MitID Integration

What it does

We have a generic integration to Criipto (a MitID, and other eID, broker) and an account, allowing for sign in and document signing with MitID/nemID.

Sign in

Sign in can be setup as a webinterface or as SSO from the login-page (not fully supported yet), the authentication method isn't available in the dropdown for the webinterface, as of writing, it has to be set through the database (the live database).

```
UPDATE forminterface SET AuthenticationType = 5 WHERE InterfaceID = [ID];
```

Document signing

One or multiple documents can be send for signature, one or more people can sign the batch and it is possible to enforce CVR or CPR. This process can be initiated from a StatusAction or be accessing a url.

A page is appended to the signed documents showing who signed it and when, in the order they signed it.

Prereq

To setup this an "Application" has to be set up in Criipto, one for sign in and one for signatures, and a CNAME dns-record has to be created.

The ID's and secrets created here will be needed later.

Setup in Criipto

First, a domain has to be added.

Haed to the Criipto Dashboard, select "Domains" in the menu, make sure that you are in "Production", click "Add production domain".

Name it `[customer]-eid.tsnocode.com` and head to CloudFlare and add a CNAME record that points to `idp.criipto.id`.

Once the domain is active you can progress.

Sign in

Coming

Document signing

Requires version 7336 or newer.

Head to "Application" in the Criipto menu, click "Add signatures application" (if it exits), otherwise click "Add login application" and add `?tags=signatures` to the end of the url.

Name the application `[customer] eSign`, select their domain and check the eID's that should be available, select `java` as technology.

A secret might pop up or be shown, take note. It is possible to add more after the fact and re-issue them.

Setup in TS

There are a lot of parameters available. Some of these must be set for the integration to work, depending on the integration.

Sign in

Coming

Policies

- oauthCriiptoAllow
- oauthCriiptoHost
- oauthCriiptoClient

- oauthCriiptoSecret

Document signing

To start the signing process, setup the configuration and execute a Status Action that executes the codeunit `dk.tempusserva.signing.criipto.CriiptoStatusAction` or `dk.tempusserva.signing.criipto.CriiptoStatusActionGenerator` or `dk.tempusserva.signing.criipto.CriiptoStatusActionGeneratorAlt` or access a url with `command=dk.tempusserva.signing.criipto.CriiptoPage&SagID=[SagID]&DataID=[DataID]` (not ready).

This will try to lookup and send the document(s) out for signing.

Configurations

Configuration	Description
Signer.MultipleSigners	Whether multiple signers is allowed (true/false)
Signer.MaximumSigners	How many signatures are needed. Defaults to "COUNT", which counts the number of signers, when using MultipleSigners, otherwise 1. Can otherwise be set to a number.
Signer.ExpiresInDays	Number of days the recipient has to sign the document. Default is 30.
Signer.FieldMaximum	Not implemented
Signer.FieldFil *	System name for the field with documents that should be send, all documents found here will be send. Defaults to "FILES". Also used to store generated files.
Signer.FieldCPR	System name for the field with a CPR, that the signer has to have to sign the document. Also used with MultipleSigners.
Signer.FieldCVR	System name for the field with a CVR, that the signer has to have to sign the document. Also used with MultipleSigners.
Signer.FieldEmail *	System name for the field with an email, that will be notified about the signature request. Also used with MultipleSigners. Defaults to "EMAIL".
Signer.FieldSigners	System name for a list-of-children-field. All records found here will be required to sign the document. Required when using MultipleSigners.
Signer.StatusError	Status that the record should enter if the signature request failed (was rejected or timed out). Defaults to 0.
Signer.StatusSigned	Status that the record should enter when all signatures are collected. Defaults to 0.
Signer.StatusUploaded	Status that the record should enter when it has been uploaded to signing service. Defaults to 0.

Signer.StatusDisableCodeunits	Whether codeunits should be executed or not, when the signing completes or fails. Defaults to false (do execute).
Signer.EmailSubject	The subject of the email send to the signer. Defaults to "Dokument til signering".
Signer.EmailBody	The email-body of the email send to the signer. Defaults to "Du kan underskrive her: {LINK}".
Signer.NotificationSubject	The subject of the email send to EmailWarner, when all signatures have been collected. Defaults to "Dokument til signering er blevet underskrevet"
Signer.NotificationBody	The body of the email send to EmailWarner, when all signatures have been collected. Defaults to "Alle parter har nu underskrevet dokumentet. {LINK}". Links to the record.
Signer.NotificationBodyExt	The body of the email send to all signers of a document, when all have signed. Defaults to "Alle parter har nu underskrevet dokumentet. Du kan downloade det underskrevne dokument her: {LINK}". Links to the signed document at Criipto.
Signer.EmailWarner	Can be an email or the system name of a field containing an email. The email found here will be notified when a signature fails and completes. If an email is not found, the email of the current user will be used, if not a status action. Required for status action.
Signer.WarningSubject	The subject of the email send to EmailWarner, when a signature request fails. Defaults to "Dokument til signering blev afvist".
Signer.WarningBody	The body of the email send to EmailWarner, when a signature request fails. Defaults to "En underskrift blev afvist. {LINK}". Links to the record.
Signer.CriiptoClientID *	The client ID from Criipto application.
Signer.CriiptoClientSecret *	The client secret from Criipto application.
Signer.OverwriteOnReupload	true/false, default false. If enabled the system allows re-sending a record for signing, overwriting the old one.
Signer.FileName	Name of generated file, defaults to kontrakt.docx, used by CriiptoStatusActionGenerator
Signer.TemplateID	ID of template to be rendered and saved before sending it of to be signed, used by CriiptoStatusActionGenerator
Signer.FileNameAlt	Name of generated file, defaults to kontrakt.docx, used by CriiptoStatusActionGeneratorAlt
Signer.TemplateIDAlt	ID of template to be rendered and saved before sending it of to be signed, used by CriiptoStatusActionGeneratorAlt

Revision #1
Created 4 April 2025 12:17:40 by Theis Villumsen
Updated 4 April 2025 12:18:59 by Theis Villumsen