before everyone can download the document.

If the package was created with the **SendNotifications** parameter set to true, then all signers will get a notification that they can download the ended package.

**Attention:** when you skip signers through the API, no info is added to the Audit proofs about who skipped them. This info can be added by means of the **Add Proof from External Source** call. See section 7.3 for more information.

### 5.10.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/skipsigners](https://[servername]:[port]/esig/webportalapi/v3/packages/{packageId}/skipsigners)

### 5.10.3 HTTP Method

POST

### 5.10.4 Template parameter

<u>PARAMETER</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
<b>packageId</b>	Unique Identifier of the Package	String

### 5.10.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT/DESCRIPTION</u>
<b>None</b>	Empty object

### 5.10.6 Example response

```
{}
```

### 5.10.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All signers that hadn't signed yet were successfully skipped.

### 5.10.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound:{packageId}
<b>409</b>	Package.ContainsNoDocuments:{packageId}
<b>409</b>	Package.InvalidStatus:{packageStatus}
<b>409</b>	Package.NotSigned:{packageId}

## 5.11 Skip Approvers

### 5.11.1 Description

A package that has been approved by at least one of multiple approvers can be sent to the signers by doing a Skip Approvers call. This call skips all approvers that haven't approved yet and sets their status to "Skipped", which results in the complete package being marked as "Pending".

This way, the package becomes available to the signers and it is no longer necessary to wait for all approvers to complete their task before anyone can sign.

If the package was created with the **SendNotifications** parameter set to true, then all approvers will get a notification that the packages has been sent to the signers.

**Attention:** when you skip approvers through the API, no info is added to the Audit proofs about who skipped them. This info can be added by means of the Add Proof from External Source call. See section 7.3 for more information.

### 5.11.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/skipapprovers](https://[servername]:[port]/esig/webportalapi/v3/packages/{packageId}/skipapprovers)

### 5.11.3 HTTP Method

POST

### 5.11.4 Template parameters

PARAMETER	CONTENT / DESCRIPTION	TYPE
<b>packageId</b>	Unique identifier of the Package	String

### 5.11.5 Response parameters

PARAMETER	CONTENT / DESCRIPTION
None	Empty object

### 5.11.6 Example response

```
{}
```

### 5.11.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	All approvers that hadn't approved yet were successfully skipped.

### 5.11.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound:{packageId}
<b>409</b>	Package.ContainsNoDocuments:{packageId}
<b>409</b>	Package.InvalidStatus:{packageStatus}

## 5.12 Download Package

### 5.12.1 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.

### 5.12.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/download](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/download)

### 5.12.3 HTTP Method

GET

### 5.12.4 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
id	Required	Unique id for the signing package	String

### 5.12.5 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.

### 5.12.6 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The package gets downloaded successfully.
404 Not Found	The package cannot be found.
409 Conflict	The package hasn't been fully signed.

### 5.12.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
404	Package.NotFound
404	Document.NotFoundInStore
409	Package.InvalidStatus

## 5.13 Download Document from Package

### 5.13.1 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.

### 5.13.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/download/{documentId}](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/download/{documentId})

### 5.13.3 HTTP Method

GET

### 5.13.4 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id for the signing package.	String
<b>documentId</b>	Required	Unique id of the document contained in the package.	String

### 5.13.5 Response

The eSignatures API will return a PDF document, or an XML file stream, depending on the value of the **DocumentType** parameter.

### 5.13.6 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The document gets downloaded successfully.
<b>404 Not Found</b>	The document cannot be found or is not part of the specified package.
<b>409 Conflict</b>	The package hasn't been fully signed.

### 5.13.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Document.NotFound
<b>404</b>	Package.ContainsNoDocumentWithId
<b>404</b>	Document.NotFoundInStore
<b>409</b>	Package.InvalidStatus
<b>409</b>	Document.InvalidStatus



## 5.14 Get Package Identity

### 5.14.1 Description

Retrieves the identity data of all **iDIN** actors of the package and its documents.

**Important:** There will be no identity data to retrieve until the package has the FullySigned state. This means the package must have been signed by all signers. If the package is rejected by one of the signers, the identity data of the signers who already signed is no longer available, and therefore the identity data cannot be returned.

Which identity data is retrieved from iDIN depends on the **ServiceID** value that the eSignatures administrator enters in the **Configuration Index > Signing Options Settings**. Which ServiceID value to enter and which identity data each value returns can be consulted in the Excel sheet provided by iDIN.

The default value is 21952 and returns the following data:

**consumer.bin** BIN is short for Bank identifying Number. It identifies the Consumer. BINs are bank specific and are unique for every Consumer-Merchant-Bank combination.

**bankid.deliveredserviceid** The number that is used to ask for a particular set of consumer attributes.

**consumer.housenosuf** House number suffix.

**consumer.dateofbirth** Date of birth of Consumer.

**consumer.country** Country code of country where Consumer currently resides.

**consumer.city** City of the Consumer's residential address. Used for NL addresses only.

**consumer.initials** The initials of the Consumer, defined as the first letter of each of the Consumer's first names

**consumer.street** Street name of the Consumer's residential address. Used for NL addresses only.

**consumer.postalcode** Postal code of the Consumer's residential address. Used for NL addresses only.

**consumer.legallastname** Legal last name of Consumer without prefixes.

**consumer.houseno** House number of the Consumer's residential address. Used for NL addresses only.

### 5.14.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/identity](https://[servername]:[port]/esig/webportalapi/v3/packages/{packageId}/identity)

### 5.14.3 HTTP Method

GET

### 5.14.4 MIME Type

Not applicable

### 5.14.5 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCES</u>	<u>CONTENT / DESCRIPTIONS</u>	<u>TYPES</u>
id	Required	Unique id for the signing package	String

### 5.14.6 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTIONS</u>	<u>TYPE</u>
PackageName	Description for the Package shown to the eSignatures Portal user as file name.	String
Initiator	Initiator field of the package as it was passed in at creation time.	String
Stakeholders	Details for each of the persons which will interact with the package	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPES</u>
Stakeholders (array of objects)	Details for each of the persons which will interact with the package.	Array of objects
StakeholderId	Unique id	String
Actors	See below	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTIONS</u>	<u>TYPES</u>
Stakeholders --> Actors (object)	Details of all steps to take.	Array of objects
ActorId	Unique id for this combination of action, stakeholder and document.	String
IdentityData	Array of IdentityData objects.	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
IdentityData --> SourceData	Details of the IdentityData	Array of objects
Key	Key of the IdentityData	String
Value	Value for the IdentityData Key	String

#### 5.14.7 Example response

```

"PackageName": "MyPackage",
  "Initiator": "mycompany@example.com",
  "Stakeholders": [
    {
      "StakeholderId": "xcvgb016-35ca-41ff-91d0-256b6c6785b7",
      "Actors": [
        {
          "ActorId": "25",
          "IdentityData": [
            {
              "SourceId": "0011200090339283",
              "SourceData": [
                {
                  "Key": "urn:nl:bvn:bankid:1.0:consumer.bin",
                  "Value":
"NLINGBcRA2QTTQUHVqITQGIweQmUnMhAR5eIBncFPe8meXC2SMYPxC+S2LwpNTcwhtIP/dGgqXCBSmRfVUFbUD51ja8A=="
                },
                {
                  "Key": "urn:nl:bvn:bankid:1.0:bankid.deliveredseviceid",
                  "Value": "21444"
                },
                {
                  "Key": "urn:nl:bvn:bankid:1.0:consumer.housenosuf",
                  "Value": "F6"
                }
              ]
            }
          ]
        }
      ]
    }
  ]

```

```

    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.dateofbirth",
      "Value": "19800101"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.country",
      "Value": "NL"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.city",
      "Value": "AMSTERDAM"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.initials",
      "Value": "J"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.street",
      "Value": "Main"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.postalcode",
      "Value": "1014BG"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.legallastname",
      "Value": "Smith"
    },
    {
      "Key": "urn:nl:bvn:bankid:1.0:consumer.houseno",
      "Value": "11"
    }
  ]
}

```

### 5.14.8 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The packageID 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.

### 5.14.9 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
404	Package.NotFound
409	Package.ApiVersionMismatch

## 5.15 Package Expiry Extension

### 5.15.1 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 approved or signed.

The “extend expiry” call allows to specify a new date and time in the future on which the package will expire, thus allowing it to be approved/signed 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.

### 5.15.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/expirytimestamp](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/expirytimestamp)

### 5.15.3 HTTP Method

PUT

### 5.15.4 MIME Type

application/json

### 5.15.5 Template parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Id</b>	Required	Unique id for the signing package	String

### 5.15.6 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ExpiryTimestamp</b>	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

### 5.15.7 Example request

```
{
  "ExpiryTimestamp": "{{eSigner - FutureDate}}"
}
```

### 5.15.8 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>None</b>	Empty response body	

### 5.15.9 Example response

Successful response

```
{}
```

### 5.15.10 Response codes

<b><u>RESPONSE STATUS CODE</u></b>	<b><u>DESCRIPTION</u></b>
<b>200 OK</b>	The update package expiry timestamp was set.
<b>400 Bad request</b>	The request contained parameters which could not be accepted.
<b>400 Not Found</b>	The package id could not be found in the database.
<b>409 Conflict</b>	When the status of the document does not allow to extend the expiration period.

#### 5.15.11 Error codes

<b><u>HTTP CODE</u></b>	<b><u>CODE</u></b>
<b>400</b>	Package.ExpiryTimestampInvalid
<b>400</b>	Package.ExpiryTimestampsRequired
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus

## 5.16 Send Package Reminders

### 5.16.1 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 approved/signed and send them an extra notification as a reminder.

Note that only the next available approver(s)/signer(s) in the workflow are notified. This means approvers/signers waiting for someone else to sign first in a serial workflow will not receive a notification.

### 5.16.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}/reminders](https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/reminders)

### 5.16.3 HTTP Method

POST

### 5.16.4 MIME Type

application/json

### 5.16.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>PackageId</b>	Required	Unique id of the package	String

### 5.16.6 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>None</b>	Empty response body	

### 5.16.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The reminders have been sent.
<b>404 Not Found</b>	The package with the given id could not be found.
<b>409 Conflict</b>	The package did not have status Pending.

### 5.16.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus

## 5.17 Packagelist (Paginated)

### 5.17.1 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.

### 5.17.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages](https://[servername]:[port]/esig/webportalapi/v3/packages)

### 5.17.3 HTTP Method

GET

### 5.17.4 MIME Type

application/json

### 5.17.5 Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ContinuationToken</b>	Optional	Token which holds information about which set of records needs to be returned.	String
<b>MaxQuantity</b>	Optional	Maximum number of records.	Int
<b>SortField</b>	Optional	Sort the returned records by the field which is specified. (Reserved for future use)	String
<b>SortOrder</b>	Optional	'ASC' or 'DESC' to specify in which way sorting happens. (Reserved for future use)	String
<b>CreatedBeforeDate</b>	Optional	Displays only the packages created before this date. ISO 8601 date format.	Date
<b>Status</b>	Optional	Returns only those packages which have the requested status. The possible statuses are: draft, pending, finished, rejected, revoked, failed.	String

### 5.17.6 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ContinuationToken</b>	Token which allows the client to retrieve the next set of records.	String
<b>MaxQuantity</b>	The highest number of records that the result may contain. Useful when no quantity was given in the request (default subject to change).	Int
<b>Total</b>	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

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Items</b>	List of packages.	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Items (array of objects)</b>	Array, 0..*	Array of objects
<b>PackageId</b>	Package id for this package	String
<b>PackageStatus</b>	Status of the package as a whole: Draft Pending Finished Rejected Revoked Failed <b>Note:</b> a package has the status <b>Failed</b> when a background operation has failed and left a message on the Poison Queue.	String
<b>ExternalPackageReference</b>	Reference given by the calling application. This will not be used by eSignatures.	String

#### 5.17.7 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
}
```

#### 5.17.8 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The packagelist gets returned successfully.
<b>400 Bad Request</b>	A request parameter was invalid.

#### 5.17.9 Error codes



<u>HTTP CODE</u>	<u>CODE</u>
400	Pagination.MaxQuantity.OutOfBounds

## 5.18 Delete Package

### 5.18.1 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.

### 5.18.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{id}](https://[servername]:[port]/esig/webportalapi/v3/packages/{id})

### 5.18.3 HTTP Method

DELETE

### 5.18.4 MIME Type

application/json

### 5.18.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>id</b>	Required	Unique id of the package	String

### 5.18.6 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The package was deleted.
<b>404 Not Found</b>	The package with the given id could not be found.
<b>409 Conflict</b>	The package's status is still in a non-final state (e.g. Pending, Expired) so it can't be deleted yet.

### 5.18.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Package.NotFound
<b>409</b>	Package.InvalidStatus
<b>409</b>	Package.ExpiryTimestampInThePast

## 5.19 Delete Document from Package

### 5.19.1 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.

### 5.19.2 URL

`https://[servername]:[port]/esig/webportalapi/v3/packages/{id}/documents/{documentId}`

### 5.19.3 HTTP Method

DELETE

### 5.19.4 MIME Type

application/json

### 5.19.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>packageld</b>	Required	Unique id of the package	String
<b>documentId</b>	Required	Unique id of the document inside the package	String

### 5.19.6 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>404 Not Found</b>	No document could be found for the given id.
<b>409 Conflict</b>	The package's status is different from [Draft] or the package was made with an old version of the api.

### 5.19.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>404</b>	Document.NotFound
<b>409</b>	Package.ApiVersionMismatch
<b>409</b>	Package.InvalidStatus

## 6. Miscellaneous Services

6.1 [Get DocumentGroups](#)

6.2 [Get Enabled Signing Types](#)

6.3 [Get Version](#)

6.4 [Get Themes](#)

6.5 [Get Contact Groups](#)

6.6 [Get Enabled Languages](#)

## 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]/esig/webportalapi/v3/documentgroups

### 6.1.3 HTTP Method

GET

### 6.1.4 MIME Type

application/json

### 6.1.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>DocumentGroups</b>	List of all document groups	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>DocumentGroups (array of objects)</b>	Document group details	Array of objects
<b>Name</b>	Name or description of the document group	String
<b>Code</b>	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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The documentgroups gets returned successfully.

### 6.1.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No error messages possible

## 6.2 Get Enabled Signing Types

### 6.2.1 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.

### 6.2.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/signingtypes](https://[servername]:[port]/esig/webportalapi/v3/signingtypes)

### 6.2.3 HTTP Method

GET

### 6.2.4 MIME Type

application/json

### 6.2.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningTypes</b>	List of enabled signing types	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>SigningTypes (array of objects)</b>	Array, 1..*	
<b>Name</b>	Name of the signing type for use in Set Process Information or Create Instant package calls.	String

### 6.2.6 Example response

```
{
  "EnabledSigningTypes":
  [
    {
      "Name": "Manual"
    },
    {
      "Name": "BeId"
    }
  ]
}
```

### 6.2.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The signing types list gets returned successfully.

### 6.2.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>None</b>	No errors

## 6.3 Get Version

### 6.3.1 Description

This call checks the version and build number of eSignatures.

### 6.3.2 URL

https://[servername]:[port]/esig/webportalapi/v3/version

### 6.3.3 HTTP Method

GET

### 6.3.4 MIME Type

application/json

### 6.3.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>ProductVersion</b>	Number of the version	String
<b>FileVersion</b>	Number of the file (build)	String

### 6.3.6 Example response

```
{
  "ProductVersion": "5.5.0",
  "FileVersion": "5.5.0.48326.02"
}
```

### 6.3.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The version and build number get returned successfully.

### 6.3.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
<b>None</b>	No error messages possible



## 6.4 Get Themes

### 6.4.1 Description

This call retrieves all the Themes and their code that are currently created in the Configuration Index.

The theme code can be entered as Request parameter when creating Packages and Instant Packages. The packages in question will then have the requested theming.

### 6.4.2 URL

https://[servername]:[port]/esig/webportalapi/v3/themes

### 6.4.3 HTTP Method

GET

### 6.4.4 MIME Type

application/json

### 6.4.5 Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
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

### 6.4.6 Response parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
Items	Optional	List of themes	Array of objects
<u>PARAMETER</u>		<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
Items (array of objects)		Array, 0..*	Array of objects
Name		Name of the theme	String
ThemeCode		Unique id of the theme	String

### 6.4.7 Example response

```

{
  "ContinuationToken": "1",
  "Items": [
    {
      "Name": "Connective Theme",
      "Code": "0000"
    },
    {
      "Name": "System Theme",
      "Code": "00001"
    },
    {
      "Name": "My theme",
      "Code": "00003"
    }
  ],
  "MaxQuantity": 20,
  "Total": 3
}

```

#### 6.4.8 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The themes list gets returned successfully.
400 Bad request	A request parameter was invalid

#### 6.4.9 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

## 6.5 Get ContactGroups

### 6.5.1 Description

This call retrieves all the contact groups and their code that are currently created in the eSignatures WebPortal.

The ContactGroup code can be entered as Request parameter when doing a Set Process Information (v3.1) call or Instant Package Creation call. Any member of the requested contact group will be able to sign the package for the entire group.

### 6.5.2 URL

https://[servername]:[port]/esig/webportalapi/v3/contactgroups

### 6.5.3 HTTP Method

GET

### 6.5.4 MIME Type

application/json

### 6.5.5 Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
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

### 6.5.6 Response parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
Items	Optional	List of contact groups	Array of objects

<u>PARAMETER</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
Items (array of objects)	Array, 0..*	Array of objects
Name	Name of the contact group	String
ContactGroupCode	Unique id of the contact group	String

### 6.5.7 Example response

```
{
  "ContinuationToken": "1",
  "Items": [
    {
      "Code": "00001",
      "Name": "Personal"
    },
    {
      "Code": "00002",
      "Name": "SharedContacts"
    },
    {
      "Code": "00003",
      "Name": "OldGroup"
    }
  ],
  "MaxQuantity": 20,
  "Total": 3
}
```

### 6.5.8 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
200 OK	The contact groups list gets returned successfully.
400 Bad request	A request parameter was invalid.

### 6.5.9 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

## 6.6 Get Enabled Languages

### 6.6.1 Description

This call retrieves the list of languages that have been enabled in the eSignatures Configuration Index and are thus available in the eSignatures Portal.

This call is useful for integrators who need to know which languages are currently enabled in a given eSignatures installation.

### 6.6.2 URL

[https://\[servername\]:\[port\]/esig/webportalapi/v3/languages](https://[servername]:[port]/esig/webportalapi/v3/languages)

### 6.6.3 HTTP Method

GET

### 6.6.4 MIME Type

application/json

### 6.6.5 Response parameters

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>EnabledLanguages</b>	List of enabled languages	Array of objects

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>EnabledLanguages (array of objects)</b>	Array, 1..*	
<b>DisplayName</b>	DisplayName of the language as defined in the Config Index.	String
<b>NativeName</b>	NativeName of the language as defined in the Config Index. The NativeName is the name of the language in its actual language. E.g. français for French.	String
<b>IsoCultureCode</b>	ISO 639-1 language code of the language.	String

### 6.6.6 Example response

```
[
  {
    "EnabledLanguages": [
      {
        "DisplayName": "Dutch",
        "NativeName": "Nederlands",
        "IsoCultureCode": "nl"
      }
    ]
  }
]
```

### 6.6.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The EnabledLanguages list gets returned successfully.

6.6.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
None	No errors

## 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 approval:

### Approval events information

- Time when package was approved
- Time when reminder was sent, content of the reminder, recipient(s) of the reminder
- Time when an invitation was sent to approve the package, content of the invitation, recipient(s)
- For all approval events mentioned above, the approver's IP address is added to the Audit proofs.

The following information is collected in base64 format about each reassignment:

### Reassignment events information

- Time when package was reassigned
- Reason why the package was reassigned
- First name of previous and new approver or signer
- Last name of previous and new approver or signer
- Email address of previous and new approver or signer

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)
- For the signing events mentioned above, the signer's IP address is added to the Audit proofs

### 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.
- Any extra proofs which were added by the client through the Post extra proof call described in section [7.3](#).
- If the setting **IsOneTimeUrlEnabled** is disabled in the Configuration Index, the following value is added for every signature that was placed by means of an "unsecure" URL: **"One-time URL was disabled when signing"**.

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:** In previous versions of eSignatures, the Audit proofs feature had a *significant* impact on the eSignatures database. Since eSignatures 5.4, the Audit proof files are stored on the storage drive of the server that hosts the data (configurable in the **File Storage Settings** of the Configuration Index). This reduces the load on the database but obviously increases the load on the storage drive. The bigger the documents, the more space will be used. The impact grows exponentially with bigger documents. So, make sure the storage has sufficient free space. How much space is needed largely depends on the size of your documents and on the retention time in the repository. Consider the following rule of thumb when using Audit proofs: The Audit proofs will have a size of 1.5x of the original document. If the Intermediate States Saved setting is enabled in the Configuration Index, this size increase is added for every signature placed on the document.

**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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/auditproof/download](https://[servername]:[port]/esig/webportalapi/v3/packages/{packageId}/auditproof/download)

[https://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/auditproof/download/{documentId}](https://[servername]:[port]/esig/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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The Audit proofs xml is returned.
<b>404 Not Found</b>	The documentation/package id could not be found.
<b>409 Conflict</b>	The package is not fully signed.

### 7.2.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
	TBD

## 7.3 Add Proof from External Source

### 7.3.1 Description

This call allows API users to add extra proofs from an external source 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://\[servername\]:\[port\]/esig/webportalapi/v3/packages/{packageId}/auditproof/proofs](https://[servername]:[port]/esig/webportalapi/v3/packages/{packageId}/auditproof/proofs)

### 7.3.3 HTTP Method

POST

### 7.3.4 MIME Type

application/json

### 7.3.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
<b>Content</b>	Mandatory	The actual content of the proof.	Base64 string
<b>LocationId</b>	Mandatory	Location of the signature for which the proof content was generated.	Guid
<b>Name</b>	Mandatory	Name of the proof.	String
<b>Type</b>	Mandatory	A machine-readable "type" key. Can be freely chosen.	String
<b>Description</b>	Optional	Brief human-readable description of the proof.	String
<b>IpAddress</b>	Optional	IP address of the end user of the external source 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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>204 No content</b>	The proof was created for the requested location.
<b>400 Bad Request</b>	The request contained parameters which could not be accepted.
<b>404 Not Found</b>	The documentId or the LocationId could not be found.
<b>409 Conflict</b>	The proof could not be added due to some other reason.

7.3.8 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
	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://\[servername\]:\[port\]/esig/webportalapi/v3/packagecorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/esig/webportalapi/v3/packagecorrelations/{correlationId}/auditproof/download)

[https://\[servername\]:\[port\]/esig/webportalapi/v3/documentcorrelations/{correlationId}/auditproof/download](https://[servername]:[port]/esig/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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
<b>200 OK</b>	The Audit proofs xml is returned.
<b>404 Not Found</b>	The document/package id could not be found.
<b>409 Conflict</b>	The package is not fully signed.

### 7.4.7 Error codes

<u>HTTP CODE</u>	<u>CODE</u>
	TBD

## 7.5 Audit Proofs XML Format

The XSD below describes the structure of the XML that will be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="qualified">
  <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:attribute type="xs:string"
name="ipAddress" use="optional" />
                                        </xs:complexType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            <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:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute type="xs:string" name="documentCorrelationId" use="optional" />
    <xs:attribute type="xs:string" name="documentId" use="optional" />
  </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:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</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 a 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 desired poison queue.
- **Resubmit Poison Queue**, to resubmit the contents of the poison queue to the desired command queue. eSignatures will retry to sign the packages.
- **Clear Poison Queue**, to clears the contents of the desired 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]/esig/webportalapi/v3/poisonqueue

### 8.1.3 HTTP Method

GET

### 8.1.4 MIME Type

application/json

### 8.1.5 Query parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Optional	Token which holds information about which set of records needs to be returned.	String
BusName	Optional	Determines the message bus of which the poison queue will be retrieved. If no value is entered, the poison queue of the message bus defined in the web.config will be retrieved.	
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

<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
ContinuationToken	Token which allows the client to retrieve the next set of records.	String



<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
MaxQuantity	The highest number of records that the result may contain.	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 items in the poison queue.	Array of objects
<u>PARAMETER</u>	<u>CONTENT / DESCRIPTION</u>	<u>TYPE</u>
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 <b>IncludeStackTrace</b> 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

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
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]/esig/webportalapi/v3/poisonqueue`

### 8.2.3 HTTP Method

POST

### 8.2.4 MIME Type

application/json

### 8.2.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
BusName	Optional	Determines the message bus to which the poison queue will be resubmitted. If no value is entered, the poison queue of the message bus defined in the web.config will be used.	String

### 8.2.6 Response parameters

N/A.

### 8.2.7 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
204 No content	The contents of the poison command queue are resubmitted successfully to the command queue.

### 8.2.8 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]/esig/webportalapi/v3/poisonqueue

### 8.3.3 HTTP Method

DELETE

### 8.3.4 MIME Type

application/json

### 8.3.5 Request parameters

<u>PARAMETER</u>	<u>OCCURRENCE</u>	<u>CONTENT/DESCRIPTION</u>	<u>TYPE</u>
BusName	Optional	Determines the message bus of which the poison queue will be cleared. If no value is entered, the poison queue of the message bus defined in the web.config will be cleared.	String

### 8.3.6 Response parameters

N/A.

### 8.3.6 Response codes

<u>RESPONSE STATUS CODE</u>	<u>DESCRIPTION</u>
204 No content	The contents of the poison command queue are cleared successfully.

### 8.3.8 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. A default signature policy will be configured by the system admin in the Config Index. This signature policy will be applied if the **SignaturePolicyId** parameter is left empty. If you want to use a customized signature policy, see **Appendix IV of Connective – eSignatures 6.x – Configuration Documentation** to learn how to do so.

- 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 6.x – 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 points wide and 70 points 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 points x 120 points. 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).

Since 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
    }
  ]
}
```



## 15.3 SigningField with FieldId reference

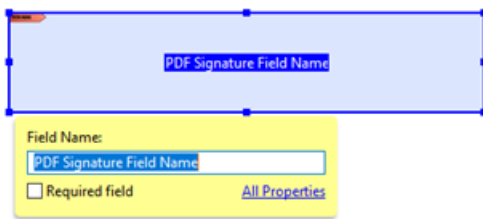
If the PDF documents you're sending contain:

- empty signature fields (created in Adobe Acrobat for instance)
- or input text fields (also created in an application such as Acrobat )

that you want to convert into SigningFields, you can use the **FieldId** parameter.

In case of empty signature fields, the Field Name of such a field will be matched against the value given in the parameter **FieldId** to see if the id of the field is equal.

In case of input text fields, the name property of the input field will be matched against the value given in the parameter **FieldId**. **Important:** the name property is not necessarily the one displayed when you hover over the field. The name can only be reliably verified when opening the PDF in Acrobat Professional's "Form Edit" mode.



Note that it's currently not supported to combine the **FieldId** parameter when selecting pdfa1a or pdfa2a as **TargetFormat** in the Add document call. When using the **FieldId** parameter it's currently mandatory to select pdf as **TargetFormat**.

### 15.3.1 Detection of PDF Text Fields

Pay attention if your PDF documents contain input text fields. Input text fields are only converted into SigningFields if the name property of a such a text field corresponds to the Text Field Format (regex) you or your administrator configured in the Configuration Index. The original text field will not be displayed. This is intended behavior.

If you want to upload PDF documents with text fields, and prevent eSignatures from converting them into SigningFields, 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. Using the default regex, this means your Text Field Name should not start with a # if you do not want them converted.

## [a-zA-Z]+(?:\d\*.)?\d+.

### 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
- 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)
- The + indicates 1 or more occurrences of the previous sub-expression

**Note:** Text Fields can't be added in eSignatures itself but must be added to the pdf before upload.

### 15.3.2 Example

```
{
  "Name": "string",
  "Language": "en",
  "IsOptional": false,
  "ExternalReference": "string",
  "Elements": [
    {
      "Type": "SigningField",
      "ExternalReference": "myBeidField",
      "FieldId": "signme",
      "SigningMethods": [
        "beid"
      ],
      "Location": {
        "Page": 2,
        "Top": 200,
        "Left": 200
      },
      "Dimensions": {
        "Width": 200,
        "Height": 200
      }
    }
  ],
  "ProofCorrelationId": "string",
  "DocumentOptions": {
    "targetType": "application/pdf",
    "pdfOptions": {
      "targetFormat": "pdfa1a"
    },
    "base64data": "string",
    "contentType": "application/pdf"
  }
}
```

For more information, see [Create SigningField element](#)

# 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 three possible values: "Approver", "Signer" or "Receiver".

### **Actor.RedirectTypeInvalid**

The RedirectType can only have one of the following values: "AfterCompletion", "AfterDelay" or "Immediately".

### **Actor.RedirectTypeNotAllowed**

No RedirectUrl was defined.

## 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 Stakholder'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.

## 11.6 Package

#### **Package.NotFound:'package id'**

The package with id 'package id' could not be found.

#### **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.ContainsNoDocuments:'packageId'**

Package with id ['package id'] contains no documents

#### **Package.ContainsDocumentWithNoSigners:'packageId'**

Package with id ['package id'] contains a document with id ['document id'] with no signers

## 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.

## 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 points.

### **SigningField.InvalidWidthCoordinate**

Signing field width needs to be larger than 112 points.

### **SigningField.InvalidHeightMarker:'markerOrFieldId'**

The given marker id contains a height which is smaller than 70 points.

#### **SigningField.InvalidWidthMarker:'markerOrFieldId'**

The given marker id contains a width which is smaller than 112 points.

#### **SigningField.MarkerOrFieldIdNotFound:#SignatureAgent**

The marker or field id for the given signature agent was not found or did not confirm to requirements. Consequently, the signing field was not created.

See section [10.2 Signing location with id reference](#) for more information on how to use markers.

#### **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.

#### **Stakeholder.UnsupportedStakeholderType**

The provided stakeholder type is not supported. The supported values are: **Person**, **PersonGroup** and **ContactGroup**.

## **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.