TrueSign Integration for OnBase

Version #9

January 2024

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
9.0	12/21/2023	Skipping to version 9.0 to synchronize Document version with Sample Life cycle version. Updated full document to reflect updates made to	Mike Ewald

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 i3 Vertical'sTrueSign 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.

For technical information about accessing the TrueSign API, check out the TrueSign API documentation here: https://api.truesign.com/docs/index.html

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 Sample Life Cycle export file
- Installer for the Service Bus Listener
- OnBase Unity Scripts
- OnBase User Forms (HTML)
- TrueSign Icons
- A test document you can use in the sample life-cycle

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 i3 Verticals 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)

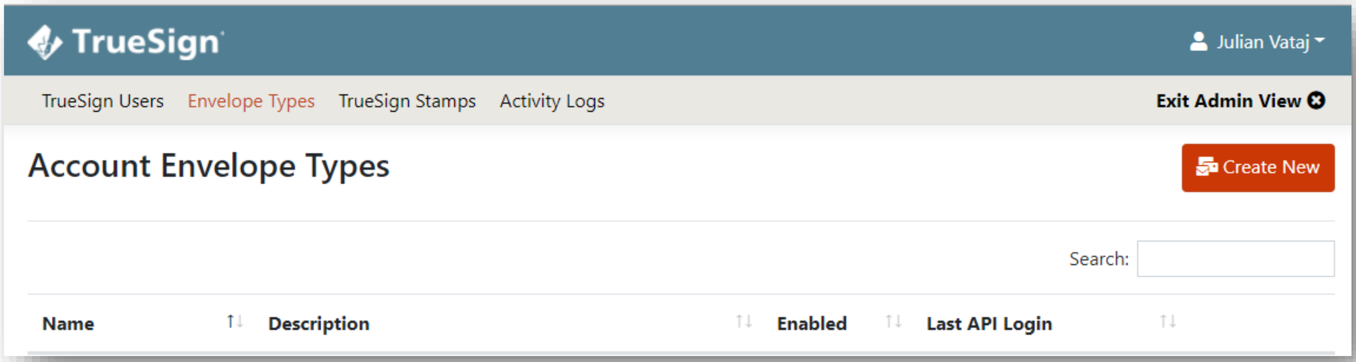
  - https://*.truesign.com (TrueSign resources)

- 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.

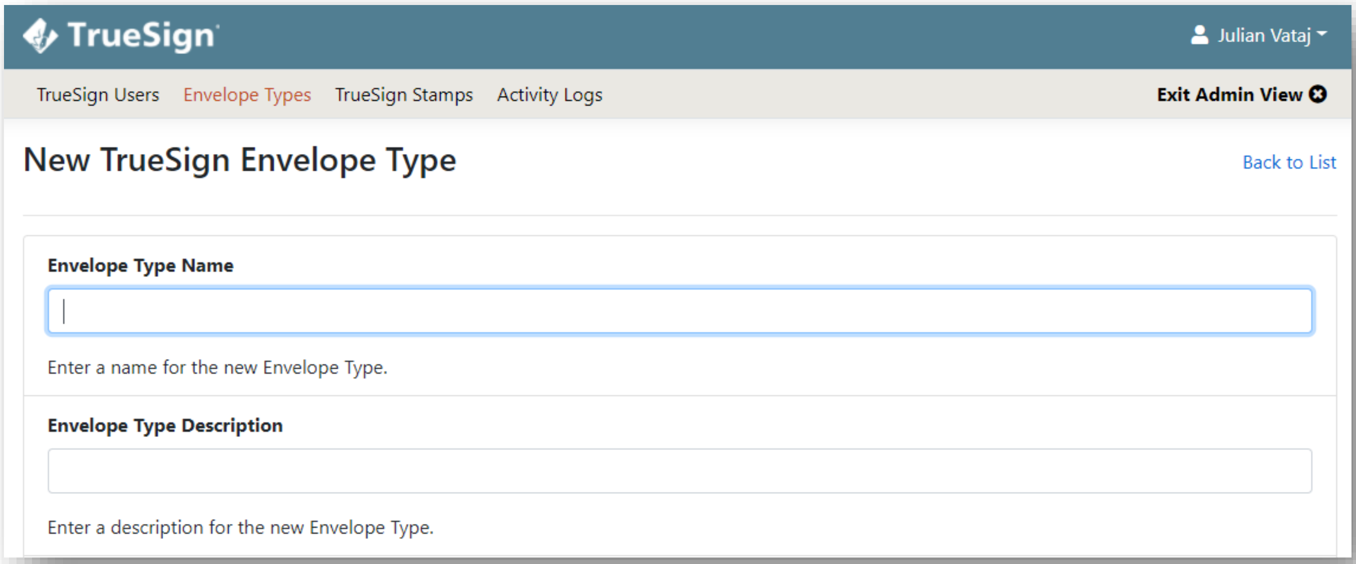
To create a new envelope type, you must be a TrueSign admin. Browse to https://app.truesign.com/admin/envelopetypes (you might be asked to authenticate) and then click the "Create New" button:



Account Envelope Types

Name	Description	Enabled	Last API Login

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:



New TrueSign Envelope Type

Envelope Type Name

Enter a name for the new Envelope Type.

Envelope Type Description

Enter a description 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:

- **Service Bus**: An Azure Service Bus provided by i3 Verticals. Messages are sent from TrueSign to an application connected to the service bus for further processing. In this example, the 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).

1. **Create** a new envelope – A title for the envelope is the only thing required for this operation.
    - This step is executed in the “TrueSign Create Envelope” Unity Script
2. **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.
    - This step is executed in the “TrueSign Create Envelope” Unity Script; this script not only creates a new envelope, but also initializes which signers / designers should be associated with the new envelope!
3. **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.
    - This step is executed in the “TrueSign Add Document” Unity Script
4. **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.

    - This step is executed in the “TrueSign Add Document” Unity Script

5. **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.
    - This step is executed in the “TrueSign Send Envelope” Unity Script
6. **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 envelope.
    - This step is executed in the “TrueSign Download” Unity Script. If you review this script, you will notice that there are two separate sections: One section for downloading an envelope interactively (such as by using the “TrueSign Now” ad hoc task), and another section for downloading the envelope via a message received by the Service Bus Listener.

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.

- In the context of the sample lifecycle, “interactive” signing refers to using the “TrueSign Now” ad hoc task, wherein the full process of uploading the document to TrueSign and signing the document happens at once. With “interactive” signing, the entire signature process starts and finishes within OnBase.
- An example of “Non-interactive” signing is the “Send for Signature” ad hoc task, where the signer can sign and complete the document without ever opening OnBase. 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

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 Service Bus delivery method documents will re-enter OnBase, once completed in TrueSign, almost instantaneously and without interrupting the users' work. I3 Verticals 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 i3 Vertical'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.

Features of the Sample Life Cycle

Sign a document directly from OnBase

You can quickly and easily electronically sign a document yourself using the “TrueSign Now” button in Workflow. Your document will near instantly be uploaded to TrueSign.com, followed by a pop-up window displaying your document in the TrueSign.com signing interface. Simply sign the document, return to OnBase, select the “I’m Done Signing” button, and watch your OnBase document update with a new signed revision.

Send a document for someone else to Sign

Let’s say you need to send a document for someone to sign that doesn’t use OnBase. The “TrueSign Design” button in Workflow allows you to prepare a document to be signed by another user. After clicking this button, the TrueSign Designer interface will allow you to select which users need to sign the document and pre-position the regions that need to be signed.

After designing your envelope, click the “Complete” button, and the signer(s) will receive an email notification letting them know that they have a document they need to sign. (In OnBase, the document will be sent to an “Awaiting Signature” queue while we wait for the signer to complete the document). After all the signers have either signed or rejected the document, the document will automatically be updated in OnBase. This process can occur asynchronously thanks to the Service Bus Listener, which receives messages whenever envelopes are completed and sends those messages to OnBase.

Once the document has been signed, the OnBase document can automatically be routed to a “Completed” queue so that users can review the signed documents and proceed with the next step of the document’s life cycle!

Automatically Upload Documents to TrueSign

The sample life cycle provides a simple framework you can use as an example for building a fully automated TrueSign process. Imagine this: you have a simple sign-off form that needs to be created monthly, which needs to be signed by members on an approval board. You could use the OnBase Document Composition module to compose a document, add the document to a TrueSign Envelope, and send it to the appropriate parties for signature without any human interaction needed. You can even pre-position the signature anchors on the document to simplify the end-user signing experience. This can be accomplished by converting your document to an image and applying special OnBase Notes included in the Sample Export such as “TrueSign Anchor (Sign Here)”. Check out the “Automatic TrueSign Upload” queue in OnBase Studio for more inspiration for how to build your own automated process.

Monitor Documents Pending Signature

Once a document has been uploaded to TrueSign and sent for signature, you may want to monitor the status of the envelope, send reminders to users, or cancel the TrueSign envelope altogether. The sample export provides an example of an “Awaiting Signature” queue with some simple examples of monitoring tools such as the “TrueSign Check Status” button, which will display the current status of the Envelope and the current signer of the document. Or take a look at the “TrueSign Delete Envelope” task, which can be used to permanently delete a TrueSign envelope if you need to cancel a signature request. Take a look at how these tools work in Studio, and try making your own TrueSign Envelope monitoring tools directly from OnBase!

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:

Keywords and Document Types

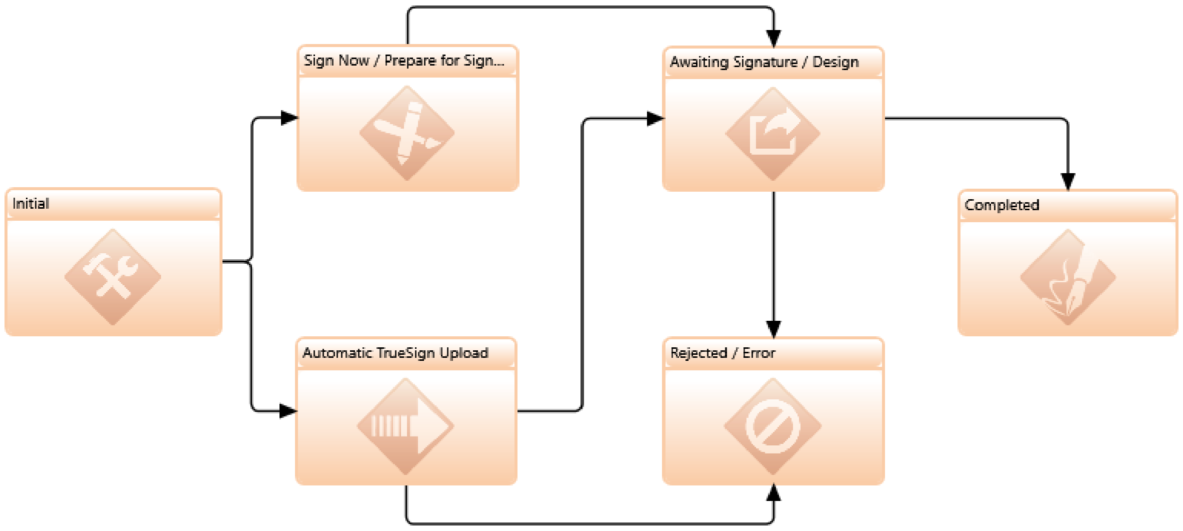
A **document type group** named "TrueSign Next" with the following document types:

- **TrueSign Completed Envelope** – This document type holds the completed envelope's JSON and is created by the Service Bus Listener. This document type gets processed by the "SYS – TrueSign Service Bus Processing" life cycle.
- **TrueSign Test Document** – This document type is for testing purposes only. It contains the following keywords:
    - **Signed:** This keyword is used to set a *Y* or *N* value indicating if the document was signed by the required signer. The value can be changed by modifying the TrueSign Library.
    - **Stamped:** This keyword is used to set a *Y* or *N* value indicating if the document was stamped by the required signer. The value can be changed by modifying the TrueSign 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, checking the status of an envelope, deleting an envelope, 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 Service Bus Listener by TrueSign. This keyword is helpful troubleshooting any issues with non-interactive signing.
    - **TrueSign Signer Email:** This keyword is used as an example for how to perform an automated signing process; you can see an example for how you could implement this in the "Automatic TrueSign Upload" queue of the Sample life cycle. The intended use is that this keyword will contain the email address of the person who needs to sign the document.
        - Also included are the "**TrueSign Signer First Name**" and "**TrueSign Signer Last Name**". All 3 of these keywords are part of a multi-instance keyword group called "TrueSign Signer Information".

Note: TrueSign currently only supports signing PDF documents. If the document you are uploading is not a PDF document, the library will attempt to convert the document to a PDF, however this is only compatible with certain file formats such as Image or Microsoft Word documents.

Life Cycle: TrueSign Sample (v9) – Not For Production

The export package contains an example **life cycle** named “TrueSign Sample (v9) – Not for Production Use”. Whenever you upload a new “TrueSign Test Document” into the system, it will arrive automatically in this life cycle for testing. This life cycle is intended for demonstration and learning purposes only; if you’d like to use any of the features in this life cycle in a production setting, you should manually copy any timers / ad hoc tasks / task lists into your own separate workflow so you can make the appropriate modifications.


Initial
Sign Now / Prepare for Sign...
Automatic TrueSign Upload
Awaiting Signature / Design
Rejected / Error
Completed

The life cycle contains the following queues:

- **Initial** – Initial queue that TrueSign Test Documents arrive. From here, documents can either be sent to the "Sign Now / Prepare for Signature" queue for manual processing, or they can be sent to the Automatic TrueSign Upload queue for automated processing.

  This queue contains the following ad hoc tasks:

  - **Send to “Sign Now / Prepare for Signature”:** Transition this document to the Interactive Signing queue. From this queue, you can either sign the document yourself or prepare the document to be signed by another user.

  - **Send to “Automatic TrueSign Upload”:** Transition the document to the "Automatic TrueSign Upload" queue. This queue will automatically upload the document to a new TrueSign envelope and send it to the email address stored in the "Required Signer" keyword.

- **Sign Now / Prepare for Signature** – From this queue you can either sign the document yourself (using the TrueSign Now task), or prepare and send a document to be signed by another person (using either the TrueSign Design or Send for External Signature tasks)

  - **TrueSign Now**: Sign selected document(s) right now. A window will open with TrueSign Iand once the envelope is completed and the window is closed, the signed document will be imported into OnBase as a new revision. This is one of the most common and accessible ways to integrate TrueSign into your environment. This task does not require using the Service Bus Listener, because the download process is programmed directly into the ad hoc task itself.

The only update(s) that are explicitly needed to make this ad hoc task work is that the TrueSignClientID and TrueSignSecret properties must be updated to the correct values for your envelope type. You can find these credentials via your admin portal here:

https://app.truesign.com/Admin/EnvelopeTypes

For this task to work, the OnBase user who is executing it must have a working TrueSign profile within your Organization's account. The OnBase user's email address associated with their account must match the email on their TrueSign account.

- **TrueSign Design**: Upload the selected document(s) to TrueSign and continue designing the envelope on TrueSign.com. The TrueSign Designer allows you to add anchors, edit documents and signers. After designing, the OnBase document routes to an Awaiting Signature queue.
  If a user selects multiple documents at once using CTRL or SHIFT in workflow, then all of the selected documents will upload to a single envelope together. This is especially useful if documents need to be signed as a group.

  There is a task list in this ad hoc task that MUST be manually modified for your solution titled: "Route Docs IMPORTANT: READ TASK LIST DOCUMENTATION"

  Please find this task list, review the recommendations within it, and create the appropriate updates so that an "Awaiting Signature" queue can be appropriately used within your integration. It is not recommended to utilize the TrueSign Design task if you do not have a properly engineered "Awaiting Signature" queue!

- **Send for Signature**: Send the selected document(s) externally to be signed via TrueSign.com. Users can select either internal or external users to send the document to via a screen. Up to 10 signers can be added.
  Note: external signer module needed in the TrueSign subscription.
  You can verify your current TrueSign licensing for external signatures via the Admin portal here:

  https://app.truesign.com/Admin/Account/Info

  After the documents have been uploaded to an envelope, the envelope can be sent off to the signer(s) for signature. After sending documents via this task, the document will be routed to the Awaiting Signature queue where it will sit until a "Completed Envelope Document" message is received from the Service Bus Listener

- **Automatic TrueSign Upload -** This queue provides a simple example for how to automatically upload a document to TrueSign. If you decide to use this example as the basis for a production implementation, please take care to update the "TrueSignFirstName" and "TrueSignLastName" properties to the signer's first/last name respectively. It is recommended that you store these values in keywords so that it can work more dynamically.

  This queue also provides some examples for how you can automatically pre-anchor a document using OnBase Notes. There is some more information you can read about in the documentation tab of the task list called "OPTIONAL: Add Anchors to Document"

- **Awaiting Signature / Design** – Documents in this queue are currently pending signature from TrueSign.com. This queue serves as an example for what an "Awaiting Signature" queue could look like in your solution. The idea with this queue is that any document that is currently pending signature on TrueSign.com should be sent to a separate queue so that the OnBase copy of the document is not interfered with during the time we are waiting for signature.

  Once an envelope has been completed, a message will be sent to the Service Bus Listener, which will be processed via the "SYS - TrueSign Service Bus Processing" life cycle. There are rules and actions within this life cycle that transition the envelope documents out of this queue and into the "Completed" queue.

  When creating your own "Awaiting Signature" life cycle in your solution, make sure you update the "SYS - TrueSign Service Bus Processing" life cycle accordingly so that your completed documents transition to the next queue after completion.

  The following ad hoc tasks are available in this queue:

  - **Check TrueSign Envelope Status -** This task is for testing purposes only, and to demonstrate the "TrueSign Check Status" script. This script can be used as a building block to create your own custom envelope monitoring solutions.

  - **Display Envelope Details -** This task will display the "Envelope Details" screen on TrueSign.com. This task is intended for troubleshooting / administrative purposes only. The envelope details screen contains a lot of useful troubleshooting information such as the current signer, the status of the envelope, the history of the envelope, etc. You can also download the JSON data of the envelope. This JSON data is the same information that gets sent to OnBase via the Service Bus Listener, so this can be a useful tool when troubleshooting issues with the integration features.

  - **View TrueSign History** - This ad hoc task will retrieve the history for this document found in the last envelope in TrueSign.

  - **Delete TrueSign Envelope** - This task deletes the current TrueSign Envelope. This action cannot be undone. The OnBase Document will not be deleted; only the copy on TrueSign.com contained within the current envelope. This task likely needs modifications to be useful in a production setting.

- **Rejected / Error -** Documents are sent to this queue if an error is encountered during the TrueSign process OR if one of the signers rejected signing the envelope. In this sample life cycle, there are two contexts in which a document could arrive in this queue:

  1) If, during the "Automatic TrueSign Upload", there is an issue where the document could not be successfully uploaded to TrueSign.com, the document will be routed to this queue.

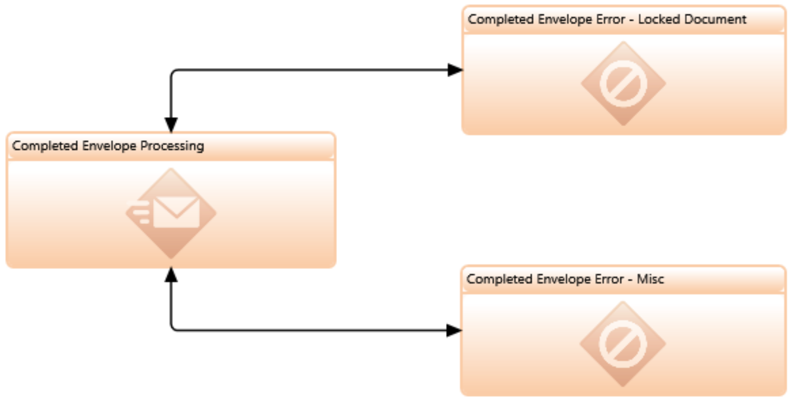
  2) If an envelope is rejected by a signer, then the document will be routed to this queue via logic that is programmed in the "SYS - TrueSign Service Bus Processing" life cycle.

  It is recommended that you create your own rejected or error queue in your TrueSign implementation; this queue exists only as an example.

- **Completed** – Documents are sent here from the "Awaiting Signature / Design" queue after they have been successfully signed. The logic for how documents get routed to this queue can be seen in the "SYS - TrueSign Service Bus Processing" life cycle.

Life Cycle: SYS – TrueSign Service Bus Processing

The export package contains a life cycle named **“SYS – TrueSign Service Bus Processing”.** Unlike the TrueSign Sample life cycle, this life cycle *can* be used for production if some minor modifications are made to it. This life cycle processes TrueSign Completed Envelope documents that are returned back from the Service Bus Listener. These documents contain JSON data which is used to download envelopes from TrueSign.com and add the signed revision of the document to OnBase. Unlike many of the other components of the sample lifecycle, you should feel free to use this life cycle for your live production system, however be aware that modifications will need to be made to ensure documents get routed correctly though your custom workflows!


Completed Envelope Error - Locked Document
Completed Envelope Processing
Completed Envelope Error - Misc

The following queues are configured in this life cycle:

- **Completed Envelope Processing** – This timer processes TrueSign Completed Envelope documents that are returned back from the Service Bus Listener. These documents contain JSON data which is used to download envelopes from TrueSign.com and add the signed revision of the document to OnBase. This Life Cycle is only used when the Delivery Method for an envelope is setup to use the Service Bus Listener. For example, this timer is not necessary when using the "TrueSign Now" task, because TrueSign Now retrieves the completed envelope data directly from the TrueSign API; there is no need to involve a separate "Completed Envelope" document in this context.

  However, when sending a document to external parties, it may be a few hours, a few days, or even a few weeks before a document receives a TrueSign signature. In this case, we need to use the Service Bus Listener to send a message to OnBase that the envelope has been signed.

  The way this data is sent is by importing a new document into OnBase called "TrueSign Completed Envelope". The contents of this document is the raw JSON data of the envelope on TrueSign.com. You can view the same JSON data via the admin portal on TrueSign.com when navigating to an envelope.

  This timer reads the JSON data stored within the Completed Envelope. This JSON data stores a lot of information; most importantly, it has the Document Handle of the document that needs to be updated, a download URL so OnBase can receive the signed rendition of the document, it contains data on whether the document was signed, stamped, or rejected, etc. All of this information is read in the "TrueSign Download" script.

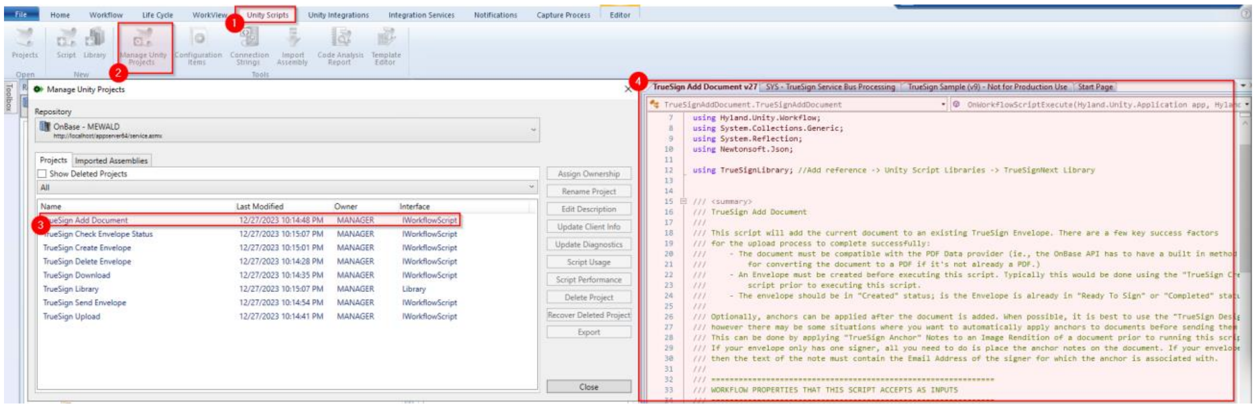
IMPORTANT: This process is configured in a timer, not system work. This is done deliberately to avoid complications with the Service Bus Listener repeatedly sending documents in the event that there are errors in the System Work. DO NOT ADD TRUESIGN COMPLETED ENVELOPE DOCUMENTS INTO ANY LIFE CYCLES WITH SYSTEM WORK CONFIGURED AS IT CAN LEAD TO ISSUES WITH THE SERVICE BUS LISTENER

- **Completed Envelope Error – Locked Document** - If an envelope document is currently locked when the Completed Envelope JSON message is returned, it will be sent to this queue for re-processing. When a completed JSON message is received, all documents contained within that envelope must be updated with a new revision. This new revision cannot be applied if the document is currently locked in OnBase. Before the TrueSign Download script executes, it runs a check to see if any of the envelope documents are currently locked, and if so, stops processing. There is a timer in this queue which will attempt to re-process the document after 2 hours, with the assumption that after 2 hours have passed, the lock will have been removed.

- **Completed Envelope Error – Misc -** If an error is encountered during the Completed Envelope Processing, the TrueSign Completed Envelope will be sent here for further investigation. After reviewing the error, administrators can execute the "Re-Process Completed Envelope" ad hoc task to send the document back to the "Completed Envelope Processing" queue.

Unity Scripts

The export package contains a **Unity Library** and several **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. This documentation contains only brief descriptions of what the scripts do, however if you open the scripts up you will find comprehensive documentation included within each of them:



OnBase - MEWALD
http://localhost/appserver64/service.asmx

Name	Last Modified	Owner	Interface
TrueSign Add Document	12/27/2023 10:14:48 PM	MANAGER	IWorkflowScript
TrueSign Check Envelope Status	12/27/2023 10:15:07 PM	MANAGER	IWorkflowScript
TrueSign Create Envelope	12/27/2023 10:15:01 PM	MANAGER	IWorkflowScript
TrueSign Delete Envelope	12/27/2023 10:14:28 PM	MANAGER	IWorkflowScript
TrueSign Download	12/27/2023 10:14:35 PM	MANAGER	IWorkflowScript
TrueSign Library	12/27/2023 10:15:07 PM	MANAGER	Library
TrueSign Send Envelope	12/27/2023 10:14:54 PM	MANAGER	IWorkflowScript
TrueSign Upload	12/27/2023 10:14:41 PM	MANAGER	IWorkflowScript

```
TrueSignAddDocument.TrueSignAddDocument    OnWorkflowScriptExecute(Hyland.Unity.Application app, Hyla
using Hyland.Unity.Workflow;
using System.Collections.Generic;
using System.Reflection;
using Newtonsoft.Json;

using TrueSignLibrary; //Add reference -> Unity Script Libraries -> TrueSignNext Library


/// <summary>
/// TrueSign Add Document
///
/// This script will add the current document to an existing TrueSign Envelope. There are a few key success factors
/// for the upload process to complete successfully:
///     - The document must be compatible with the PDF Data provider (ie., the OnBase API has to have a built in method
///         for converting the document to a PDF if it's not already a PDF.)
///     - An Envelope must be created before executing this script. Typically this would be done using the "TrueSign Cr
///         script prior to executing this script.
///     - The envelope should be in "Created" status; is the Envelope is already in "Ready To Sign" or "Completed" stat
///
/// Optionally, anchors can be applied after the document is added. When possible, it is best to use the "TrueSign Desi
/// however there may be some situations where you want to automatically apply anchors to documents before sending the
/// This can be done by applying "TrueSign Anchor" Notes to an Image Rendition of a document prior to running this scri
/// If your envelope only has one signer, all you need to do is place the anchor notes on the document. If your envelo
/// then the text of the note must contain the Email Address of the signer for which the anchor is associated with.
///
/// ==============================================================
/// WORKFLOW PROPERTIES THAT THIS SCRIPT ACCEPTS AS INPUTS
```

- **TrueSign Library** – a Unity script library that contains core logic to connect and call different TrueSign API methods. All of the Unity Scripts mentioned below leverage this library.

- **TrueSign Create Envelope -** This script creates a new empty TrueSign envelope. The end result of this script is that a TrueSign envelope will be initialized in the "Created" status. The envelope should have signer(s) associated to it.

- **TrueSign Add Document -** This script will add the current document to an existing TrueSign Envelope. There are a few key success factors for the upload process to complete successfully:

    - The document must be compatible with the PDF Data provider (ie., the OnBase API has to have a built in method for converting the document to a PDF if it's not already a PDF.)
    - An Envelope must be created before executing this script. Typically this would be done using the "TrueSign Create Envelope" script prior to executing this script.
    - The envelope should be in "Created" status; if the Envelope is already in "Ready To Sign" or "Completed" status, the upload will fail

- **TrueSign Send Envelope -** This script sends a TrueSign envelope to the signers for signature. This is typically the 'last step' of preparing an envelope for signature. Pre-requisites for running this script are creating a new TrueSign envelope via the "TrueSign Create Envelope" script, and then adding at least one document to the envelope using the "TrueSign Add Document" script.
- **TrueSign 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.
- **TrueSign Check Envelope Status -** Provide an envelope ID via a property to check the status of the associated envelope. The status will be stored in the "TrueSignStatus" property. Possible Statuses that can be returned back are: Created, ReadyToSign, Signed, ReadyToNotify, Completed, Deleted, Rejected
    - This script also detects the current signer associated to the envelope. The current signer's email address is stored in the "TrueSignCurrentSigner" property
    - This script also stores the first Designer associated with the envelope in the property "TrueSignCurrentDesigner"
- **TrueSign Delete Envelope -** This script will permanently delete a TrueSign Envelope, as well as all of the documents contained within that envelope. The GUID of the envelope you wish to delete must be provided in the workflow property TrueSignEnvelopeId.
    - The action of deleting a TrueSign Envelope cannot be undone, so use this script carefully. This script will delete the envelope regardless of what status the envelope is in, so you may want to use this script in conjunction with the "TrueSign Check Status" script to verify that an envelope is in the correct state before deleting it permanently.
    - Please also note that deleting the envelope on TrueSign.com will not delete the copies of the documents in OnBase.

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

HTML Forms

The export package contains a collection of 6 **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 Launch Viewer** – 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 New Envelope – Single -** 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 New Envelope – Multiple** – 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 View 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.
- **TrueSign Display Envelope Details** – This form launches the Envelope Details page, which is intended to be used by System Administrators, developers, and power users. This screen contains useful troubleshooting information.
- **TrueSign Launch Designer** – This form launches the TrueSign Designer interface, wherein users can prepare a document for signature by placing signature anchors on the document and configuring which people need to sign the 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.

Note Types & Anchors

There are a few OnBase Note types that are also included in the export file:

- **TrueSign Error –** This note is added to the "TrueSign Completed Envelope" document if it encounters an error during processing.
- **TrueSign Rejected –** This note is added to a document if one of the Signers rejected the signature. The reason for rejection is included in the body text of the note.

Additionally, some of the notes included are intended to be used as Anchors (anchors are placeholders that indicate where a signer should place their signature or markup on the document). Whenever possible, it is best to place anchors using the TrueSign Design ad hoc task. This interface is the most compatible and user-friendly way of anchoring a document.

However, there could be situations where you want to automatically anchor a document in OnBase, such as a document that you composed using the Document Composition module. In this scenario, it is possible to pre-anchor a document using OnBase Notes before uploading the document to TrueSign. When any of the following note types are positioned on an Image File Format document, they will be converted to TrueSign anchors of their respective type:

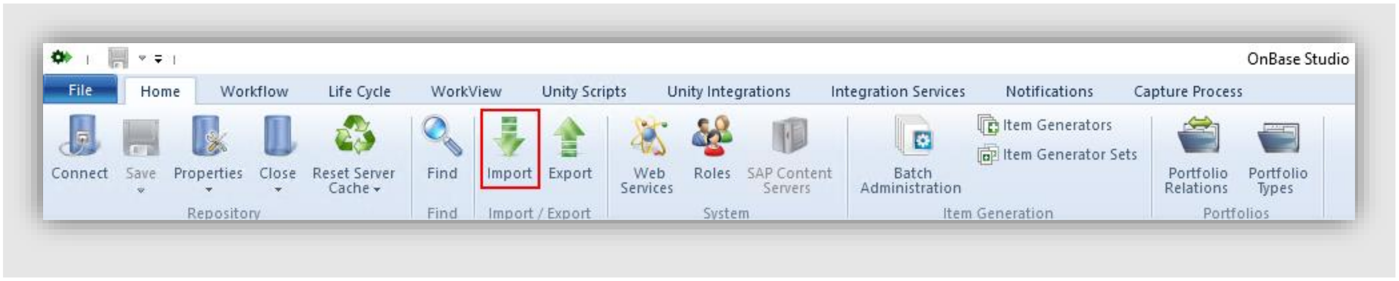
- **TrueSign Anchor (Sign Here)**
- **TrueSign Anchor (Initial Here)**
- **TrueSign Anchor (Date Here)**
- **TrueSign Anchor (Checkmark Here)**
- **TrueSign Anchor (Textbox Here)**

There are a couple of shortcomings with the OnBase Note Types & Anchors that you should be aware of:

- There is currently no support in this sample lifecycle for ‘dynamically’ placing anchors in the correct position; you can only hard-code anchors to be positioned in a pre-determined X / Y coordinate and pre-determined width / height. For this reason, any forms that you are trying to auto-anchor need to be built so that the signature positions to not nudge or adjust. The simplest way to accomplish this is to always put your signature page as a separate final page of your document (ie., using CTRL+ENTER in word to add a page break).
- If your image document uses a “non-standard” DPI or resolution, the notes may not position correctly when being converted form OnBase Notes to TrueSign Anchors. It is recommended that any document you are adding anchors to automatically is a 300 DPI TIFF document. If you need the anchors to work with other image sizes, you can try adjusting the property TrueSignDPIScale (see properties section below for more information). There is currently no support for dynamically detecting the DPI / resolution of an image you are uploading for signature.

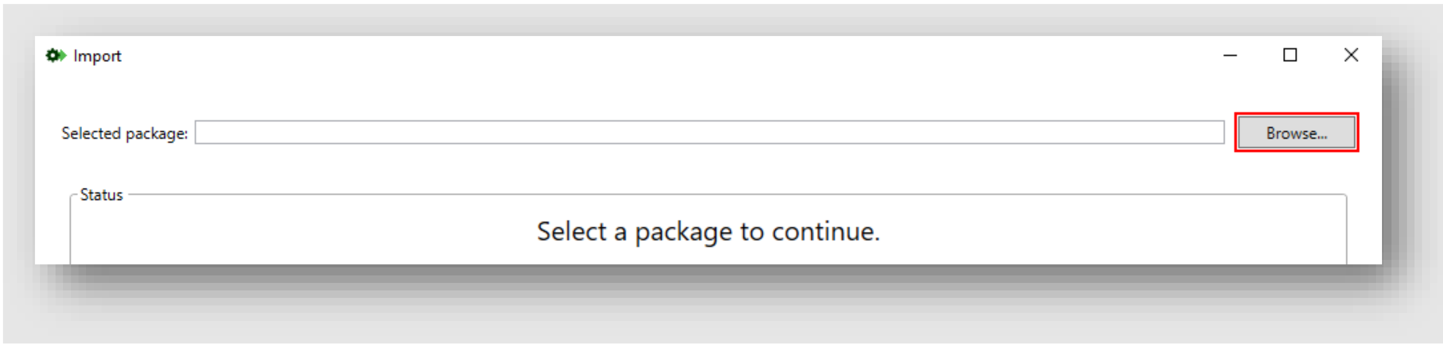
Starting the Import

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 i3 Verticals.

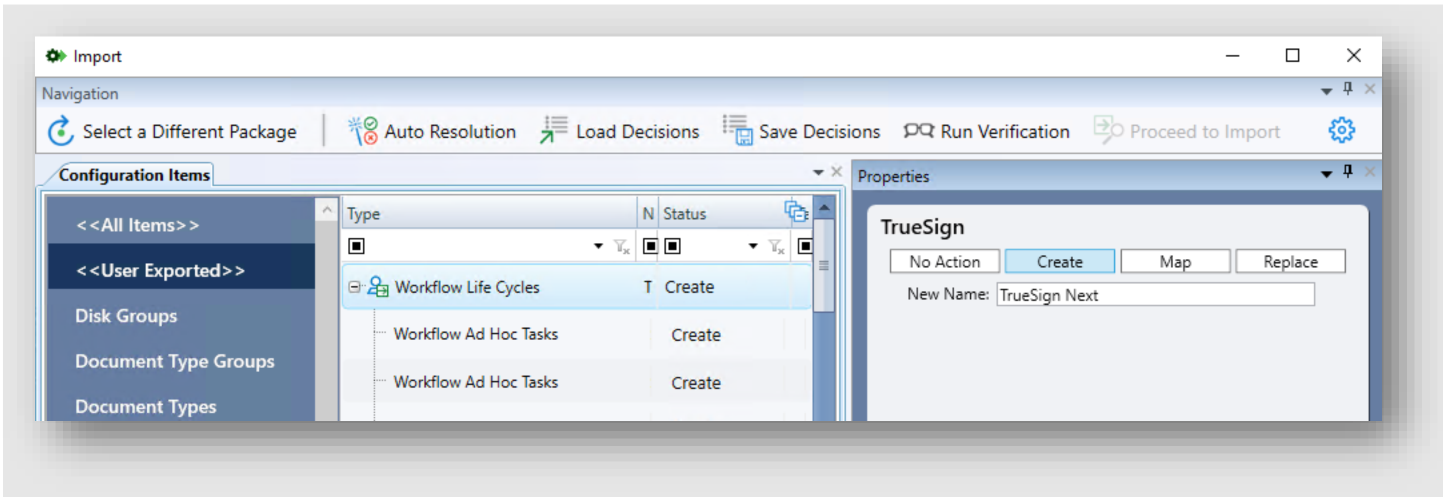
Note: The provided export might be from a different OnBase version than the one you have. Please reference the OnBase Studio module reference guide to resolve this conflict.



Select a package to continue.

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: The package 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.


Import

Navigation

Select a Different Package | Auto Resolution | Load Decisions | Save Decisions | Run Verification | Proceed to Import

Configuration Items

<<All Items>>
<<User Exported>>
Disk Groups
Document Type Groups
Document Types

Type	N	Status
Workflow Life Cycles	T	Create
Workflow Ad Hoc Tasks		Create
Workflow Ad Hoc Tasks		Create

Properties

TrueSign

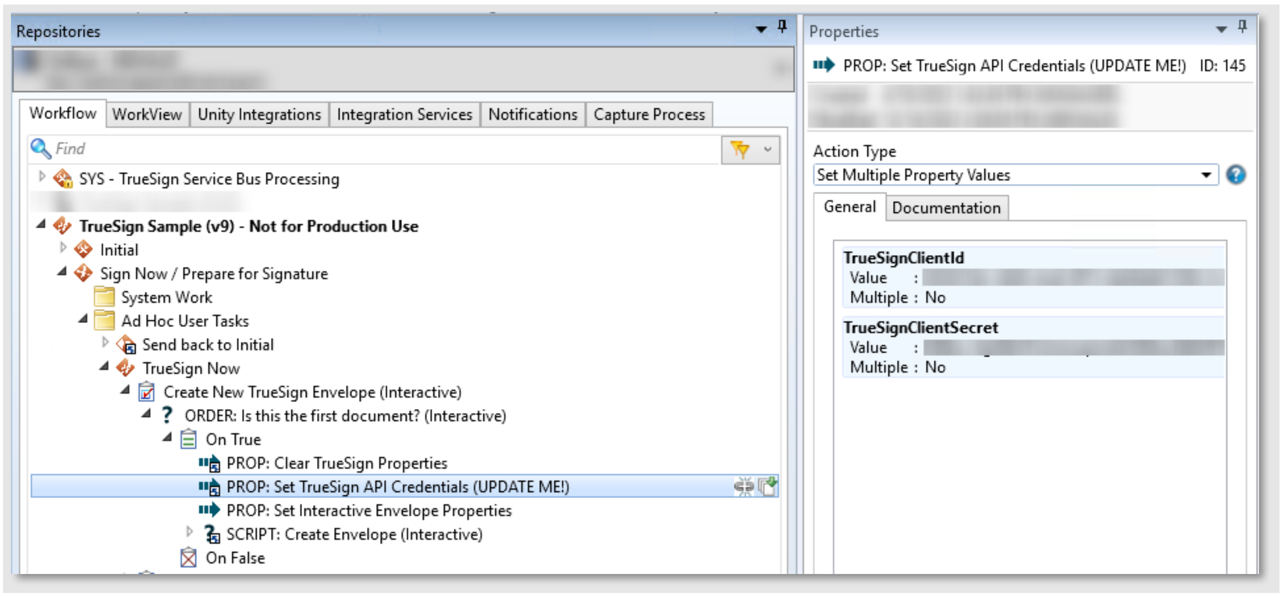
No Action | Create | Map | Replace

New Name: TrueSign Next

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.


Repositories

[redacted]

Workflow | WorkView | Unity Integrations | Integration Services | Notifications | Capture Process

Find

- SYS - TrueSign Service Bus Processing
- [redacted]
- TrueSign Sample (v9) - Not for Production Use
  - Initial
  - Sign Now / Prepare for Signature
    - System Work
    - Ad Hoc User Tasks
      - Send back to Initial
      - TrueSign Now
        - Create New TrueSign Envelope (Interactive)
          - ORDER: Is this the first document? (Interactive)
            - On True
              - PROP: Clear TrueSign Properties
              - PROP: Set TrueSign API Credentials (UPDATE ME!)
              - PROP: Set Interactive Envelope Properties
              - SCRIPT: Create Envelope (Interactive)
            - On False

Properties

PROP: Set TrueSign API Credentials (UPDATE ME!) ID: 145

[redacted]

Action Type: Set Multiple Property Values

General | Documentation

TrueSignClientId
Value : [redacted]
Multiple : No

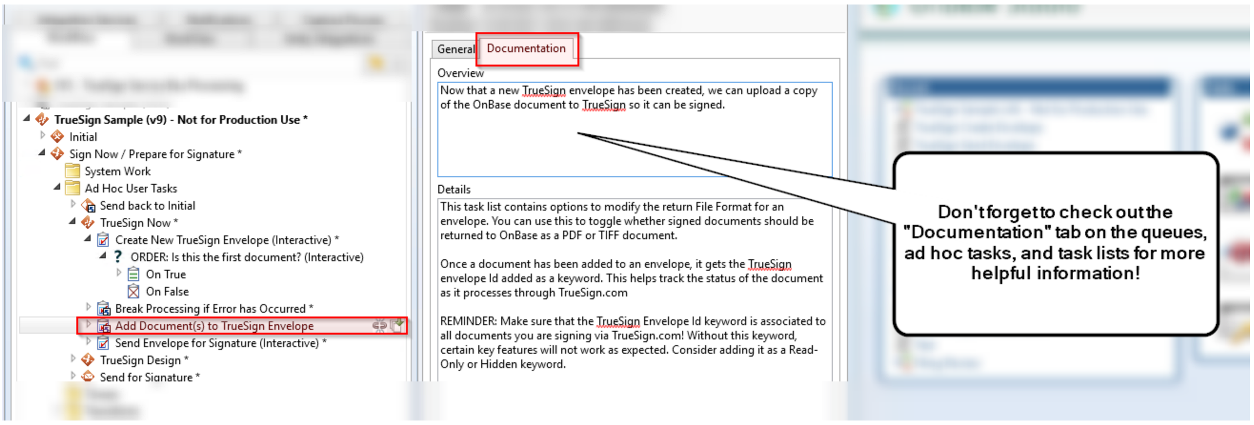
TrueSignClientSecret
Value : [redacted]
Multiple : No

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 "Sign Now / Prepare for

Signature" queue since the Service Bus Listener has not been configured yet.

Throughout the life cycle, there are various actions or task lists that are labeled "UPDATE ME!" or "READ TASK LIST DOCUMENTATION!"; these are here to help guide your attention to the areas of the life cycle that will need to be tweaked to your unique needs. Whenever you see these, always review them carefully and check out the Documentation Tab for helpful tips on what actions you should take!


TrueSign Sample (v9) - Not for Production Use *
Initial
Sign Now / Prepare for Signature *
System Work
Ad Hoc User Tasks
Send back to Initial
TrueSign Now *
Create New TrueSign Envelope (Interactive) *
ORDER: Is this the first document? (Interactive)
On True
On False
Break Processing if Error has Occurred *
Add Document(s) to TrueSign Envelope
Send Envelope for Signature (Interactive) *
TrueSign Design *
Send for Signature *
General
Documentation
Overview
Details
Don't forget to check out the "Documentation" tab on the queues, ad hoc tasks, and task lists for more helpful information!

Now that a new TrueSign envelope has been created, we can upload a copy of the OnBase document to TrueSign so it can be signed.

This task list contains options to modify the return File Format for an envelope. You can use this to toggle whether signed documents should be returned to OnBase as a PDF or TIFF document.

Once a document has been added to an envelope, it gets the TrueSign envelope Id added as a keyword. This helps track the status of the document as it processes through TrueSign.com

REMINDER: Make sure that the TrueSign Envelope Id keyword is associated to all documents you are signing via TrueSign.com! Without this keyword, certain key features will not work as expected. Consider adding it as a Read-Only or Hidden keyword.

Upgrading from Prior Version (Version 8 > Version 9)

If you have implemented a TrueSign Integration between 2020-2024, it is possible that you are using a previous version of the TrueSign Sample Life Cycle (version 8). In this scenario, it is recommended that you upgrade and replace the v8 components with the new v9 components described in this guide. The TrueSign Integration for OnBase v8 guide is still available for your reference online here:

TrueSign Integration for OnBase (VERSION 8, MARCH 2021)

It is possible that modifications have been made to the v8 scripts and workflows, so we cannot provide a comprehensive step-by-step upgrade path for transition, however we can provide some best practices and considerations for your upgrade process.

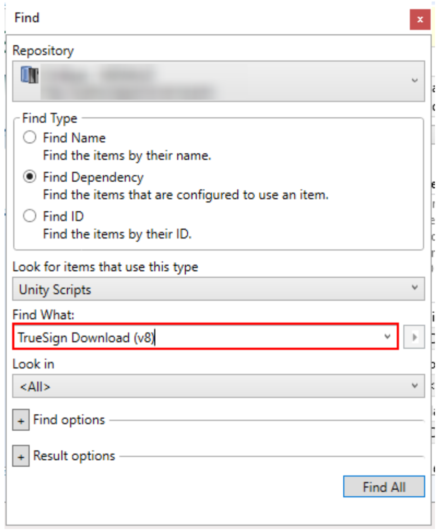
Prep Tasks

- Before importing the new v9 integration into your system, we recommend to re-name all your existing Unity Scripts to append "v8" to the end of them, like so:
  - Rename "TrueSign Upload" script to "TrueSign Upload (v8)"
  - Rename "TrueSign Download" to "TrueSign Download (v8)"
  - Rename "TrueSign Library" to "TrueSign Library (v8)"
- Before importing the new v9 integration into your system, re-name the old Sample Lifecycle to add "v8" to the end of the name (ie., TrueSign Sample (v8))
  - You may even consider fully disabling or deleting the old v8 Sample Lifecycle if you are sure it is not being used for any production purposes (it is never recommended to use the sample life cycles in a production setting)

- Remove the Life-cycle associations on the “TrueSign Test Document” and “TrueSign Completed Envelope” document types. We do not want these document types to enter both the new AND the old life-cycles

Import Process

- In general, when performing the Studio Import, you should be very cautious about using the “REPLACE” option. For most of the import decisions, you should be using the “CREATE” option.
    - One exception to this rule is that you should be safe to use the “REPLACE” option on the Document Types and Keywords included in the export package
- Most importantly, make sure that all of the Unity Scripts, Unity Library, and Workflows are being imported as new items. You should not replace the v9 Unity Scripts with the pre-existing v8 scripts. You should not overwrite the v8 life cycle with the v9 life cycle. The end result after importing the v9 system is that you should have a set of v8 TrueSign Unity Scripts, and a set of v9 TrueSign Unity Scripts
- After you have successfully imported the v9 system into OnBase, you should perform a full audit of all existing TrueSign integrations in your system. Because the v8 and v9 Unity Scripts can co-exist in the same OnBase solution, you can slowly replace the old v8 workflow logic with the v9 logic and test things at your own pace.
    - You can use the CTRL+F function in OnBase Studio to perform a “Dependency Search”, which can help you locate where any v8 scripts were being used. This will help you find the areas in workflow you will want to replace.



Find

Repository

Find Type
- Find Name — Find the items by their name.
- [✓ Find Dependency] — Find the items that are configured to use an item.
- Find ID — Find the items by their ID.

Look for items that use this type: Unity Scripts

Find What: TrueSign Download (v8)

Look in: <All>

    - For example, if you are using the old version 8 copy of the “TrueSign Now” ad hoc task, we recommend that you remove that task and replace it with the version of “TrueSign Now” that is included in the new version 9 of the sample life cycle. This version of “TrueSign Now” has some extra error handling and should work more reliably than the previous version of this ad hoc task.
    - Make sure you understand if any modifications have been made to your pre-existing scripts and workflows before replacing them. It is recommended that you are as comprehensive as possible with your environment review before you start replacing any of the existing workflow logic.

- The v8 and v9 integrations can co-exist in the same OnBase System, but having two versions of these scripts will likely lead to confusion. It is preferable that after your upgrade process has concluded that the v8 Unity Scripts and workflows are fully deleted. Once you have replaced all of your pre-existing v8 TrueSign integrations in workflow, you can start the process of deleting the old v8 Unity Scripts and workflows.

About the TrueSign Launch User Form

While i3 Verticals 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 Launch Viewer” 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 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.

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:

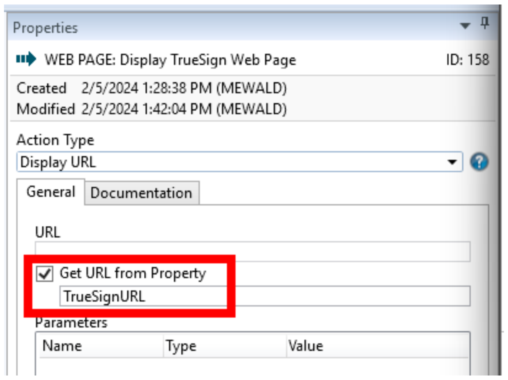
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT\microsoft-edge]
"URL Protocol"=""
@=URL:microsoft-edge

Recommendations for OnBase 23 and Greater

OnBase 23.1 added some options to the “Display URL” workflow action which can be used to replace some of the E-Forms provided in this solution. There are 3 e-forms that have the sole purpose of launching a web page to the user:

- TrueSign Launch Viewer
- TrueSign Launch Designer
- TrueSign Display Envelope Details

To check if you can replace these E-Forms, open OnBase Studio and create a new action with the type “Display URL”. If your OnBase version has an option to “Get URL from Property”, then you can replace these HTML forms with a couple native OnBase actions.

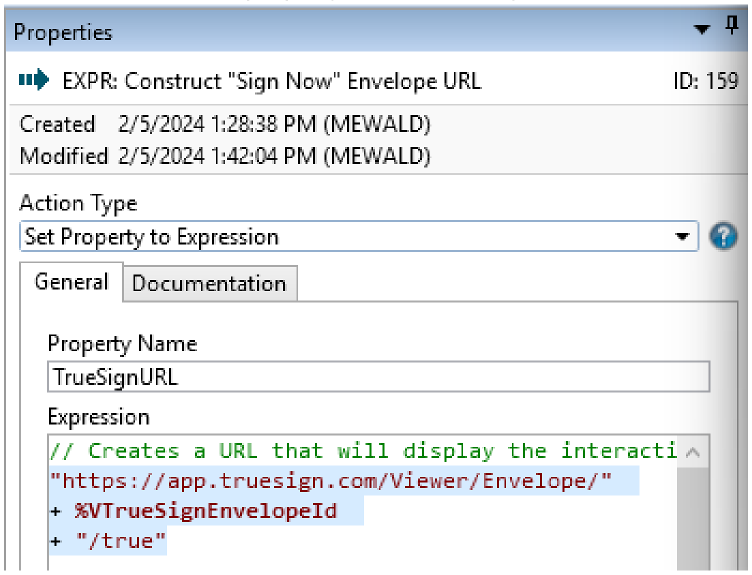

Properties
WEB PAGE: Display TrueSign Web Page ID: 158
Created 2/5/2024 1:28:38 PM (MEWALD)
Modified 2/5/2024 1:42:04 PM (MEWALD)
Action Type: Display URL
General | Documentation
URL
☑ Get URL from Property: TrueSignURL
Parameters

Name	Type	Value

Provided is an example for how to replace the “TrueSign Now” html form to use this action instead:

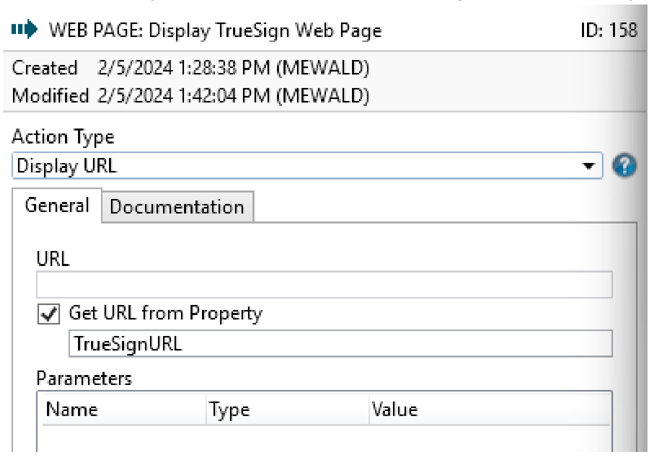
1) Create a new action after the “SCRIPT: Send Envelope” rule that uses the “Set Property to Expression” action type, and use the following expression: "https://app.truesign.com/Viewer/Envelope/" + %VTrueSignEnvelopeId + "/true"

   Store this value in a property called TrueSignURL:


Properties
EXPR: Construct "Sign Now" Envelope URL ID: 159
Created 2/5/2024 1:28:38 PM (MEWALD)
Modified 2/5/2024 1:42:04 PM (MEWALD)
Action Type: Set Property to Expression
General | Documentation
Property Name: TrueSignURL
Expression

```
// Creates a URL that will display the interacti
"https://app.truesign.com/Viewer/Envelope/"
+ %VTrueSignEnvelopeId
+ "/true"
```

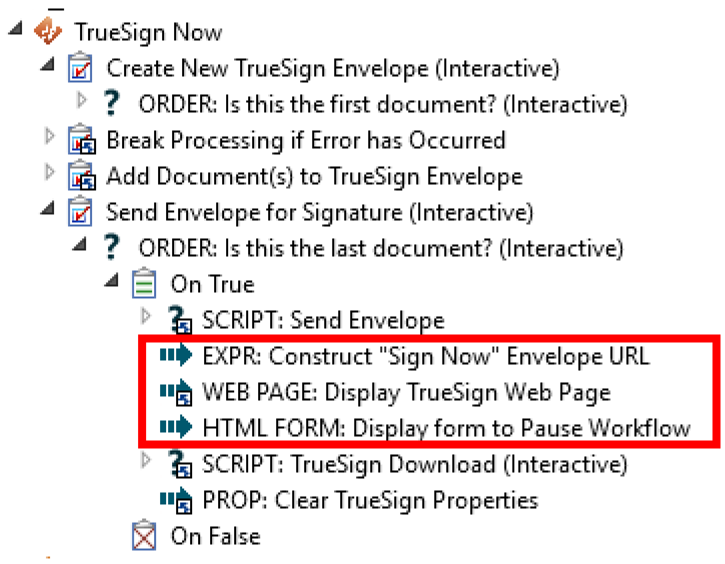
2) Create another action that uses the “Display URL” action type, click the checkbox to “Get URL from Property”, and pass in the TrueSignURL property that we constructed in the previous step:


WEB PAGE: Display TrueSign Web Page ID: 158
Created 2/5/2024 1:28:38 PM (MEWALD)
Modified 2/5/2024 1:42:04 PM (MEWALD)
Action Type: Display URL
General | Documentation
URL
☑ Get URL from Property: TrueSignURL
Parameters

Name	Type	Value

3) Finally, in the case of the TrueSign Now ad hoc task, you will still need a form to display to the user to pause workflow while they are signing the document, so you will still need to add an action after displaying the URL to display a form to the user to click “I’m Done Signing”.

4) Here is an example of what your workflow actions should look like if you’ve set this up correctly:



```
◢ TrueSign Now
    ◢ Create New TrueSign Envelope (Interactive)
        ▷ ORDER: Is this the first document? (Interactive)
    ▷ Break Processing if Error has Occurred
    ▷ Add Document(s) to TrueSign Envelope
    ◢ Send Envelope for Signature (Interactive)
        ◢ ORDER: Is this the last document? (Interactive)
            ◢ On True
                ▷ SCRIPT: Send Envelope
                  EXPR: Construct "Sign Now" Envelope URL
                  WEB PAGE: Display TrueSign Web Page
                  HTML FORM: Display form to Pause Workflow
                ▷ SCRIPT: TrueSign Download (Interactive)
                  PROP: Clear TrueSign Properties
              On False
```

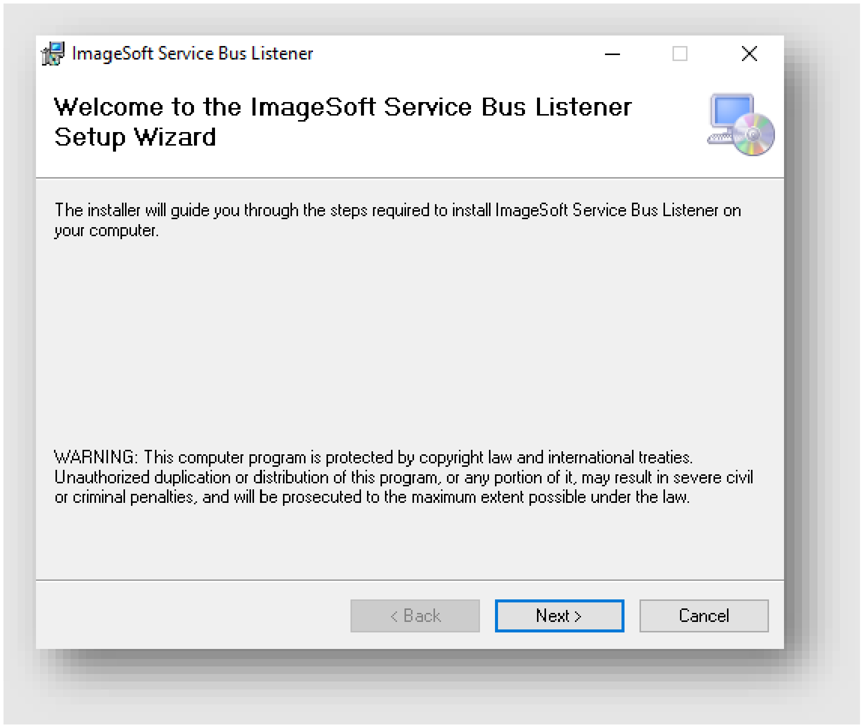
Using the “Display URL” ad hoc task is preferable to using the custom HTML forms because this action will natively open a separate browser window in the user’s default internet browser.

Service Bus Listener

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



Welcome to the ImageSoft Service Bus Listener Setup Wizard

The installer will guide you through the steps required to install ImageSoft Service Bus Listener on your computer.

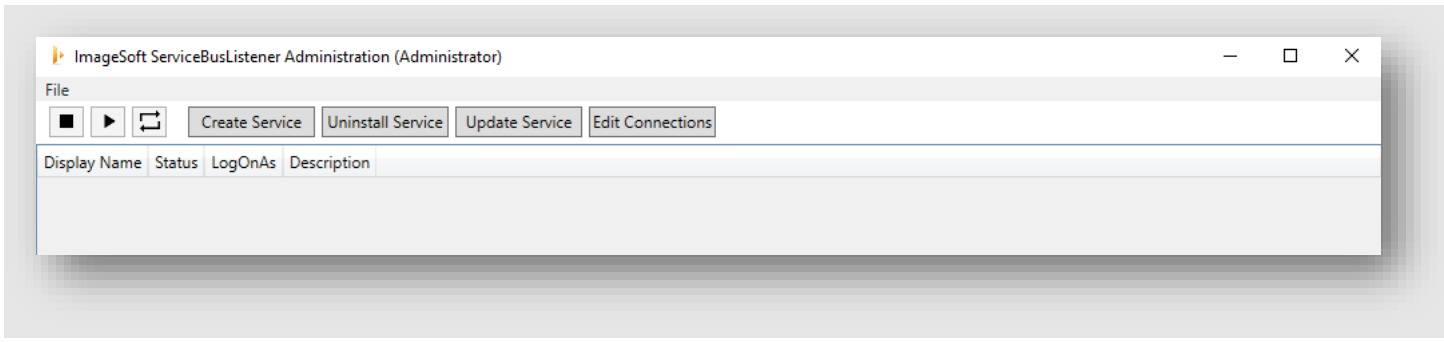
WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

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

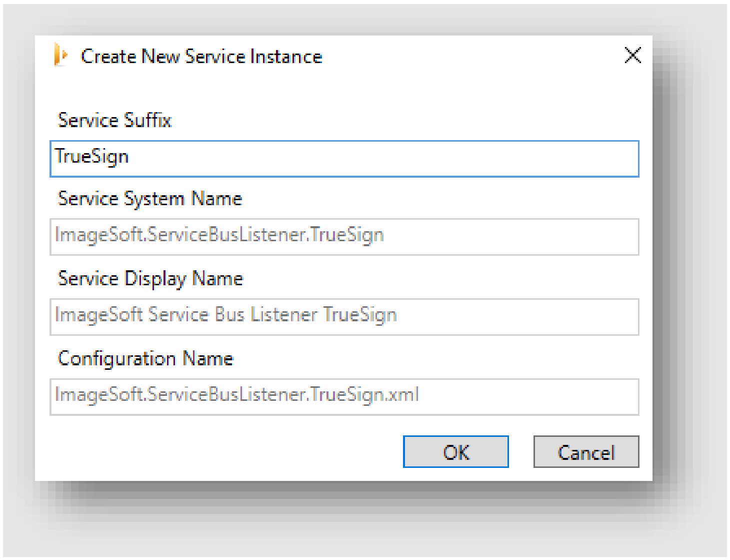
Configuration

Once the 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".


Create New Service Instance

Service Suffix: TrueSign

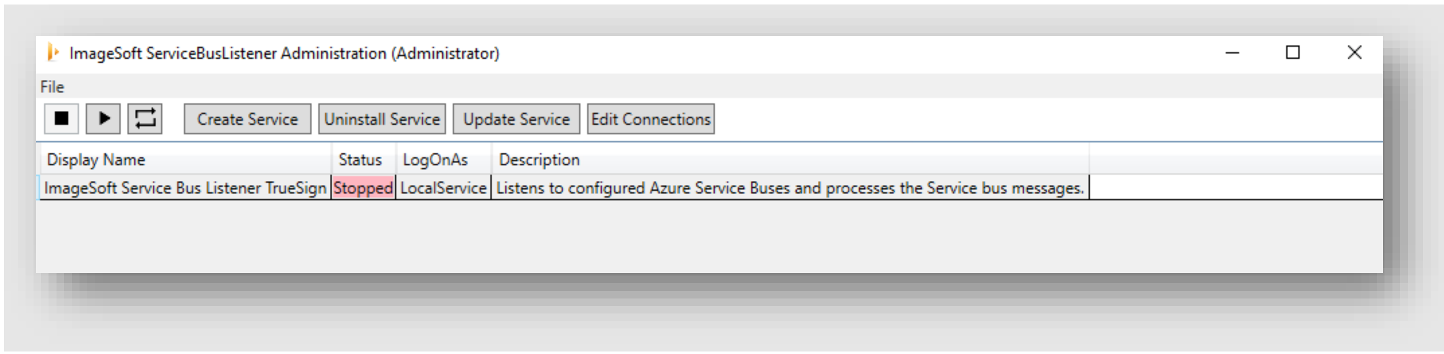
Service System Name: ImageSoft.ServiceBusListener.TrueSign

Service Display Name: ImageSoft Service Bus Listener TrueSign

Configuration Name: ImageSoft.ServiceBusListener.TrueSign.xml

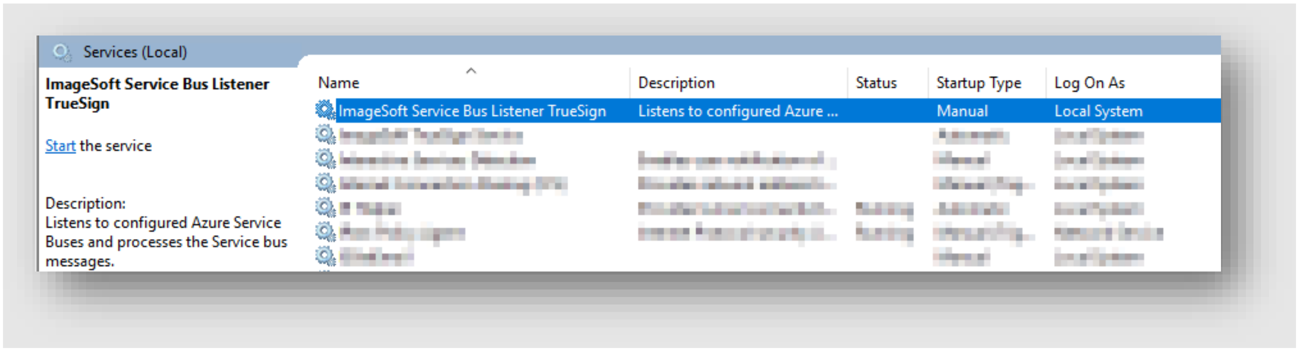
You will see an "Install Complete" message, click OK to continue. You will now be prompted with an informational message reminding you to set any desired service properties via the Window's Services Console.

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



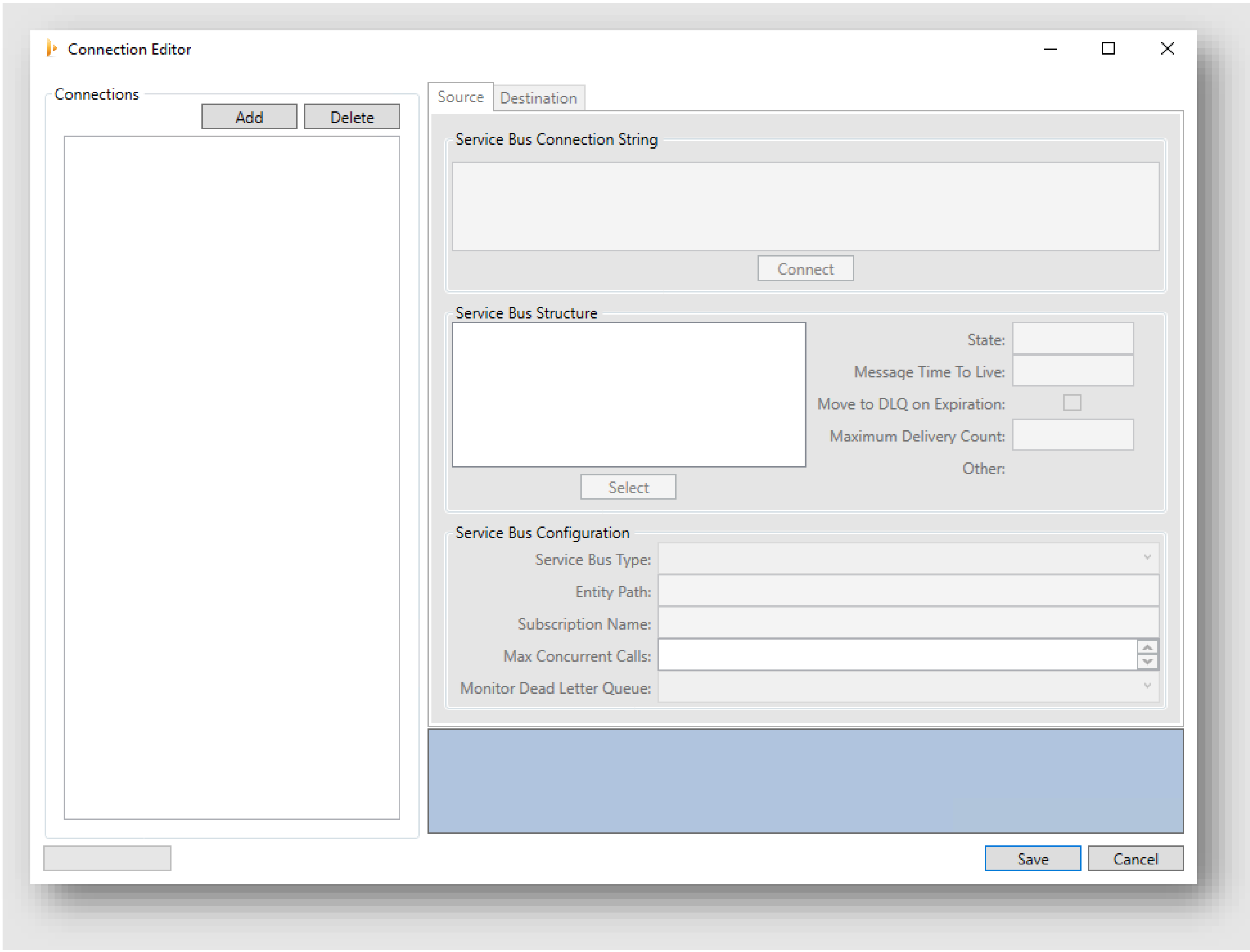
Display Name	Status	LogOnAs	Description
ImageSoft Service Bus Listener TrueSign	Stopped	LocalService	Listens to configured Azure Service Buses and processes the Service bus messages.

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.

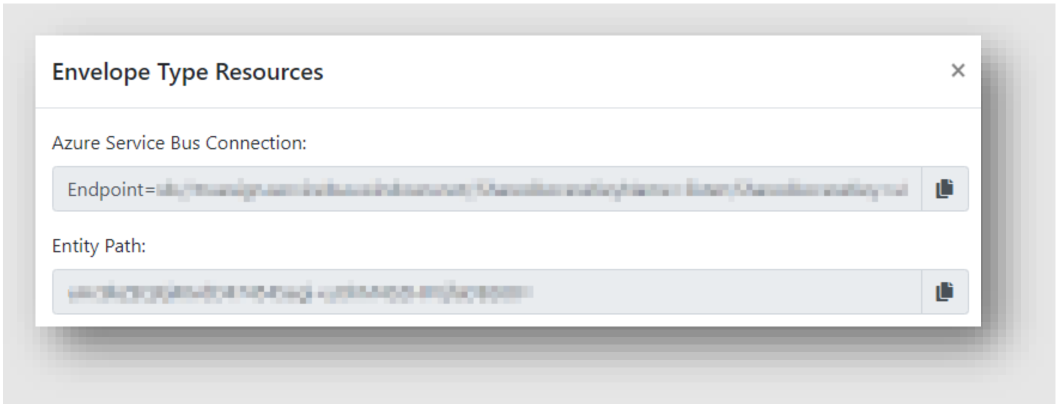

Services (Local)
ImageSoft Service Bus Listener TrueSign
Start the service
Description:
Listens to configured Azure Service Buses and processes the Service bus messages.

Name	Description	Status	Startup Type	Log On As
ImageSoft Service Bus Listener TrueSign	Listens to configured Azure ...		Manual	Local System

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:



Envelope Type Resources

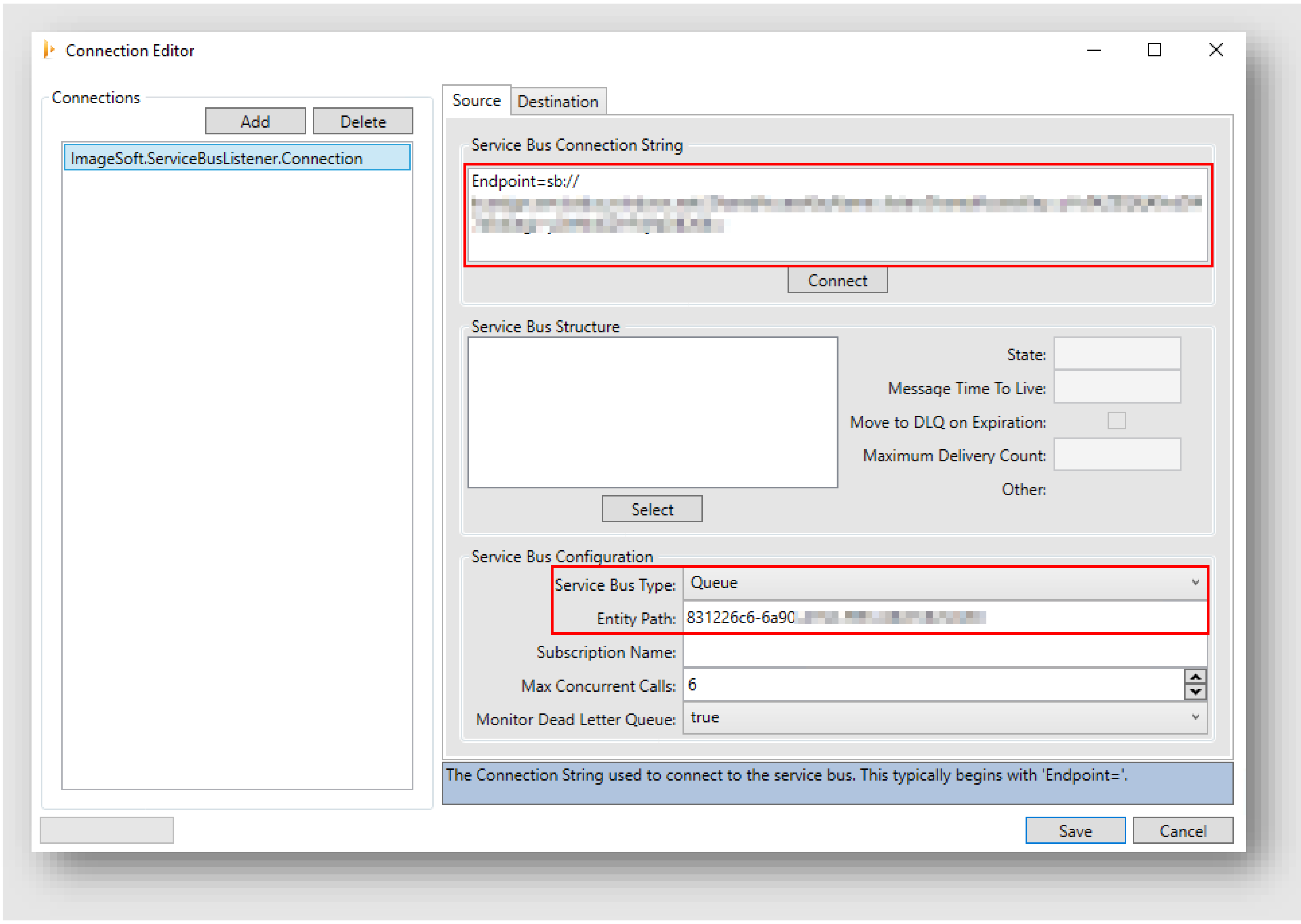
Azure Service Bus Connection:

Endpoint=[redacted]

Entity Path:

[redacted]

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



ImageSoft.ServiceBusListener.Connection

Service Bus Connection String

Endpoint=sb://
[redacted]

Service Bus Structure

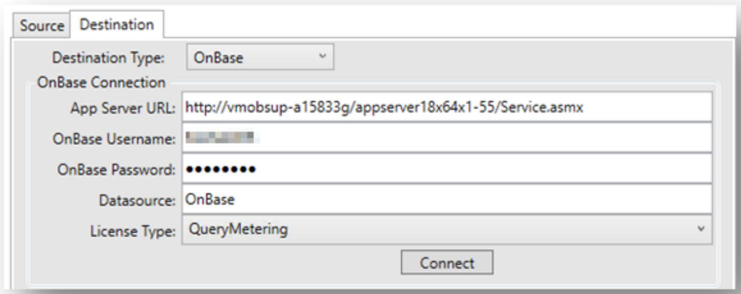
State:
Message Time To Live:
Move to DLQ on Expiration:
Maximum Delivery Count:
Other:

Service Bus Configuration

Service Bus Type:	Queue
Entity Path:	831226c6-6a90[redacted]
Subscription Name:	
Max Concurrent Calls:	6
Monitor Dead Letter Queue:	true

The Connection String used to connect to the service bus. This typically begins with 'Endpoint='.

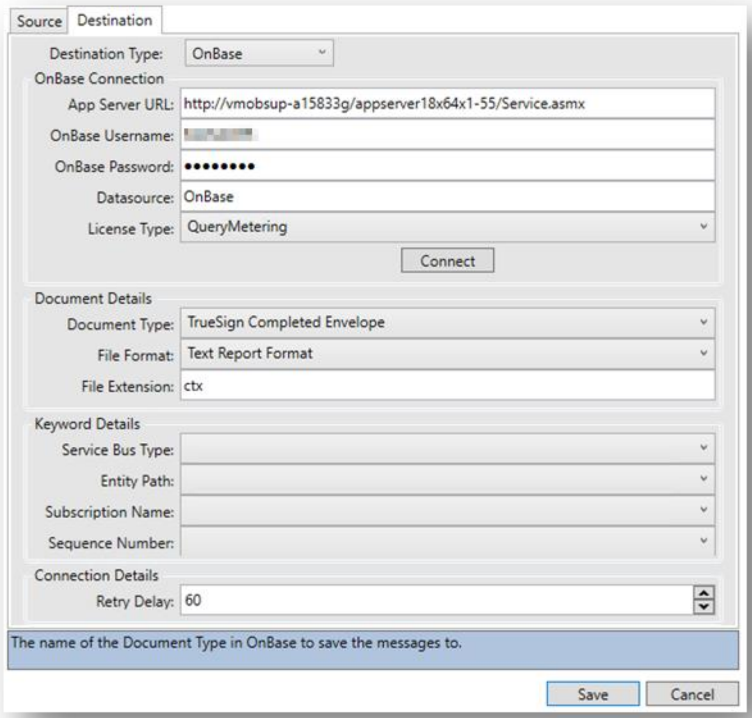
Once done with the Source connection strings, we need to configure the Destination for the incoming messages. In the Destination tab, we will configure the destination to OnBase and select a document type where the new messages will be stored under. For this step you will need an OnBase user with access 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”.



Destination Type: OnBase
OnBase Connection
App Server URL: http://vmobsup-a15833g/appserver18x64x1-55/Service.asmx
OnBase Username: [redacted]
OnBase Password: ••••••••
Datasource: OnBase
License Type: QueryMetering

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:



Destination Type: OnBase
OnBase Connection
App Server URL: http://vmobsup-a15833g/appserver18x64x1-55/Service.asmx
OnBase Username: [redacted]
OnBase Password: ••••••••
Datasource: OnBase
License Type: QueryMetering

Document Details
Document Type: TrueSign Completed Envelope
File Format: Text Report Format
File Extension: ctx

Keyword Details
Service Bus Type:
Entity Path:
Subscription Name:
Sequence Number:

Connection Details
Retry Delay: 60

The name of the Document Type in OnBase to save the messages to.

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

Workflow Properties are used very heavily within the Sample Lifecycle as well as all of the Unity Scripts. This section

Name	Description
TrueSignAnchors	Holds anchor data in JSON format for a document. This property is set and used by the TrueSign Add Document script.
TrueSignClientId	Holds the TrueSign Client Id of the envelope type that will be connecting to the TrueSign API. This property is required to use ANY of the TrueSign Scripts.
TrueSignClientSecret	Holds the TrueSign Client Secret of the envelope type that will be connecting to the TrueSign API. This property is required to use ANY of the TrueSign Scripts.
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. This property is used by the TrueSign Create Envelope script.
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. This property is used by the TrueSign Create Envelope script.
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. This property is used by the TrueSign Create Envelope script.
TrueSignEnvelopeId	Holds a GUID of the new envelope, populated from the TrueSign Create Envelope 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). This property must be set to the GUID of a valid envelope for any of the following Unity Scripts to work correctly: TrueSign Add Document TrueSign Check Envelope Status TrueSign Delete Envelope TrueSign Send Envelope TrueSign Download (if using in an “Interactive” signing context such as TrueSign Now)
TrueSignEnvelopeTitle	Holds a string that represents the title of the new envelope. This property is used by the TrueSign Create Envelope script. It is optional; if this property is not set, a default envelope name will be generated by the TrueSign Library.
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. This property is used by the TrueSign Create Envelope script.

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. This property is used by the TrueSign Create Envelope script.
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.
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. This property is used by the TrueSign Create Envelope script.
TrueSignNotifySigner	Holds a single or array of Boolean (true/false) values indicating whether an internal signer should receive a notification about the envelope or not. This is an optional property **Note**: in a multiple signer situation, the index must match to the correct signer – see example below. This property is used by the TrueSign Create Envelope script.
TrueSignError	If an error is encountered during the execution of **any** of the TrueSign scripts, the error message will be stored in this property. Please note that the text of the error message will be truncated to under 250 characters so that it will fit within an OnBase Note or OnBase Workflow Message.
TrueSignOverrideDeliveryMethod	Allows you to tell TrueSign how the completed envelope data should be delivered. Generally, with an OnBase integration the Delivery Method will either be "TrueSignAPI" or "ServiceBus". This property can optionally be used by the TrueSign Create Envelope script; if this is not set, the Envelope Type's default delivery method is used
TrueSignCancel	This property is set to “true” in workflow if an error is encountered that is severe enough that processing needs to be cancelled. This property helps to ensure that ad hoc tasks fail gracefully in scenarios where a user executed an ad hoc task against a group of 2 or more documents.
TrueSignDeliverAsTiff	If this property is set to "true", then once the document has been signed, it will return back to OnBase as a TIFF image rather than a PDF. This is an optional property used by the TrueSign Add Document script. If it is not set, the document will default to return as a PDF.
TrueSignDPIScale	Set the default DPI Scale for converting Notes to Anchors. This Scale value is used to transform the Width, Height, and X / Y coordinates of the OnBase Note into the Anchor position in TrueSign. The default value of this scale is 92; if you are noticing that anchors are scaling too small, you may need to lower this value to something like 32. If anchors are too large, you may need to increase the value to something like 150. This property is ONLY used in the OnBase Note > TrueSign Anchor conversion; it is not used anywhere other than the TrueSign Add Document script.

TrueSignEnvelopeRejected	This property is set by the TrueSign Download script. If the envelope was rejected by the signer, this property will be set to a value of "true"
TrueSignRejectedReason	This property is set by the TrueSign Download script. If the envelope was rejected by the signer, the reason for the rejection will be stored in this property.
TrueSignEnvelopeDocs	This property is set by the TrueSign Download script. Contains an array of all the OnBase document handles that were included in the envelope. (ie., if 3 documents were uploaded to a single envelope, this property would be set to a value like this: "3515316, 4316834, 6534145")
TrueSignEnvelopeDocHandle	This property is set by the TrueSign Download script. Contains the document handle of the TrueSign Completed Envelope document imported via the Service Bus Listener
TrueSignDownloadLock	This property is set by the TrueSign Download script. In order for the TrueSignDownload script to work correctly, none of the documents contained within the envelope can be locked. If the documents are locked, the script will be unable to update the revision. If a document is locked, this property will be set to "true" so that the completed envelope can be sent to a separate "Locked Document" error queue where it can be retried.
TrueSignURLMissing	After a TrueSign document has been signed and completed on TrueSign.com, it sometimes takes a few seconds for the signed document to be fully rendered and available for download over the API. This can cause problems in the TrueSign Now task, where a user may hit the “I’m Done Signing” button before TrueSign.com has had time to fully render the document. In this situation, the TrueSign Download script will wait 10 seconds and then try downloading it again. If the signed documents are STILL not available after 10 seconds, this property gets set to “true” so we can handle the issue further downstream in workflow.
TrueSignStatus	This property is set by the TrueSign Check Envelope Status script. It is set to the status of the current envelope. This property will be blank if the envelope cannot be found.
TrueSignCurrentSigner	This property is set by the TrueSign Check Envelope Status script. It stores the email address of the current signer of the document.
TrueSignCurrentDesigner	This property is set by the TrueSign Check Envelope Status script. It stores the email address of the first designer of the document (if there are multiple designers, only one is stored here)
TrueSignRemovePermissions	During an automated TrueSign upload, you may wish to limit the markup functionality that a signer is able to perform on a document. This property allows you to remove permissions from a signer. This property is used by the “TrueSign Create Envelope” script. For example, let's say you're uploading a document automatically to TrueSign that already has the correct anchors applied. In this case, you may not want the signer to be able to manually drag their signature onto a document; you'd prefer that they can only sign the document in the pre-anchored positions. In this case, you could set the "TrueSignRemovePermissions" property to "Signatures", which removes the ability for a signer to place an adhoc signature.

	If you want to remove multiple permissions, this property can be set to an array, for example if you wanted to remove the signer's ability to add Checkmarks, Dates, and Highlights, you will set the "TrueSignRemovePermissions" property = "Checkmark,Dates,Highlight" with the array option enabled. Here is a list of the permissions that can be removed: **Signatures** - Remove the ability to add adhoc signatures **Stamps** - Remove the ability to add stamps **Freetext** - Remove the ability to add freetext **Highlight** - Remove the ability to highlight **Checkmark** - Remove the ability to add a checkmark **Dates** - Remove the ability to add a date **CustomDate** - Remove the ability to add a custom date **Shapes** - Remove the ability to draw shapes **TypedSignature** - Remove the ability to create a new signature as typed **Sidebar** - Remove the sidebar (search, thumbnails) **QRCodeSignature** - Remove the ability to create a new signature via QR Code **GuidedSigning** - Remove the ability to use guided signing **NOTE:** If there are multiple signers on an envelope, setting this property will remove the permissions for ALL signers. There is no way to selectively remove permissions from one signer but not another. This would require custom updates to the library for your specific needs.

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 both internal signers or external signers for an envelope and signing is done in series by each signer.

Example of Multiple Signers

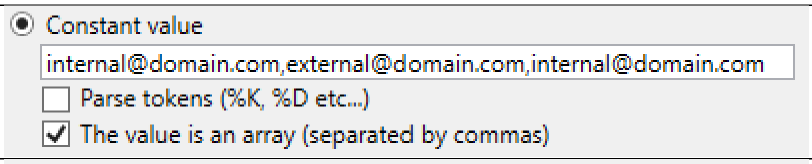
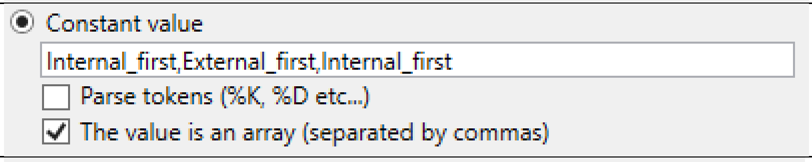
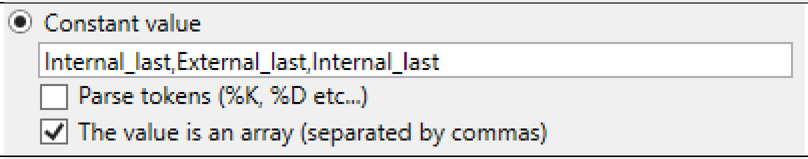
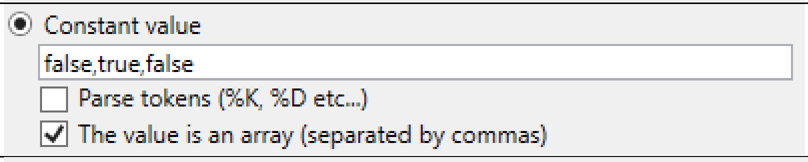
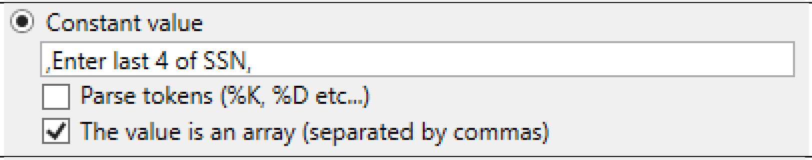
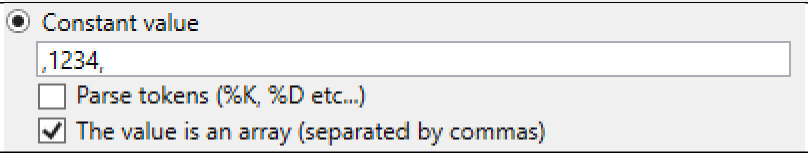
Let’s say we have a document that needs to be signed by one internal signer, one external signer and then by another internal signer. The following relevant property bags should be populated as follows:

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*
TrueSignNotifySigner	*True*	NULL	*False*

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	 ● Constant value internal@domain.com,external@domain.com,internal@domain.com ☐ Parse tokens (%K, %D etc...) ✓ The value is an array (separated by commas)
TrueSignFirstName	 ● Constant value Internal_first,External_first,Internal_first ☐ Parse tokens (%K, %D etc...) ✓ The value is an array (separated by commas)
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)

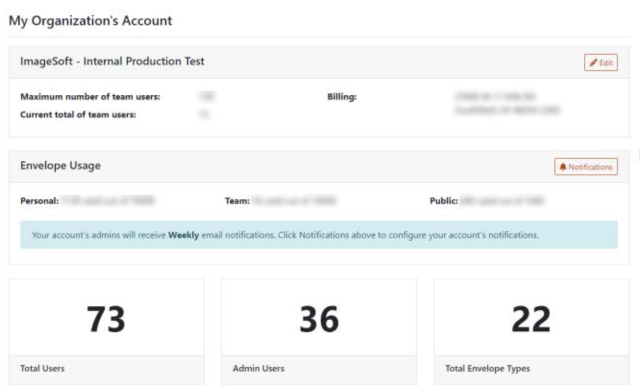
The same thing can be done using multi-instance keywords. An example of multi-instance keywords with required signers can be seen in the “Automated TrueSign Upload” queue, which extracts signer information from the “TrueSign Signer Information” keyword group and adds them as external signers to an envelope.

Licensing Considerations

Before implementing any TrueSign integrations into your production environment, you should be aware of what licenses you have purchased, and the implications of how your TrueSign integrations could consume those licenses.

To begin, you can view the current status of your TrueSign licenses via the TrueSign admin portal: https://app.truesign.com/Admin

From the admin portal, click on the "TrueSign Account" button to view your organization account dashboard:



My Organization's Account

ImageSoft - Internal Production Test

Maximum number of team users: [redacted]
Current total of team users: [redacted]
Billing: [redacted]

Envelope Usage

Personal: [redacted] Team: [redacted] Public: [redacted]

Your account's admins will receive **Weekly** email notifications. Click Notifications above to configure your account's notifications.

73 Total Users
36 Admin Users
22 Total Envelope Types

There are two types of licenses within TrueSign:

1. **Unlimited User License** – This can be applied to a TrueSign user and allows a user to send an unlimited number of licenses (regardless of type) from the TrueSign web interface.

2. **Envelope License** – Customers can purchase a set number of envelope licenses annually. If a user is not assigned an Unlimited License, then they will consume envelope licenses from this pool. All instances where the API performs the "Send" of the envelope will consume an Envelope License as well.

   However, If the TrueSign API is being used to send an envelope…

   - If the envelope is being sent only to internal TrueSign users who have an **Unlimited User License**, then *no per envelope license is consumed.*

   - If any signers on the document (whether internal or external) do not have an **Unlimited User License,** *an envelope license is consumed.*

The TrueSign Sample Life Cycle provides 4 distinct methods for uploading a document to TrueSign.com. Here are some explanations for how this licensing model works using these upload methods as an example:

- **TrueSign Now -** In this ad hoc task, this task is only configured to work if the signer is an internal user, **so if the user has a TrueSign account with an "Unlimited User License", no envelope will be consumed when using this task.**

- **TrueSign Design** - In this ad hoc task, the API only creates the envelope and adds the documents; it is always a user in the TrueSign Design web interface that clicks the "Send" button, **so an Envelope will not be consumed if the user designing the envelope has an "Unlimited User License"**

- **Send for Signature** - In this ad hoc task, the action of sending the document for signature is executed by the API, **so an Envelope license will be consumed if the document is sent to signer who does not have an "Unlimited User License"**

- **Automatic Upload** - In this scenario, a timer automatically uploads the document and sends it for signature exclusively via the API, **so an Envelope license will be consumed if the document is sent to signer who does not have an "Unlimited User License"**