TrueSign Integration for OnBase

Release 3.1

March 2021

www.imagesoftinc.com

25900 W. 11 Mile Rd., Suite 100 ▪ Southfield, MI 48034 ▪ phone: (248) 948-8100 ▪ fax: (248) 948-8146

Revision History

Version	Date	Description	Author
1.0	05/05/2020	Initial release	J. Vataj
2.0	06/22/2020	Release with new sample life cycle	J. Vataj
3.0	08/30/2020	Updated sample life cycle with August release	J. Vataj
3.1	03/20/2021	Added troubleshooting steps	J. Vataj

Table of Contents

Introduction

TrueSign can be integrated with any number of applications. In this document we will guide organizations who use OnBase by Hyland on how to integrate with ImageSoft's TrueSign product. We will be going through an overview of what TrueSign is and then create a simple workflow to perform basic TrueSign actions from OnBase. To learn more about TrueSign, visit TrueSign.com.

Online Version

This guide is also available online here.

Resource Files

All the resource files needed for this guide are available here.

This zipped folder contains:

- TrueSign Next Sample Life Cycle export file
- Installer for the ImageSoft Service Bus Listener
- OnBase Unity Scripts
- OnBase User Forms

TrueSign

Introduction

TrueSign is a web application that helps organizations jump on the digital signature train fast and securely. Documents can be sent anywhere in the world to be signed in real time and make it back to your system in no time. TrueSign allows you to sign interactively from OnBase (the OnBase user is signing the document) or by sending the documents up to TrueSign.com where they can be signed via any browser.. TrueSign supports signing by internal/named users (I.e. someone inside the organization, such as a department manager) or by external parties (such as a vendor).TrueSign is comprised of the following modules:

- **ImageSoft Identity Server**: This app is the authenticator for TrueSign. Named users will be required to login via this app and then redirected to the TrueSign application. Anyone can register for this app at our Identity site: https://identity.imagesoftinc.com
- **TrueSign App**: This is the main component of TrueSign, with a rich interface for users and administrators. In this application, users can login, review and sign envelopes. The admins, on the other hand, have all the resources needed to manage their TrueSign account. Each user is invited to TrueSign by an account admin. The first admin of any organization is invited to TrueSign by ImageSoft when their account is created. The app is located at: https://app.truesign.com
- **TrueSign API**: The API app is the backbone of the TrueSign solution. It allows any organization to integrate with TrueSign in a straightforward way, using any coding language. It contains different versions of the TrueSign API, and it is well documented for developers. You can find the TrueSign API at: https://api.truesign.com

Requirements

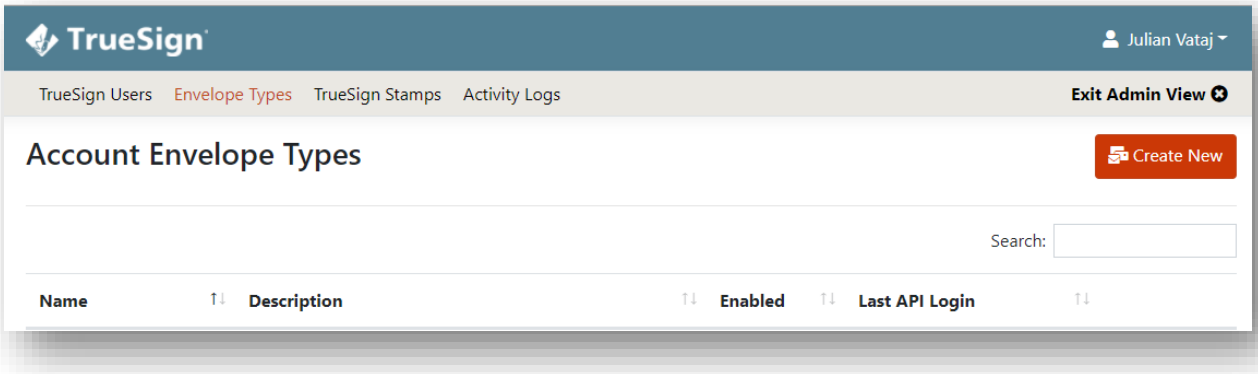
The following are base requirements for any organization integrating with TrueSign:

- A TrueSign subscription with at least one user.
- An Envelope Type from TrueSign with credentials to connect to the TrueSign API.
- A machine with an internet connection and the ability to connect to the following domains (including their subdomains):
    - https://*.windows.net (Microsoft Azure resources)
    - https://*.truesign.com (TrueSign resources)
- Port 5671 opened for outbound traffic on the server where the Service Bus listener is installed.

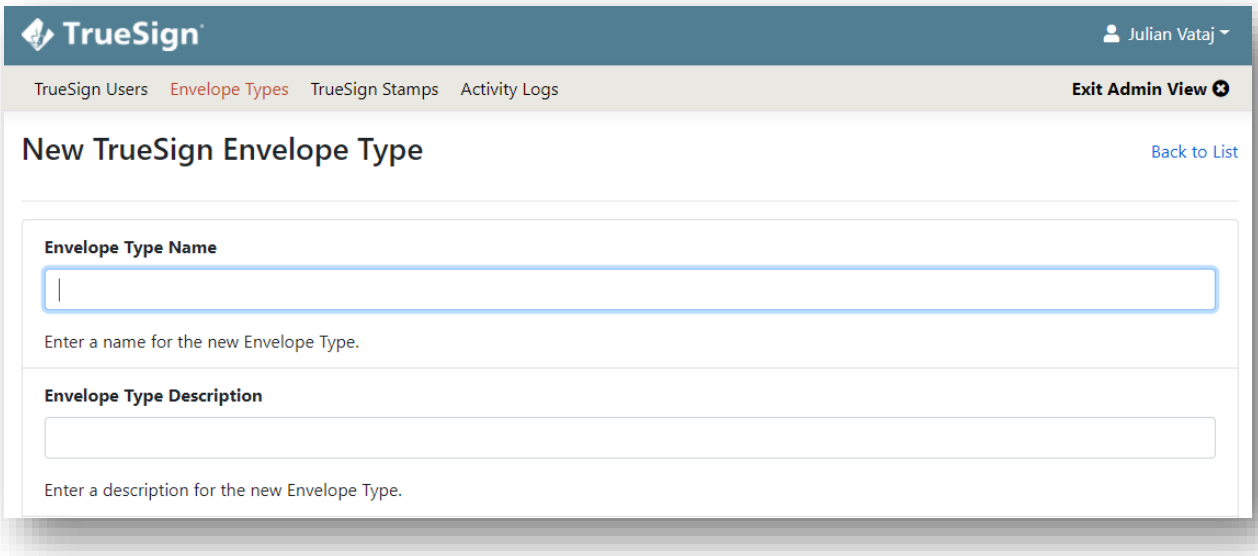
Configuring an Envelope Type on TrueSign

Before we begin importing the provided sample life cycle, you need to have an envelope type created in TrueSign. An envelope type is just a container that we will be creating envelopes under. The envelope type will give us the credentials needed to connect to the TrueSign API. You may choose to have one envelope type for each OnBase environment, but you may create as many as you need: for example, one for each division of your organization. There are reasons why you might want to create more than one envelope type, one of which is that there are certain settings that can be applied at the envelope type level.

To create a new envelope type, you must be a TrueSign admin. Browse to https://app.truesign.com/admin/envelopetypes (you might be asked to authenticate) and then click the "Create New" button:



Then you will be redirected to the "New TrueSign Envelope Type" page, where you will enter a name and other settings for the new envelope type:



Once done editing, click the "Create Envelope Type" button all the way to the bottom to save.

About Envelope Delivery Methods

TrueSign allows you to configure different delivery methods for a created Envelope Type. You will always be able to retrieve an envelope that was uploaded to TrueSign by calling the TrueSign API. The other delivery methods are:

- **ImageSoft Service Bus**: An Azure Service Bus provided by ImageSoft. Messages are sent from TrueSign to an application connected to the service bus for further processing. In this example, the ImageSoft Service Bus Listener is the application connecting to the service bus and processing messages that TrueSign sends over. These messages are then loaded into OnBase via the OnBase Unity API, where workflow then takes over to complete the processing.
- **Email**: You may choose to receive the completed envelope JSON for an Envelope Type via email. You may choose that the JSON is placed on the body of the email, attached as a JSON file, or both.
- **Web Hook**: You may choose to receive the completed envelope JSON for an Envelope Type via a HTTP(s) POST to a given URL. You can also add a list of headers and TrueSign will post them with the request.

You choose one default delivery method for each Envelope Type you create. However, you may override this delivery method with another pre-configured method when a new instance of an envelope is created. Please check out our API documentation for more information at https://api.truesign.com/docs/.

Order of Operations

TrueSign requires any integrator to complete the following steps, in the order shown below, to successfully create and complete an envelope. These operations are made by calling the TrueSign API (but we have done all that for you in the included OnBase Workflow export).

- **Create** a new envelope – A title for the envelope is the only thing required for this operation.
- **Add documents** to the created envelope – A list of metadata for the documents that will be in this envelope is sent to TrueSign. TrueSign will return a list of links where you should upload the actual bytes of these documents. **Note**: This step can be included on the Create step.
- **Upload** the content of the documents – Here you will upload the actual bytes of the document to Microsoft Azure Storage, using the URL that TrueSign returned in the prior step.
- **Add a signer** (internal or external) – At this step, you will either tell TrueSign an internal user's email or the full name and email address of an external signer (if the organization has signed up for external signing). For external signers, it is possible to require an access code for them to confirm before signing.

- **Send** the created envelope – Call the TrueSign API with the envelope ID to send the envelope to the required signer. An internal signer will get a notification based on their preferences and an external signer will always receive an email.
- **Download** or wait for a message – Once the envelope has been signed by the required signer, call the TrueSign API to get the signed envelope (in an interactive signing case) or wait for the ImageSoft Service Bus Listener to receive the completed.

Note: The download script will create a temporary file on the server running the script. This file will be deleted once a new revision is created in OnBase.

Best practice

It is recommended that you integrate with TrueSign in the non-interactive way. This will give your users the ability to sign without having to be in OnBase, using any supported device (mobile, desktop, tablet, etc.).

It is also recommended that you place the documents that have been sent to TrueSign in a "Waiting" queue, to make sure that no one else is updating these documents in OnBase. Once the documents make it back into OnBase, you should move the received documents from the waiting queue to their next step.

Integration with OnBase

Introduction

The integration between TrueSign and OnBase is simple and easy to setup and works from the Thick, Thin, and Unity Clients. OnBase will be the initiator of the envelope creation via scripting in Workflow. Using the out of the box ImageSoft Service Bus delivery method documents will re-enter OnBase, once completed in TrueSign, almost instantaneously and without interrupting the users' work. ImageSoft provides a base life cycle export that can be imported using OnBase Studio and, with minor tweaks, the integration between OnBase and TrueSign can be achieved in your solution in under an hour. Because of the wide array of possibilities provided by the TrueSign API, an organization can build advanced TrueSign solutions in OnBase with the help of an in-house developer or ImageSoft's Professional Services team.

Note: This is only a sample life cycle. It is intended to give you an idea on how to integrate with TrueSign from OnBase. Please do not use it in your production environment. Instead, study the logic this sample life cycle has and implement something similar, fitting your business requirements, in your production system.

Components of the Export File

The quickest way to integrate TrueSign with OnBase is to import the provided sample life cycle via OnBase Studio. If you are unfamiliar with OnBase Studio or the import process in general, please reference the Hyland's OnBase Studio module reference guide.

The import will create the following in your OnBase system:

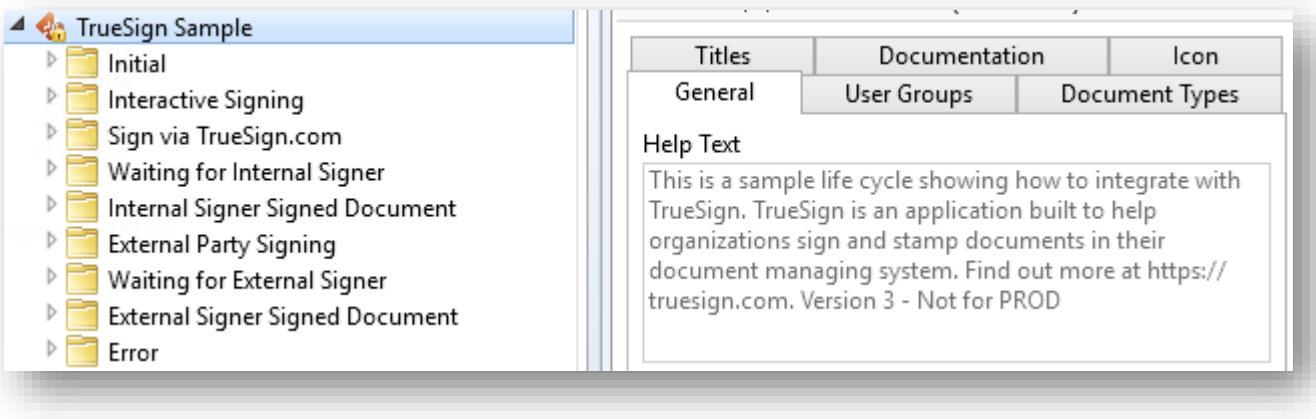
- A **document type group** named "TrueSign Next" with the following document types:
    - **TrueSign Completed Envelope** – This document type holds the completed envelope's JSON and is created by the ImageSoft Service Bus Listener.
    - **TrueSign Test Document** – This document type is for testing purposes only. It contains the following keywords:
        - **Signed:** this keyword is used to set a *TRUE* or *FALSE* value indicating if the document was signed by the required signer. The value can be change by modifying the TrueSign Next Library.
        - **Stamped:** this keyword is used to set a *TRUE* or *FALSE* value indicating if the document was stamped by the required signer. The value can be change by modifying the TrueSign Next Library.
        - **TrueSign Envelope Id:** this keyword is used to store the envelope ID assigned by TrueSign. This ID can be used to complete other operations with the TrueSign API

later. These operations can be deleting the envelope from TrueSign, downloading the history, re-downloading the signed documents, etc.

- **TrueSign Envelope Doc Handle:** this keyword is used to store the OnBase document handle for the TrueSign Completed Envelope document, which contains the JSON message sent to the ImageSoft Service Bus Listener by TrueSign. This keyword is helpful troubleshooting any issues with non-interactive signing.

- An example **life cycle** named "TrueSign Sample", which is a multipurpose life cycle: it handles the JSON files imported from the ImageSoft Service Bus Listener and it serves as a test life cycle where test documents are added so they can be signed.

Note: TrueSign currently only supports singing PDF documents.



The lifecycle contains the following queues:

- **Initial** – contains system work required to download completed envelopes from TrueSign back into OnBase. A new document will be created by the ImageSoft Service Bus Listener with the configured document type. This document type is associated with the TrueSign Next life cycle and system work will read the contents (JSON) of the file when the download script is called. The life cycle also contains the following Ad Hoc User Tasks:
  - **Forward to Interactive Signing:** transitions the selected document(s) to the Interactive Signing queue.
  - **Forward to Sign via TrueSign.com:** transitions the selected document(s) to the Sign via TrueSign.com queue.
  - **Forward to External Party Signing:** transitions the selected document(s) to the External Party Signing queue.

- **Interactive Signing** – in this queue, you sign documents in an interactive way from within OnBase. For users familiar with older versions of TrueSign, this was the only way to sign in those versions.
    - **TrueSign Now**: allows you to sign an envelope (technically the documents within the envelope) in an interactive way. This task will upload the selected document(s) to TrueSign, setting the signer account to the email address you entered above, and open a browser window (via a user form) for the user to sign and complete the created envelope. Once the user closes the opened windows, a workflow action, running the download script, will download the signed documents and create a new PDF revision of the document in OnBase.
    - **TrueSign Design**: This task will upload the selected document(s) to TrueSign and set a designer for the envelope. The designer, who is a user with a valid TrueSign account, can then go to the TrueSign Designer interface and: modify signers, modify envelope documents, and send the envelope for signature.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **Sign via TrueSign.com** – in this queue, you sign documents in a non-interactive way from OnBase. System work in this queue will automatically create a new envelope and upload the document(s) to it for the required signer, per the property bags you configured above (API credentials, envelope title, signer, etc.). The envelopes created in this queue are signed by the required signer outside of OnBase and the ImageSoft Service Bus Listener will be responsible for importing the completed envelope back to OnBase, unless the delivery method for the authenticated envelope type is set to something else.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **Waiting for Internal Signer** – documents uploaded to TrueSign from the "Sign via TrueSign.com" queue are placed in this queue. Once the signed document comes back from TrueSign, the document is moved to "Internal Signer Signed Document" queue.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **Internal Signer Signed Document** – Signed documents from the "Sign via TrueSign.com" queue are placed in this queue.
    - **TrueSign History**: retrieves all the history for the selected document from the last time it was signed in TrueSign. A user form will then display a table with the history. ***Note***: This user form contains sample code on how to connect to the TrueSign API via JavaScript.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.

- **External Party Signing** – in this queue you send documents for signing to external users outside your organization. ***Note***: The External Signer module is needed in your TrueSign subscription for this functionality to work.
    - **Send to External**: creates an external envelope with the selected document(s) and prompts the user (via a user form) to input the required signer's email, first and last name, etc. Once the envelope is created, the document(s) are moved to the Waiting for External Signer queue.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **Waiting for External Signer** – documents uploaded to TrueSign from the "External Party Signing" queue are placed in this queue. Once the signed document comes back from TrueSign, the document is moved to "External Signer Signed Document" queue.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **External Signer Signed Document** – Signed documents from the "External Party Signing" queue are placed in this queue.
    - **TrueSign History**: retrieves all the history for the selected document from the last time it was signed in TrueSign. A user form will then display a table with the history.
    - **Send to Initial**: transitions the selected document(s) to the Initial queue.
- **Error** – Documents that have encountered an error in the prior steps will be moved into this queue with an error note attached. Envelope JSON, that comes into OnBase from the ImageSoft Service Bus Listener, can also end up in this queue if an issue is encountered while processing.
    - **Reprocess**: Reprocess envelope JSON that came in from the Service Bus Listener and errored out. This will attempt to download the signed envelope back into OnBase.
    - **Remove**: remove the selected document(s) from workflow.

- A **Unity Library** and two **IWorkflow** scripts: These are the core components of the integration and rely on properties set by workflow actions. Changes to these scripts should be vetted by someone who understands C# and REST APIs.
    - **TrueSignNext Library** – a Unity script library that contains core logic to connect and call different TrueSign API methods.
    - **TrueSignNext Upload** – an IWorkflow script that is responsible for connecting to TrueSign, creating envelopes and adding files to TrueSign storage. References the TrueSign Next Library.

- **TrueSignNext Download** – an IWorkflow script responsible for downloading envelopes from the TrueSign API and reading the JSON content. This script will also create a new revision for the documents in OnBase. References the TrueSign Next Library.

Note: TrueSign API requires TLS 1.2 to be enabled. This is forced by the Unity Scripts.

- A collection of 4 **HTML forms**: These forms will help you to collect values for property bags needed in the unity scripts. All these forms are for demo purposes.
  - **TrueSign Next Launch** – a form that launches a browser page to the TrueSign Viewer for interactive signing. This form will read a property bag with the envelope ID that was set in the upload script. The main purpose of this form is to "pause" the workflow logic and wait for the user to sign the documents and close the user form.

Note: Since Internet Explorer is not supported, this form will attempt to launch TrueSign via Microsoft Edge. Make sure your Edge browser is updated to the Chromium version. See the "About the TrueSign Launch User Form" section below for more detail.

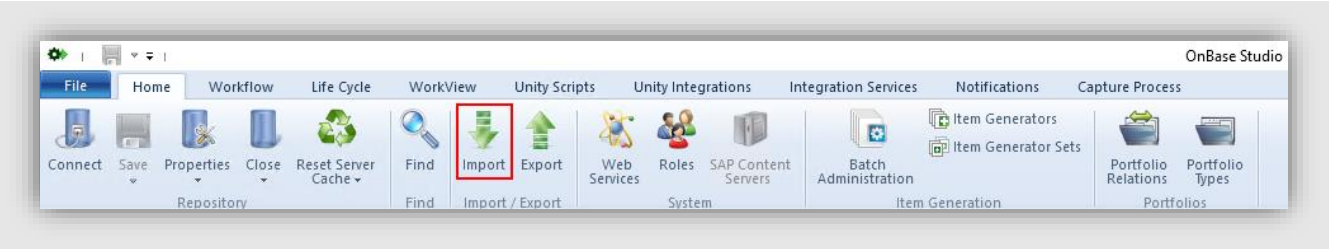
  - **TrueSign Next New Envelope** – a form that prompts the user to select a title for the new envelope and pick an internal signer from a dropdown list. The data is then saved in property bags to be consumed by the unity scripts later.
  - **TrueSign Next New Envelope – External** – a form that prompts the user to select a title for the new envelope and input all required information for an external user. The data is then saved in property bags to be consumed by the unity scripts later.
  - **TrueSign Next History** – a form that connects to the TrueSign API via the credentials previously stored in property bags using jQuery/JavaScript. All history is retrieved from TrueSign for the last envelope that contained this document.

Note: These forms reference jQuery and Bootstrap hosted on a CDN. If the end user is extremely locked down, these files might need to be downloaded and referenced locally.

Note: You may choose not to import this life cycle into your environment, in which case, you will need to manually create all these components.

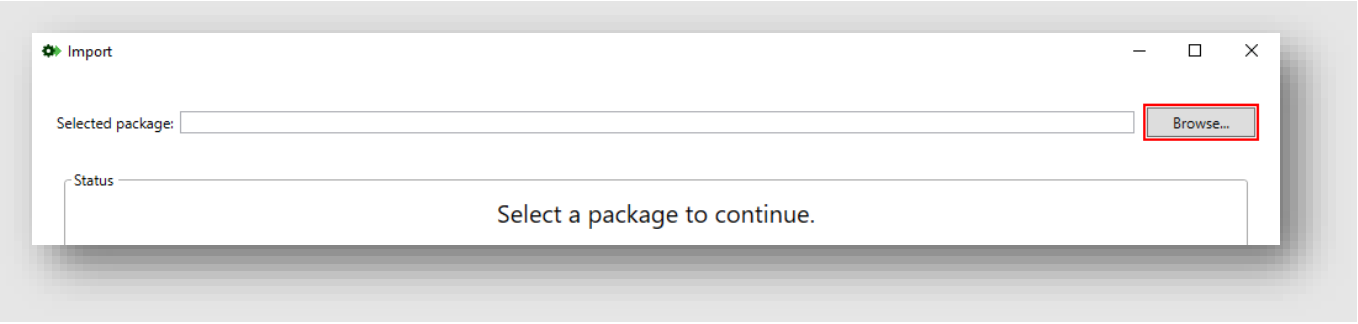
Starting the Import

Open OnBase Studio and login to the correct application server. Once you're logged in, click the "Import" button on the "Home" tab:



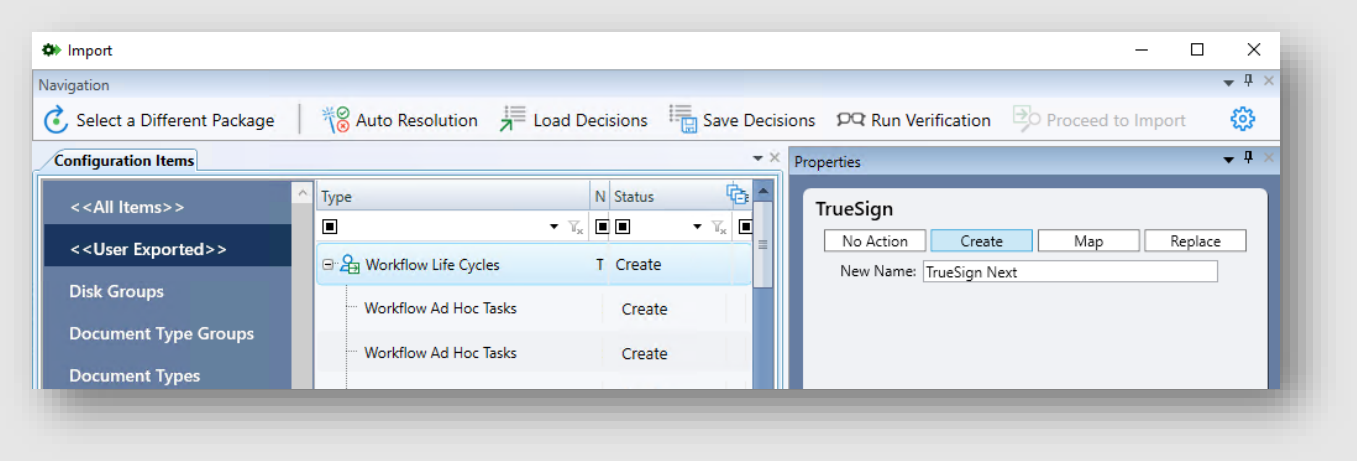
This will open the import screen and direct you to browse to the location of the life cycle export provided by ImageSoft.

Note: The provided export might be from a different OnBase version than the one you have. Please reference the OnBase Studio module reference guide to resolve this conflict.



Once the export file has been selected, the decision screen will appear. Please make sure to select the items described in the "Export Contents" section and either create or map them to your existing items. Once you have made all the necessary decisions, click the "Run Verification" button to make sure every required decision has been made.

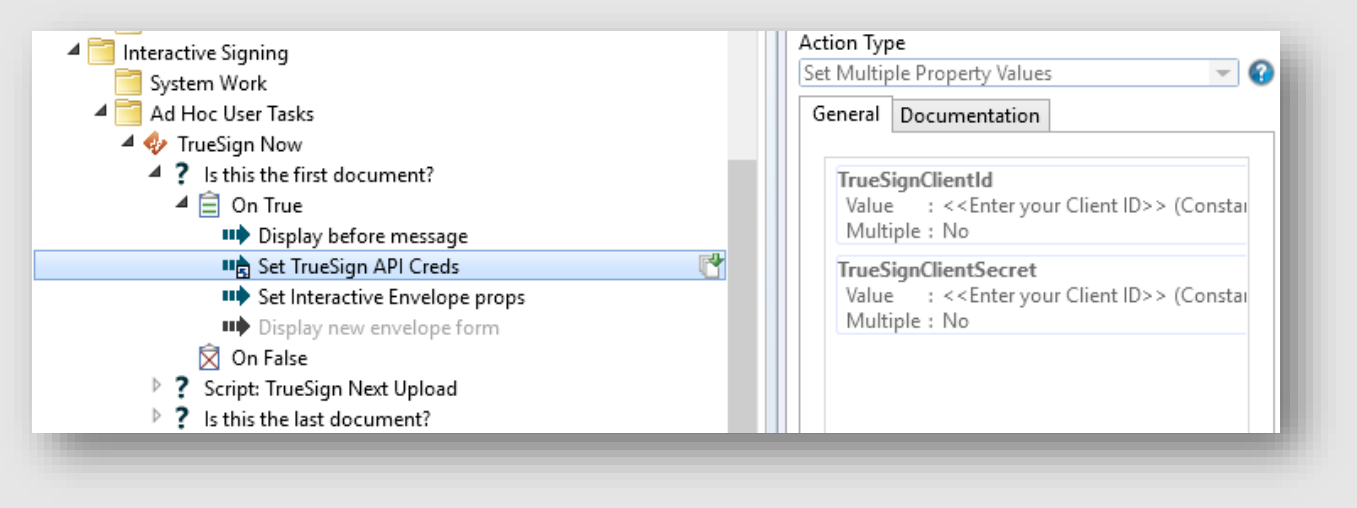
Note: ImageSoft also provides a "Decision" file that you can load from the import screen. These decisions will create the required resources and map the common ones.



Once all the decisions have been made, click the "Proceed to Import" button and follow the steps to save all the changes, and complete the import. Once the import has finished, you will find a new life cycle in your life cycle list called "TrueSign Sample".

Updating the Sample Life Cycle

Now that you have successfully imported the sample life cycle, it is time to add your own API credentials to connect to TrueSign. You will need credentials for an envelope type, which can be found in the Admin View of the TrueSign application. Each envelope type comes with a client ID and two client secrets. Copy the client ID and one of the secrets for the envelope type you are connecting with and update the "Set TrueSign API Creds" action with the correct values for the property bags.



Once the TrueSign API credentials have been updated, we need to update the rest of the property, depending on the features you want to use. Please refer to the property bag page for a full list.

Once all the changes have been made successfully, save the repository in Studio and reset the server cache. At this point you can only test with the "TrueSign Now" Ad Hoc Task in the Interactive Signing Life Cycle since the Service Bus Listener has not been configured yet.

About the TrueSign Launch User Form

While ImageSoft encourages you to sign in a non-interactive way while integrating from OnBase, we are also committed to help you sign documents right from the OnBase client. When signing a document interactively from OnBase, it is necessary to pause the OnBase workflow so the user can open the envelope in the TrueSign viewer, sign the document and then return to OnBase for further processing. Here is where we use the "TrueSign Next Launch" user form. This user form contains JavaScript code that will read a property bag called "TrueSignEnvelopeId" and launch a new browser window to a URL that looks like this: https://app.truesign.com/Viewer/Envelope/<TrueSignEnvelopeId>/true.

Currently, TrueSign Next does not support Internet Explorer, due to IE not supporting technologies like web workers. The launch form will attempt to open the above URL in an Edge window, since the latest version of Edge supports URI handlers. The user form will execute the following code to launch the viewer window: microsoft-edge:https://app.truesign.com/Viewer/Envelope/<TrueSignEnvelopeId>/true. This method works for all OnBase clients.

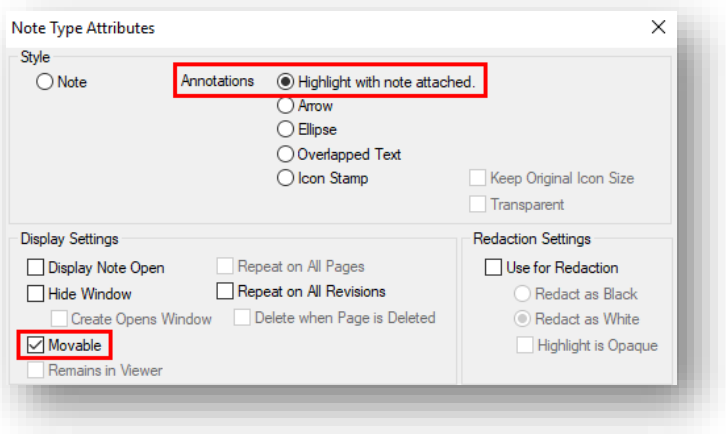
Since Google Chrome and Firefox do not have a URI handler, interactive signing is only available in the Chromium version of Microsoft Edge.

Note: You can test if this will work in the end users' machine by opening any browser and going to **microsoft-edge:https://truesign.com**. If this command does not open Microsoft Edge, please contact support.

Interactive signing is the only thing restricted to Edge only. The TrueSign Web app, where you would sign envelopes outside of OnBase, supports Google Chrome, Firefox, Microsoft Edge (Chromium) and Safari.

About Anchor Signing

Anchor signing is possible in TrueSign when signing TIFF documents and the envelope has only one required signer. Anchor signing is done via an OnBase Note Type acting as a "Sign Here" flag. The OnBase note type must be configured as an "Annotation" with the "Highlight with note attached" option. The display setting should be set to "Movable":



Once the note type has been configured, the Note Type Id needs to be entered as the constant value of the **TrueSignSignerNoteId** property bag. In the OnBase client, the user adds the highlight annotation on the opened TIFF document over the places where the single required signer for this document needs to place their signature. The TrueSign Upload Unity script will then check if the new envelope contains only one signer and if the documents being uploaded have a note of the note type with the same id you entered in the **TrueSignSignerNoteId** property bag. If the document has such note(s), then the script will create TrueSign anchor objects for each note on each document and append them to the signer object. This way when the required signer opens the envelope in TrueSign, will see the "Sign Here" anchors that require their signature.

About Multiple Signers

The TrueSign Upload Unity script is built in a way to support property bags containing a single value or an array of values. This allows you to use the same script to send to a single signer or multiple signers. TrueSign supports a mix of signers for an envelope and signing is done in series by each signer. Anchor signing is not supported from OnBase when requiring multiple signers to sign an envelope. While you are free to implement scripts that make this possible via the TrueSign API, ImageSoft recommends that the TrueSign Designer is used in these “complicated” scenarios.

Troubleshooting

Here are some common issues encountered while integrating OnBase with TrueSign.

Unable to connect to the TrueSign API

The server where the script is running from must allow outbound traffic to the TrueSign API (https://api.truesign.com) and the scripts must use Tls 1.2. To test if the server is able to reach the TrueSign API, open PowerShell ISE and run the following code:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
$wc = New-Object system.Net.WebClient;
$result = $wc.downloadString("https://api.truesign.com/v1/status")
Write-Output($result)
```

This call should return a successful response similar to the following:

```
{"callerIP":"x.x.x.x","serverTime":"2020-09-24T15:31:06.8245524+00:00","status":"OK"}
```

Unable to launch Microsoft Edge

The provided OnBase user form uses the Microsoft Edge URI handler to launch a new Edge window with the TrueSign Viewer for the user to sign an envelope in interactive signing. Make sure that the machine you are launching TrueSign from has been upgraded to Microsoft Edge (Chromium). Then open another browser (IE, Chrome) besides Edge and type in the URL address: microsoft-edge:https://app.teruesign.com and hit enter. This should launch a new Edge window/tab and take you to the TrueSign page. If nothing happens when you hit enter, and the Edge browser has been installed, then you need to make sure the Edge URI handler has been registered in the machine’s registry. You can save the following settings to a .reg file and run it to ensure the URI handler is registered:

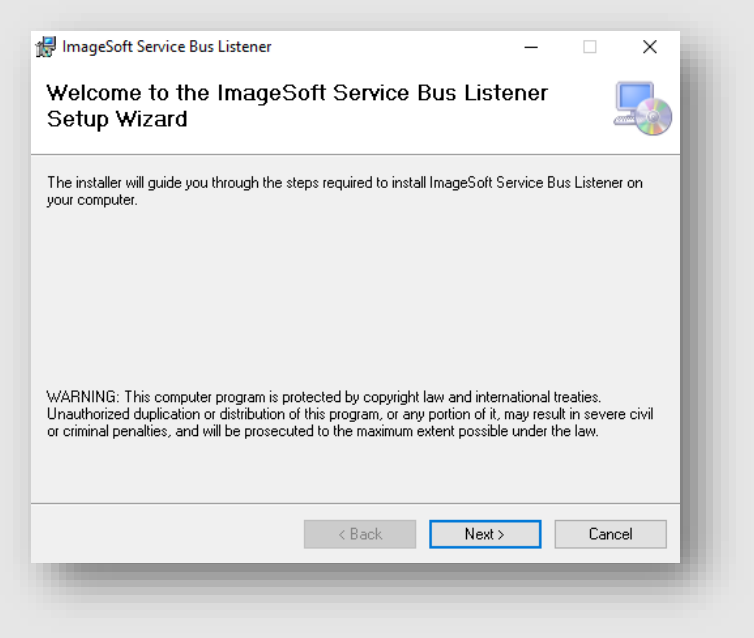
```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT\microsoft-edge]
"URL Protocol"=""
@=URL:microsoft-edge
```

ImageSoft Service Bus Listener

The ImageSoft Service Bus Listener is a Windows service application that listens to an Azure Service Bus queue or topic and bring the message that the queue or topic sent down into your system. It allows you to either, store the received message as a document into OnBase, or create a file in the server's file system. We will configure this application to listen to a TrueSign envelope type and store the received message as a new document into OnBase via the Unity API.

Installation

Run the msi or setup file in the ImageSoft Service Bus Listener folder that came with the OnBase resources zip file. This will begin the installation process and show you the setup wizard:

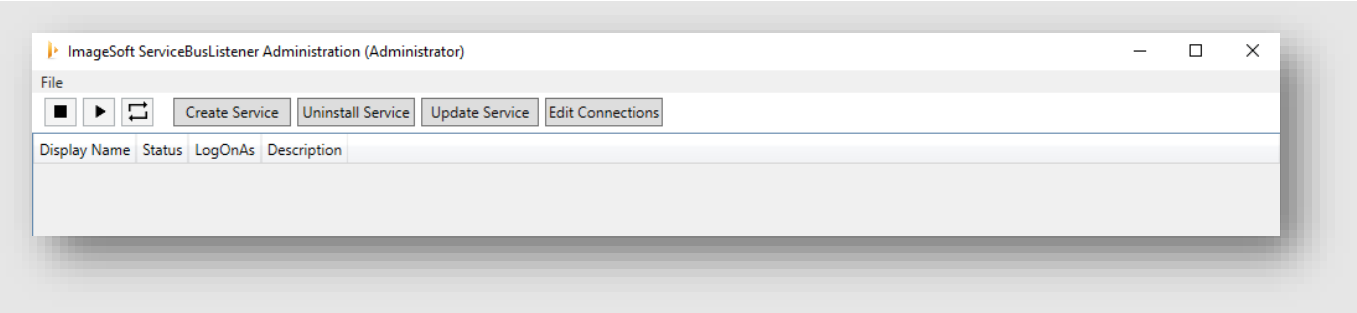


Click next and select the folder where the listener should be installed in. Click next and then finish to complete the setup.

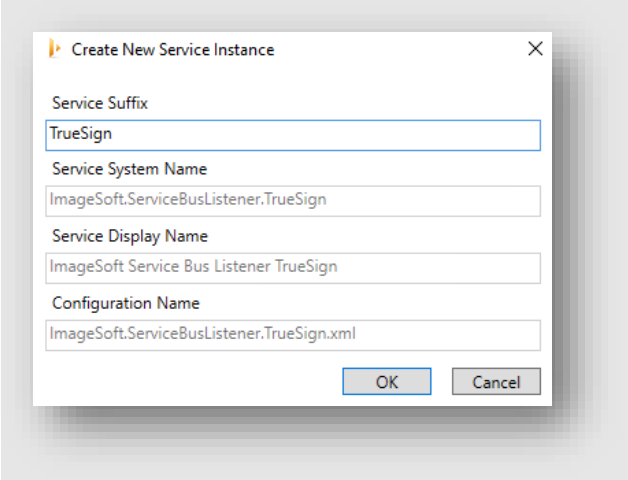
Configuration

Once the ImageSoft Service Bus Listener has been installed on the machine, you will find a shortcut to the application on the desktop called "ImageSoft ServiceBusListener Admin". Double-click the shortcut to launch the application.

Note: The application will prompt you to run it as an administrator. The user must be an admin to run this application because we are creating Windows services on the machine.

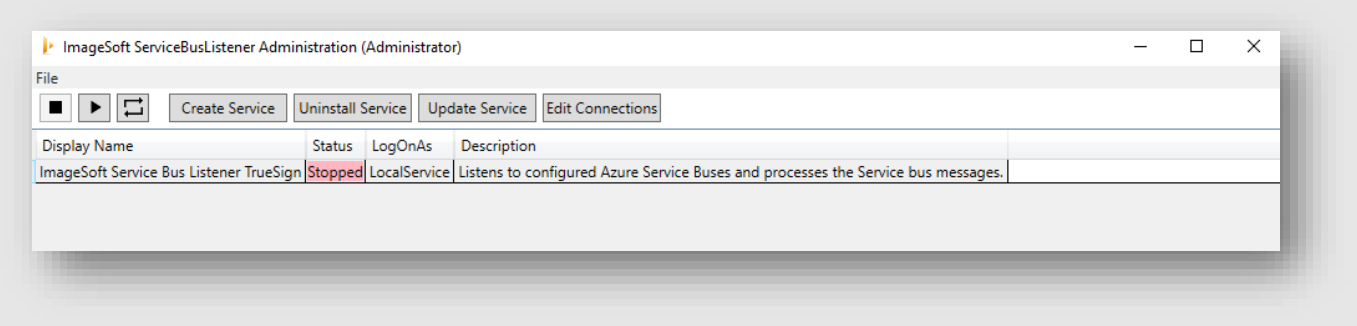


After the initial installation the administration window will not list any services, so let's create one for TrueSign. Click the "Create Service" button to start. The "Create New Service Instance" window will appear, where we will choose a suffix for the service's name. Type "TrueSign" and click "OK".

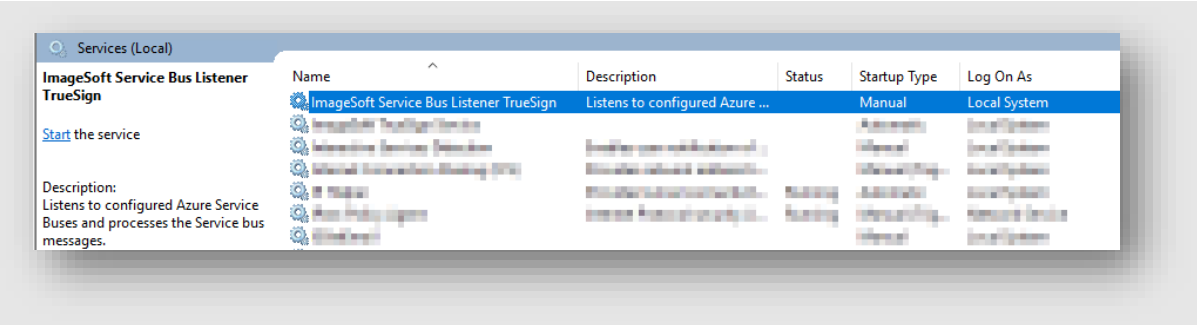


You will see an "Install Complete" message, click OK to continue. You will now be prompted with an informational message reminding you to set any desired service properties via the Window's Services Console.

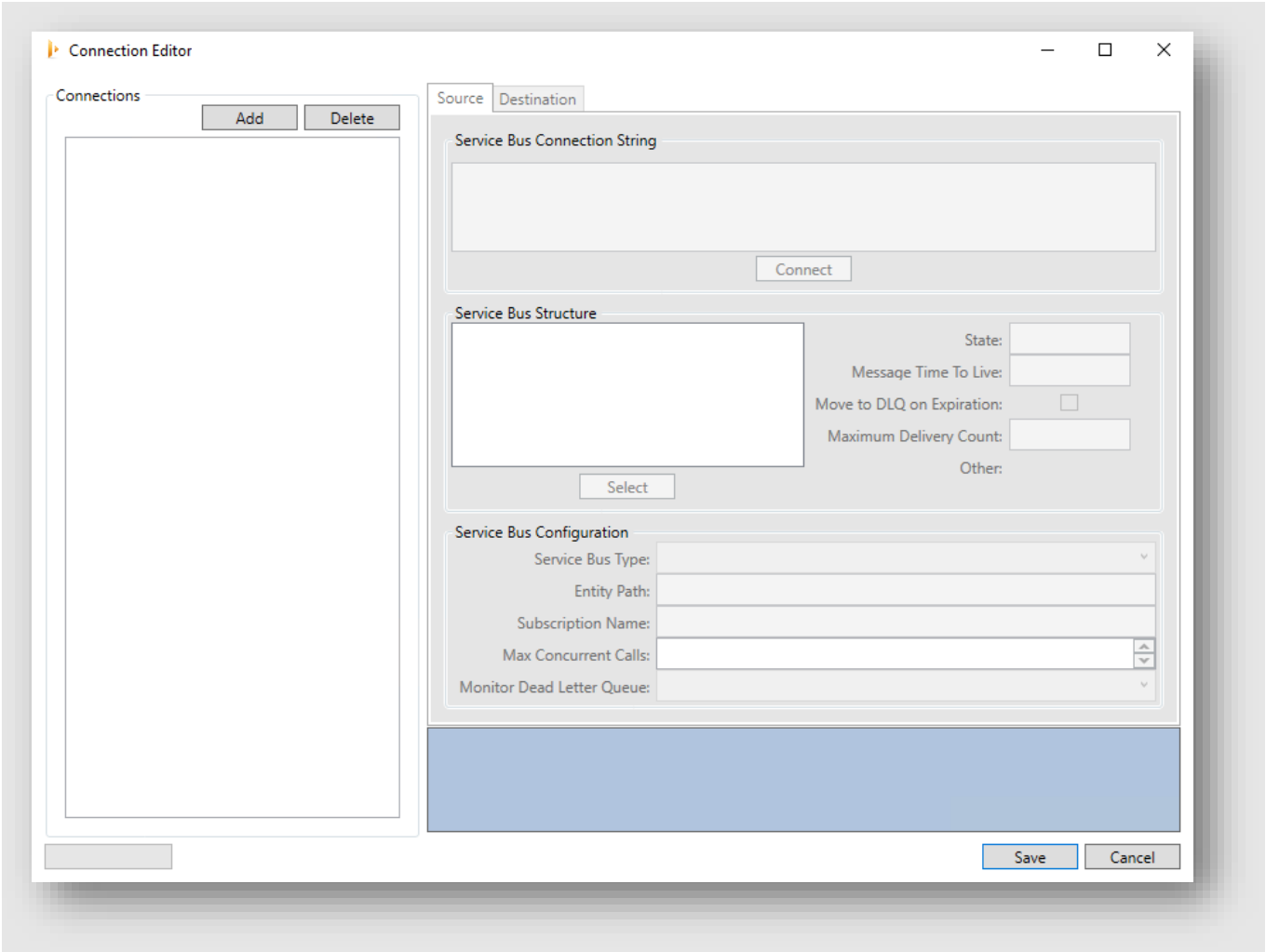
Now you will see one service listed in the admin window: ImageSoft Service Bus Listener TrueSign.



The service is in a stopped status and configured to run as LocalService. You may change the service's Startup Type and Log On As from the Window's Services Console. Once you have the service configured and running correctly, please ensure the Startup Type is set to be Automatic.

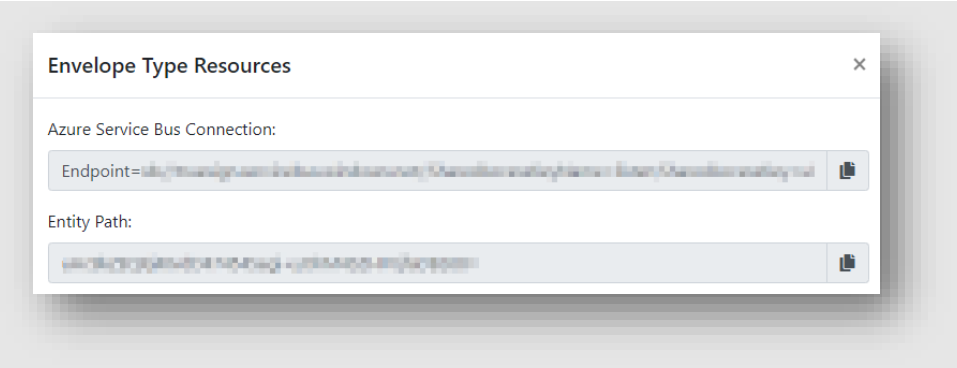


We will now configure the new service we created to connect to our envelope type's Azure Service Bus queue, to listen for messages that TrueSign sends when an envelope has been completed by a user and the Envelope Type's delivery method is ImageSoft Service Bus. Select the TrueSign service in the admin window and click the "Edit Connections" button. This will bring up the "Connection Editor" window where we will connect to the Azure Service Bus and OnBase.

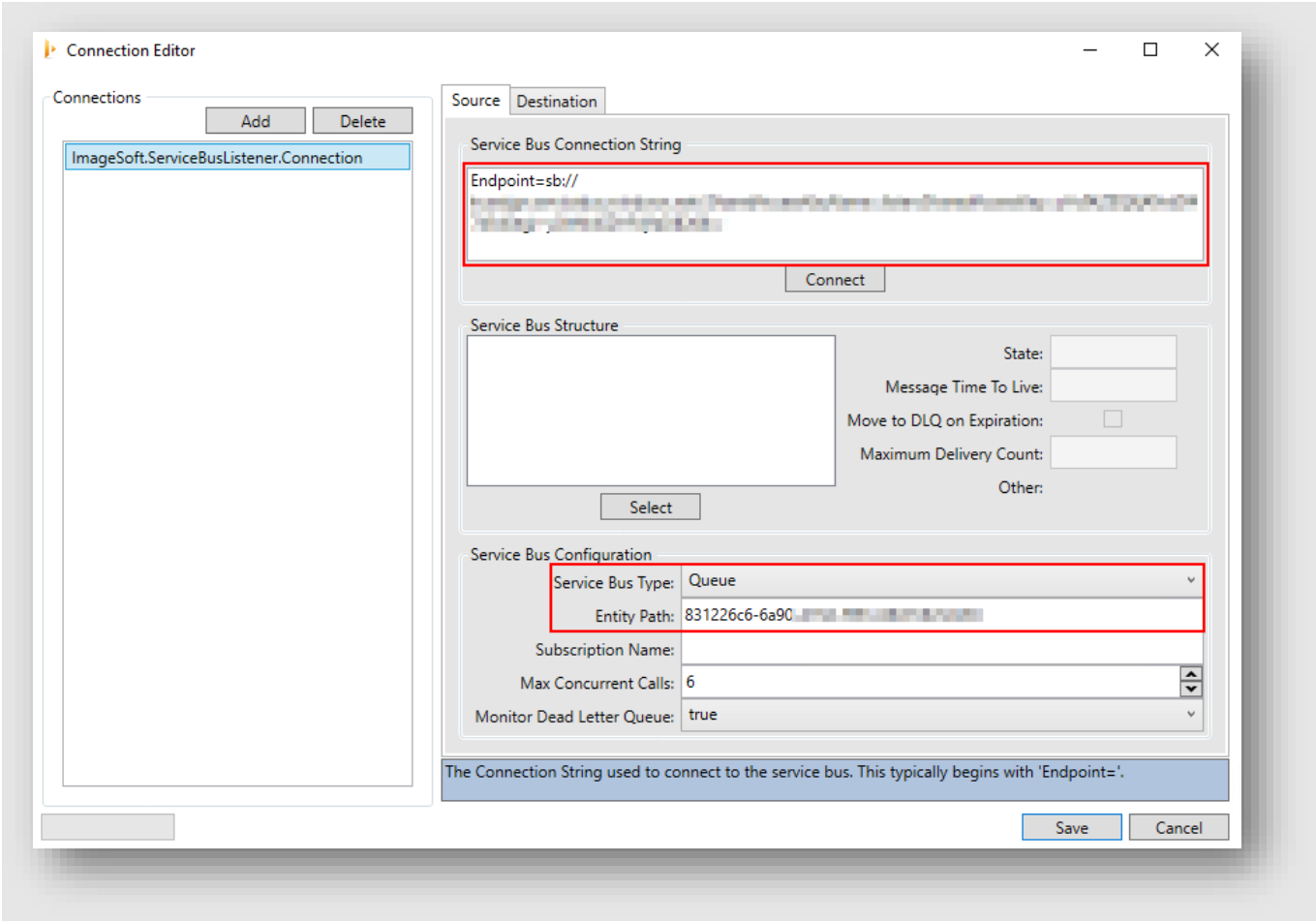


Click the "Add" button to create a new source connection. A new connection will appear in the connections list. Click the new connection and the "Source" tab will be highlighted. It is time to go to the TrueSign application and retrieve the service bus connection string from the Envelope Type in the Admin View. Once you have found the Envelope Type you want to connect to, click "Details" and then click the

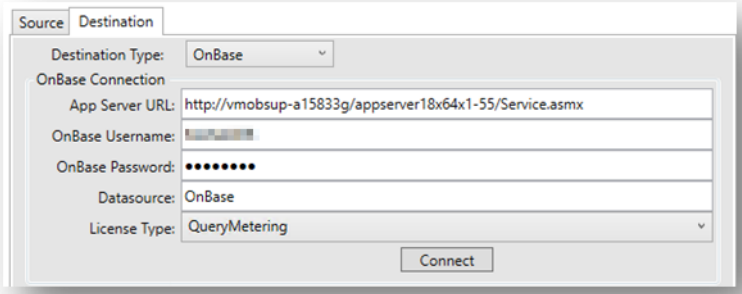
'key' icon next to the Service Bus deliver method. The following window will show you the connection string and entity path you need:



Here are the completed connection settings for the "Source" tab. Only the highlighted fields have been modified.

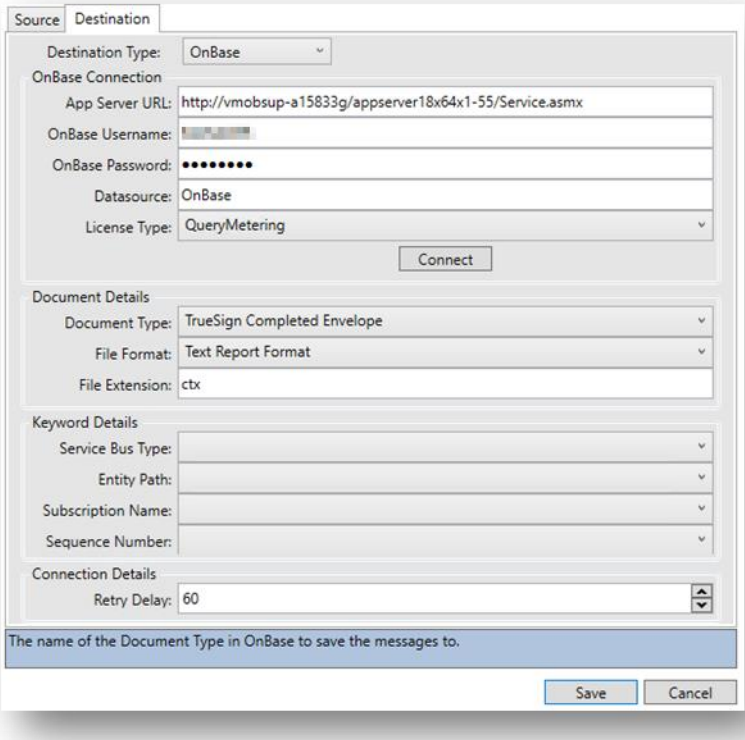


Once done with the Source connection strings, we need to configure the Destination for the incoming messages. In the Destination tab, we will configure the destination to OnBase and select a document type where the new messages will be stored under. For this step you will need an OnBase user with access to the "TrueSign Completed Envelope" document type that was created during the life cycle import. Provide the connection information for OnBase and then click "Connect".



Note: This connection will use the OnBase Unity API. Make sure your organization is properly licensed.

If the connection was successful, the "Document Type" dropdown will be populated with document types that the user you are connecting with has access to. Make sure the "TrueSign Completed Envelope" document type is selected. If you do not see this document type, please check user access and/or reset server cache. Here is a screenshot of the completed settings for the OnBase destination:



Click "Save" to save the settings and close the window. Now we are ready to start the service! Select the TrueSign service and click the start icon: ▶

The service now should be in a "Running" status and that means the service was configured successfully.

OnBase Property Bags for TrueSign

In the sample life cycle, we use session property bags to save data that is then retrieved by the Upload Unity Script and the launch TrueSign user form. This table lists and describes all the property bags, alphabetically:

Name	Description
TrueSignAnchors	Holds anchor data in JSON, for each selected document, generated by the Upload script.
TrueSignClientId	Holds the TrueSign Client Id of the envelope type that will be connecting to the TrueSign API.
TrueSignClientSecret	Holds the TrueSign Client Secret of the envelope type that will be connecting to the TrueSign API.
TrueSignCodeDesc	Holds a single or an array of string values representing an external envelope's optional access code description. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignCodeVal	Holds a single or an array of string values representing an external envelope's optional access code value. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignDesign	Holds a Boolean (true/false) value indicating if the envelope should be sent to sign or it should be marked for design. **Note**: The TrueSignDesigners must be populated.
TrueSignDesigners	Holds an array of strings representing the email addresses of the users that will design the envelope. The email must match the TrueSign account email for the user.
TrueSignEnvelopeId	Holds a GUID of the new envelope, populated from the Upload script. This property is reused in case of multiple documents being upload on the same envelope (example: when selecting multiple documents in a queue and clicking TrueSign Now).
TrueSignEnvelopeTitle	Holds a string that represents the title of the new envelope.
TrueSignExternal	Holds a single or array of Boolean (true/false) values if the required signer is an external signer. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignFirstName	Holds a single or an array of string values representing the first name(s) of the required signer(s) for the envelope. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignInteractive	Holds a Boolean (true/false) value indicating if the envelope should be signed interactively – via the Launch TrueSign user form.

TrueSignLastName	Holds a single or an array of string values representing the last name(s) of the required signer(s) for the envelope. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignNotifySigner	Holds a Boolean (true/false) value indicating if the **first internal** signer of the envelope should be notified or the new envelope. This is should be set to false in case of interactive signing.
TrueSignSignerEmail	Holds a single or an array of string values representing the email address(es) of the required signer(s) for the envelope. **Note**: in a multiple signer situation, the index must match to the correct signer – see example below.
TrueSignSignerNoteId	The note type ID (from OnBase Config) that is used as a *Sign Here* flag. Anchors will be built by the Upload script based on notes of this type located on the documents being uploaded to the TrueSign envelope. **Note**: this only works when there is one signer for the envelope.

Example of Multiple Signers

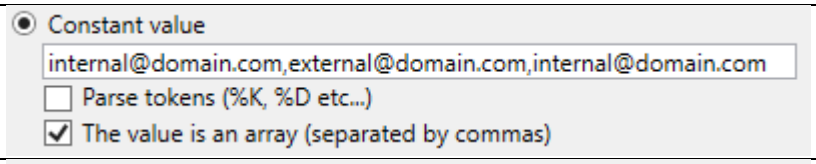
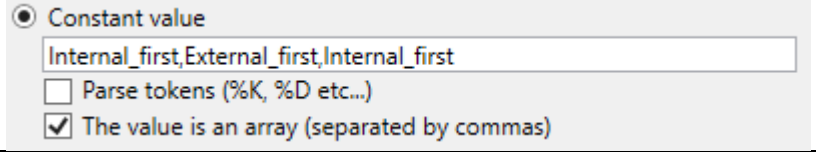
Let's say we have a document that needs to be signed by one internal signer, one external signer and then by another internal signer. The following relevant property bags should be populated as follows:

TrueSignSignerEmail	internal@domain.com	external@domain.com	internal@domain.com
TrueSignFirstName	Internal_first	External_first	Internal_first
TrueSignLastName	Internal_last	External_last	Internal_last
TrueSignExternal	False	True	False
TrueSignCodeDesc	*NULL*	Enter last 4 of SSN	*NULL*
TrueSignCodeVal	*NULL*	1234	*NULL*

If these properties are being populated from a workflow action and:

a) Using a **keyword value**: make sure the "Set property to all value instances" is checked.
b) Using a **constant value**: make sure "The value is an array (separated by commas)" is checked.

Screenshots of the above example populating the property bags via a constant value:

TrueSignSignerEmail	
TrueSignFirstName	

TrueSignLastName	Constant value Internal_last,External_last,Internal_last Parse tokens (%K, %D etc...) The value is an array (separated by commas)
TrueSignExternal	Constant value false,true,false Parse tokens (%K, %D etc...) The value is an array (separated by commas)
TrueSignCodeDesc	Constant value ,Enter last 4 of SSN, Parse tokens (%K, %D etc...) The value is an array (separated by commas)
TrueSignCodeVal	Constant value ,1234, Parse tokens (%K, %D etc...) The value is an array (separated by commas)

The same thing can be done using multi-instance keywords.