ESig Implementation Guide

Instructions you can use for NGSD implementations

This guide will walk you through how to implement the new NGSD common ESignature component using either AlphaTrust or DocuSign, as well as what customizations are possible.

Table of Contents

[Prerequisites 1](#_Toc6398074)

[NGSD Screen Setup 1](#_Toc6398075)

[Copy Screens 1](#_Toc6398076)

[Adding Custom Variables 2](#_Toc6398077)

[Adding Role Variables 3](#_Toc6398078)

[Nested Product Mapping 4](#_Toc6398079)

[JavaScript Custom Assembly Setup 8](#_Toc6398080)

[Using the JSCA 8](#_Toc6398081)

[Lexis Nexis KBA 10](#_Toc6398082)

[Forms.Config Setup 12](#_Toc6398083)

[Sub Company Keys 13](#_Toc6398084)

[Server Procedure Configs 14](#_Toc6398085)

[.NET Custom Assembly 15](#_Toc6398086)

[PDF Mapping 15](#_Toc6398087)

[Workflows/Use Cases 16](#_Toc6398088)

[HIPAA Workflow 16](#_Toc6398089)

[APPEND Package Workflow 17](#_Toc6398090)

[Audit Reports 17](#_Toc6398091)

[Troubleshooting 17](#_Toc6398092)

# Prerequisites

NGSD understanding

At least base 9.1.12.1

Understanding of the JavaScript Custom Assembly

Understanding of the .Net Custom Assembly

You have contacted Michael Vergara to have your Carrier/Supplier set up in the eSig Service.

# NGSD Screen Setup

The first step is you’ll want to copy screen from the Test eSig Supplier project into your project. These will become your project screens, it is highly reccomended that you don’t alter these screens after being copied into your project.

## Copy Screens

Reach out to Software Config at @Software Configuration Requests and ask them to copy the following screens into your project.

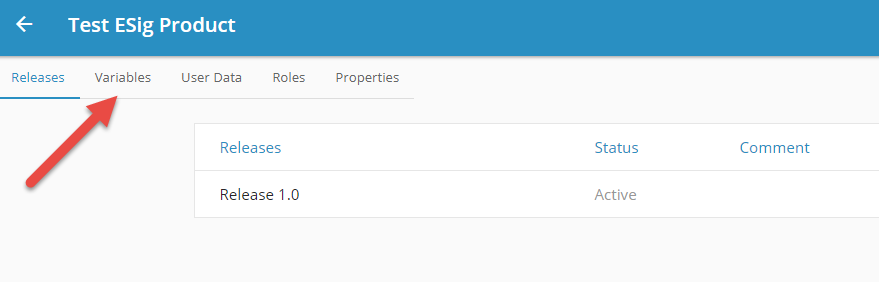
* SignatureMethod (**5c1124bce8dee6ce7c5578d0**)
* Clickwrap LoadScreen (**5baa79ecaec9c193fe7d3a9a**)
* Clickwrap LeaveBehind (**5babca774656cd76d8627dee**)

To Project version #: [Put your project’s version # here]

You should also create a ValidateAndLock screen in your project. You can then drop the global Standard\_ValidateAndLock template onto that screen.

Once these screens are copied to your project you’ll need to make sure you move them into the correct order within your application. (Signature Method screen should be somewhere after your validate and lock screen)

## Adding Custom Variables

At this point you’ll need to set up some custom variables that can be set, we will need to set up a translation of these to the common eSignature component later. Start by clicking the Variables tab. 

Add the following variables as a textbox to this list.

* TOUPath
* ESIG\_packageGuid
* ESIG\_Consent\_title
* ESIG\_Consent\_review
* ESIG\_Consent\_message
* ESIG\_Consent\_chkAgree
* ESIG\_Consent\_btnReview
* ESIG\_Consent\_lnkPrint
* ESIG\_Consent\_chkTOU
* ESIG\_Consent\_btnCancel
* ESIG\_Consent\_btnDecline
* ESIG\_Consent\_btnAgree
* ESIG\_Consent\_hideviewforms
* ESIG\_Consent\_norequirereviewforms (used to show the Review Forms button, but not be required)
* ESIG\_Consent\_hidecancel
* ESIG\_Consent\_hidedecline
* ESIG\_UseDisclosure

NOTE: If above is set to False, then the following items are not needed for disclosure

* ESIG\_Disclosure\_title
* ESIG\_Disclosure\_proof
* ESIG\_Disclosure\_cancel
* ESIG\_Disclosure\_decline
* ESIG\_Disclosure\_next
* ESIG\_Disclosure\_message
* ESIG\_Disclosure\_showDL
* ESIG\_Disclosure\_showPassport
* ESIG\_Disclosure\_showGC
* ESIG\_Disclosure\_showGovID
* ESIG\_Disclosure\_showPK
* ESIG\_Disclosure\_showStateID
* ESIG\_Disclosure\_hidecancel
* ESIG\_Disclosure\_hidedecline
* ESIG\_Disclosure\_showinputfields
* ESIG\_Disclosure\_Accepted
* ESIG\_ThankYou\_title
* ESIG\_ThankYou\_message
* ESIG\_ThankYou\_text
* ESIG\_ThankYou\_btnViewForms
* ESIG\_ThankYou\_success
* ESIG\_ThankYou\_btnClose
* ESIG\_ThankYou\_closemessage
* ESIG\_Decline\_title
* ESIG\_Decline\_message
* ESIG\_Decline\_btnResume
* ESIG\_Decline\_btnCancel
* ESIG\_Decline\_declined
* ESIG\_Mobile\_Message

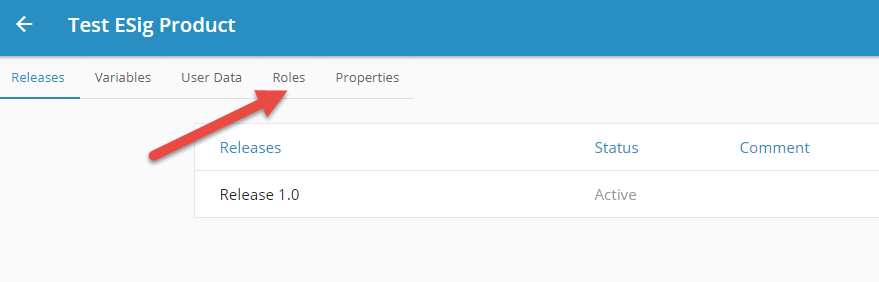
If you will use the disclosure screen to capture ID info for the signer.

* ROLE1- [your max role, not agent]
  + \_CW\_DeclineReason (Ex. ROLE1\_CW\_DeclineReason)
  + \_DLState (Ex. ROLE1\_DLState)
  + \_Disclosure\_ID
  + \_ID
  + \_IDNumber
  + \_IDOtherNumber

## Adding Role Variables

Next, we’ll need to map out the user’s available, these are suggestions but can be altered to meet your specific project, but please bear in mind, and changes to the suggested values will require you to review the rules on the screen copied into your project via the step above.

Start by clicking the Roles tab.



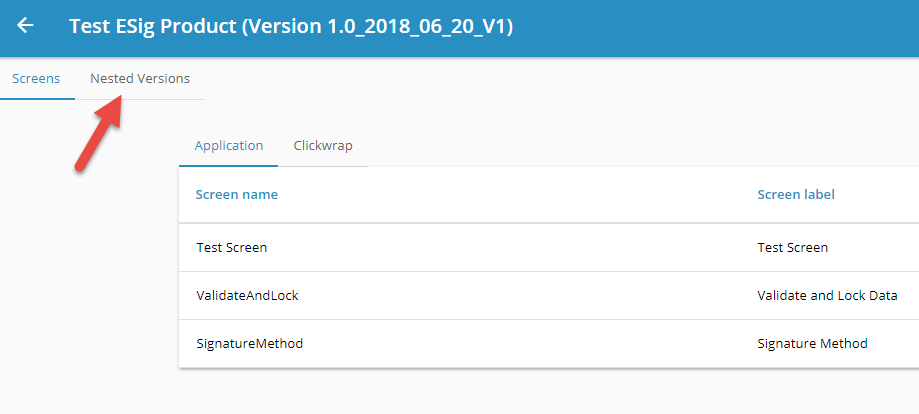
The roles and number of roles can vary based on your project, here are the ones used by the standard.

* Agent 99
* Proposed Insured 1
* Other Insured 2
* Owner 1 3
* Owner 2 4
* Owner 4 5
* Owner 5 6
* Payor 7
* Payor 2 8
* Parent 9 (if different than applicant)
* Applicant 10 (if not owner or insureds)
* Role 70 Leave Behind 70 (this does not directly tie to a specific role above)
* Role 71 Leave Behind 71
* Role 72 Leave Behind 72
* Role 73 Leave Behind 73

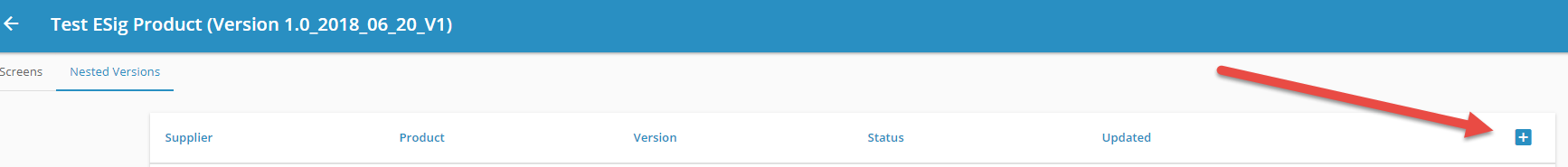
## Nested Product Mapping

A nested product is another product that can be referenced and used directly into your project. In this case this will allow your project to open up a modal window and display all the eSignature logic without you having to build or develop these screens.

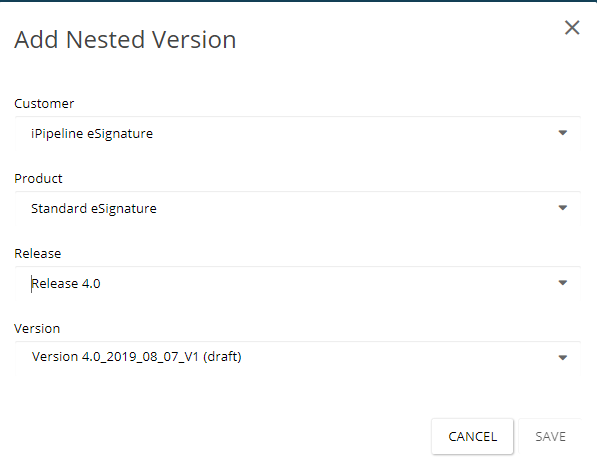
Start by going into your product within NGSD and click the Nested Versions tab.



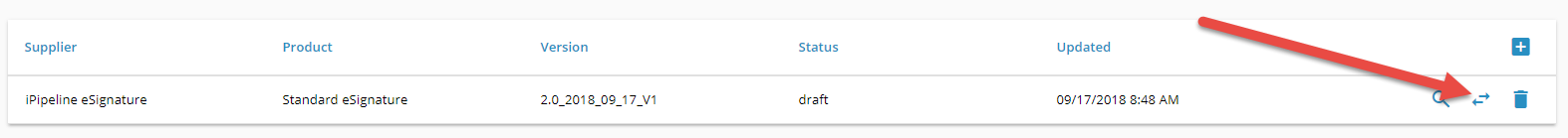
From here you’ll need to click the “+” plus icon to the far right.



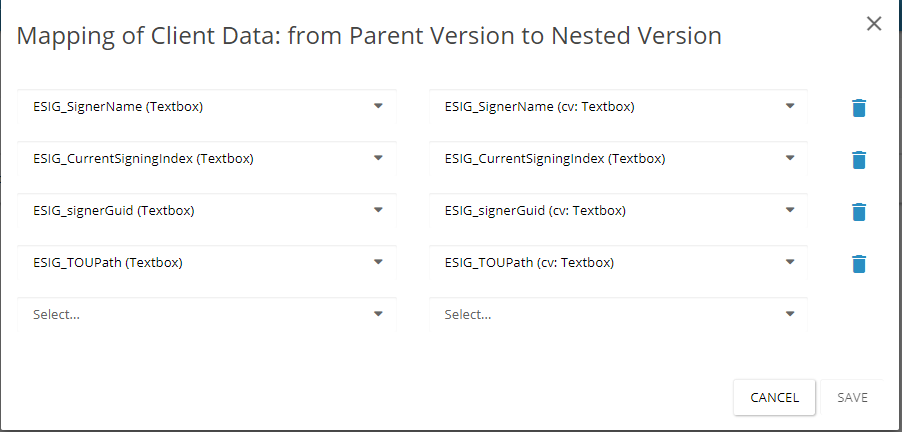
Next, you’ll see a popup, enter the following info to link to the standard eSignature project.



Now you’ll see the “Nested” eSignature project. From here you’ll need to map some variables from your project in order to transfer over to the eSignature project. Please note, the eSignature project DOES NOT HAVE ACCESS TO CLIENT DATA. So, you need to map certain elements from your project to see it in the eSignature project. To do this click the “Arrows” icon.



Here you’ll see a popup like the one below.



The dropdowns on the left represent the variables within your project, the dropdowns on the right represent the variables available within the eSignature project. Here, you’ll want to create a 1 to 1 mapping of these variables. The following are the minimum needed.

ESIG\_SignerName (This is used to send the signer name to display in eSig)

ESIG\_CurrentSigningIndex (This is used to determine the role in eSig)

ESIG\_signerGuid (Used to tell eSig who the current signer is)

ESIG\_TOUPath (Path to tell eSig where to get the html file for the Terms of Use)

ESIG\_SignerRoleType (Indicate the role to be displayed Ex: Propose Insured)

Carrier\_itmtxt (This is used to display the carrier’s name in eSig for the Thank you screen)

ESIG\_Consent\_title

ESIG\_Consent\_review

ESIG\_Consent\_message

ESIG\_Consent\_lnkPrint

ESIG\_Consent\_chkTOU

ESIG\_Consent\_chkAgree

ESIG\_Consent\_btnReview

ESIG\_Consent\_btnDecline

ESIG\_Consent\_btnCancel

ESIG\_Consent\_btnAgree

ESIG\_Consent\_hidereviewforms

ESIG\_Consent\_norequirereviewforms

ESIG\_Consent\_hidecancel

ESIG\_Consent\_hidedecline

ESIG\_Consent\_Resume

ESIG\_Consent\_chkTOU\_Accepted

ESIG\_Consent\_chkAgree\_Accepted

ESIG\_UseDisclosure (This indicates if you what to use the disclosure screen, or go right to the TOU Screen)

NOTE: If above is set to False, then the following items are not needed for disclosure

ESIG\_Disclosure\_cancel

ESIG\_Disclosure\_next

ESIG\_Disclosure\_decline

ESIG\_Disclosure\_message

ESIG\_Disclosure\_proof

ESIG\_Disclosure\_title

ESIG\_Disclosure\_showDL

ESIG\_Disclosure\_showPassport

ESIG\_Disclosure\_showGC

ESIG\_Disclosure\_showGovID

ESIG\_Disclosure\_showPK

ESIG\_Disclosure\_showStateID

ESIG\_Disclosure\_hidedecline

ESIG\_Disclosure\_hidecancel

ESIG\_Disclosure\_showinputfields

ESIG\_Disclosure\_Accepted

ESIG\_ThankYou\_title

ESIG\_ThankYou\_message

ESIG\_ThankYou\_text

ESIG\_ThankYou\_btnViewForms

ESIG\_ThankYou\_success

ESIG\_ThankYou\_btnClose

ESIG\_ThankYou\_closemessage

ESIG\_Decline\_title

ESIG\_Decline\_message

ESIG\_Decline\_reason

ESIG\_Decline\_confirm

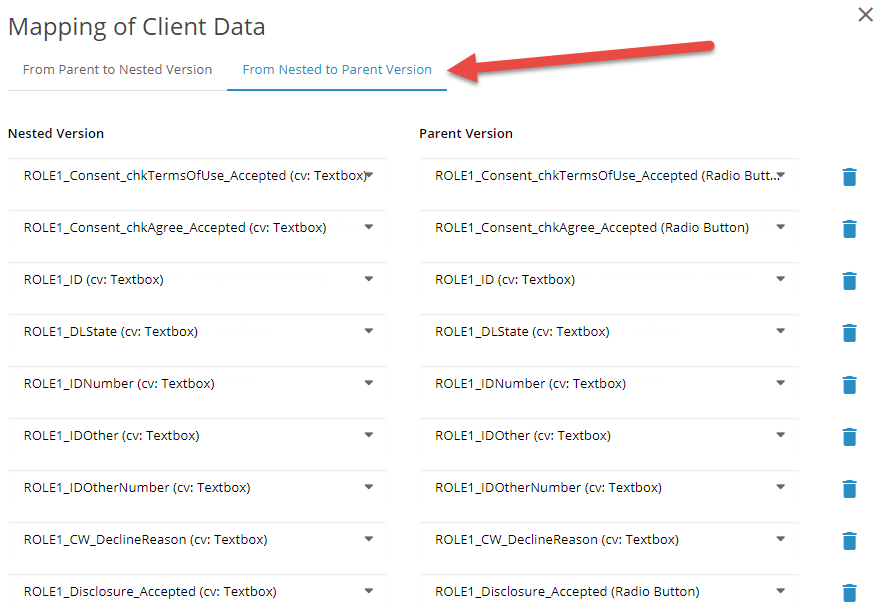
ESIG\_Decline\_btnResume

ESIG\_Decline\_btnCancel

ESIG\_Decline\_declined

ESIG\_Mobile\_Message

To get data back from the eSignature common component back into your project you’ll need to click the tab at the top of the popup that says “From Nested to Parent Version”



Here you’ll need to create the custom variables in your project just like you did previously. However, this time you can limit what you need to capture. If you don’t plan on using the common component Disclosure screen you can skip most of the next items with the exception of the highlighted items below. If you do, you can follow the following format.

ROLE(x)\_ID (This will save the chosen ID type from the Disclosure screen)

ROLE(x)\_DLState (This will save the state chosen if the ID is driver’s license or state id)

ROLE(x)\_IDNumber (This will save the ID value entered)

ROLE(x)\_IDOther (If id of Other is chosen then what’s entered on the Dicslosure screen will be saved)

ROLE(x)\_IDOtherNumber (If id of Other is chosen then the number entered on the Disclosure screen will be saved)

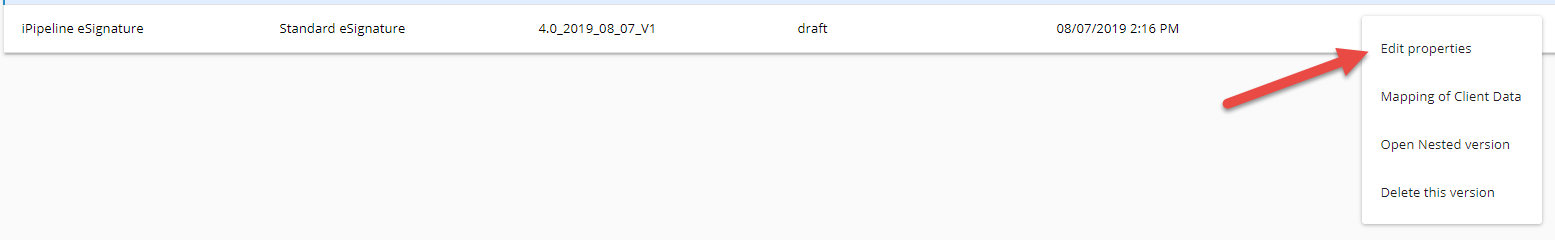
For the clickwrap decline reason you’ll want to map the following.

ROLE(x)\_CW\_DeclineReason.

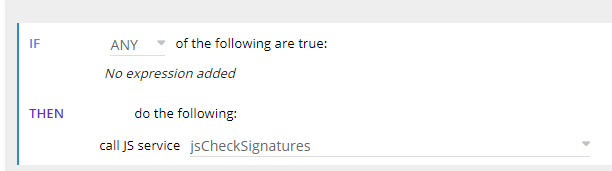
ROLE(x)\_Consent\_chkTermsOfUse\_Accepted

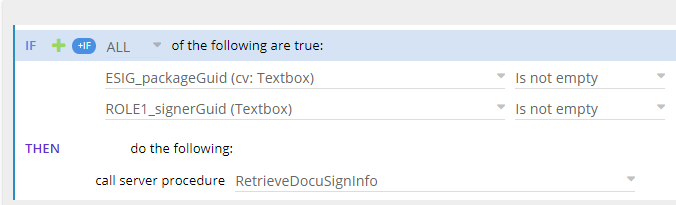
ROLE(x)\_Consent\_chkAgree\_Accepted

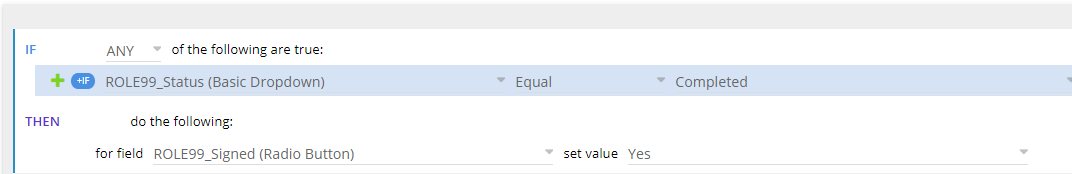
Finally, the last step is to add the 4 rules to the NestedModalClose function in your nested solution. Go to Nested Versions, click the ellipsis at the end, and choose Edit Properties.

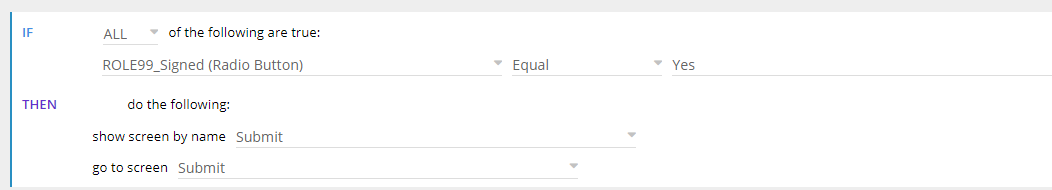


There, you will need to write 4 small rules.

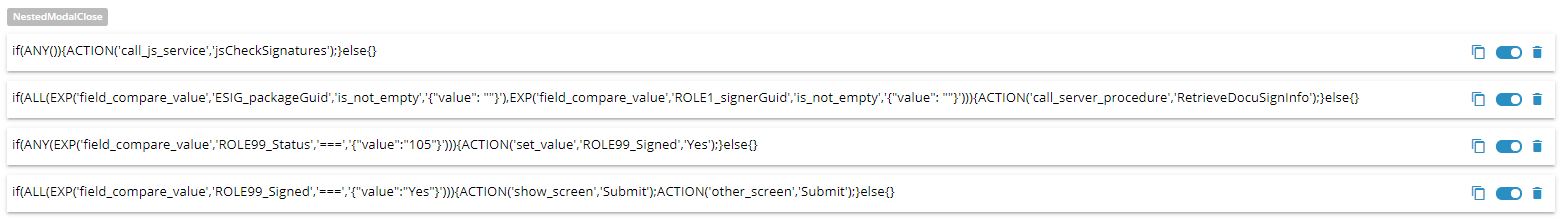








Your end result should have your rules in the following order.



You may find it necessary to change some of the logic base on your project.

# JavaScript Custom Assembly Setup

To set up the JSCA you’ll need to add a few js files to your custom assembly. You can grab the following js files from this TFS location. ($/CossTools/Demo/9801/Main/ServerSideProcedures/SDR/Standard\_ESignature)

Download the items from TFS and open up the workspace in “Visual Studio Code”. Copy the following items over to your workspace project. (If you already have one of these items please view the content and copy over only what you feel is necessary.

//Constants/gaids.js

//Constants/iccStates.js

//Constants/prefixes.js

//customAssemblies/lockApp.js

//customAssemblies/signatureFunctions.js

//customAssemblies/unlockApp.js

//customAssemblies/lockHIPAA.js

//customAssemblies/checkSigner.js

//customAssemblies/checkPackage.js

Then review the main.js and ensure you add the follow three mappings, make sure to add any that you choose from the above list.

import lockApp from 'product/customAssemblies/lockApp';

import unlockApp from 'product/customAssemblies/unlockApp';

import { checkIfAllSignaturesCaptured } from 'product/customAssemblies/signatureFunctions'

window.global.App.jsCA = {

jsLockApp: lockApp,

jsUnlockApp: unlockApp,

jsCheckSignatures: checkIfAllSignaturesCaptured,

}

## Using the JSCA

The JSCA is based on your understanding of your project, the logic examples defined below are only examples and you MUST changes these to match your project’s logic.

#### Gaids.js

Change the GAID value in here to a comma separated list of your GAIDs.

#### iccStates.js

This is a comma separated list of ICC states. You can alter this list, but also use it as an example to list other values specific to your project (NAIC states would be another example).

#### Prefixes.js

Some projects have this defined already, but if not, the point here is to list out all of the prefixes for the different signing parties to keep everything in one spot should a change need to happen.

#### UnlockApp.js

The Unlock app is strictly used for deleting anything related to eSignature, you can delete a specific value by using cdHelper.deleteValue, or anything with a prefix of by using ceHelper.deleteValuesByPrefix

#### signatureFunctions.js

This file is used to determine if and when the agent can sign. In the globalValues you’ll want to list all of your ROLE(s) with the suffix of \_Signed or \_Needed. Then create an if statement to check each role value and set the Boolean value to determine if the agent is ready to sign.

#### LockApp.js

You will spend more of your time here. Review the code, it should be commented enough to provide some understanding of how it works. This will be written reading the code bottom up.

You will need a role place holder for each signer, this will include the signing index and any signing suffix that goes to that signer.

For example, a Proposed Insured is usually and index of 1, and signs most of the PDFs using SIG\_X\_PI, in this case, the suffix is PI. You may also run into a situation where you have a HIV form where depending on what the PI answers on the screen, maps to a Yes or No. In this situation you may map SIG\_X\_PIHIVYES and SIG\_X\_HIVNO. Now you’ll write logic to append one of those suffixes (PIHIVYES or PIHIVNO) to that role 1 index. So if the Proposed Insured answers Yes, the result of the role1Placeholder would be 1,PI,PIHIVYES.

The above must be repeated for each unique situation for each signer. Once all of this is complete, the client data variable or ROLE\_Suffixes must be set. This is what base uses to determine who will sign. An example here using the above with an agent will look like this... 1,PI,PIHIVYES,99,AGENT.

Also, you need to create logic within your project to determine if a signer is even needed based on the question asked within the app. In that case you’ll need to set specific info for that signer (see the code for detail). If you plan to include a signer in your request, you MUST ensure they have a task to perform (sign, initial, etc), otherwise you’ll receive an error creating the package.

Another item to make note of are dates, you cannot have a date without a signature or initial being associated with it. To do this you need to set a DateAnchor variable. So, if you want a date associated with the SIG\_X\_PI, then you also will need to set SIG\_X\_PI\_DateAnchor, and set its value to what this date anchor belongs to, so in this case SIG\_X\_PI.

Signature mappings MUST follow the standard signature mapping.

SIG\_X = Signature

SIG\_I = Initial

SIG\_D = Date

SIG\_Z = Date / Time

SIG\_T = Time

SIG\_CITY = City

SIG\_STATE = State

Lastly, there are specific elements that you can set for the signing experience itself. You’ll need to provide data to those variables you mapped above when referencing the Nested Version. Also, you can set the follow specific signature options.

ESIG\_FontSize (this is based on pixels, the higher the number, the larger the font.)

ESIG\_Font (This sets the font type, options are, Jenna Sue (default), Script, Segue and Arial. It’s recommended to not set this value).

ESIG\_FontColor (This set the color of the signature, options are Red, Blue and Black).

ESIG\_Underline (puts and underline under the signature)

ESIG\_Bold (Bolds the signature)

ESIG\_Italic (Makes the signature italicized)

## Lexis Nexis KBA

The Lexis Nexis KBA will force the signer to answer knowledge-based authentication questions generated by Lexis Nexis. To do so see the following code.

if (cdHelper.getValue('UseKBA') === 'Yes')

{

//-----Set your signing location info here, this is not required-----

cdHelper.setData('ROLE1\_FullName', cdHelper.getValue('PI\_FullName'));

cdHelper.setData('ROLE1\_FirstName', cdHelper.getData(`${prefixes.PIPREFIX}${nameFields.FIRST\_NAME}.value`));

cdHelper.setData('ROLE1\_LastName', cdHelper.getData(`${prefixes.PIPREFIX}${nameFields.LAST\_NAME}.value`));

cdHelper.setData('ROLE1\_SSN', cdHelper.getData(`${prefixes.PI\_PREFIX}SSN.value`));

cdHelper.setData('ROLE1\_Address1', cdHelper.getData(`${prefixes.PI\_PREFIX}AddressLine1.value`));

cdHelper.setData('ROLE1\_Address2', cdHelper.getData(`${prefixes.PI\_PREFIX}AddressLine2.value`));

cdHelper.setData('ROLE1\_City', cdHelper.getData(`${prefixes.PI\_PREFIX}City.value`));

cdHelper.setData('ROLE1\_State', cdHelper.getData(`${prefixes.PI\_PREFIX}State.value`));

cdHelper.setData('ROLE1\_ZipCode', cdHelper.getData(`${prefixes.PI\_PREFIX}ZipCode.value`));

cdHelper.setData('ROLE1\_DOB', cdHelper.getData(`${prefixes.PI\_PREFIX}DOB.value`));

cdHelper.setData('ROLE1\_Phone', cdHelper.getData(`${prefixes.PI\_PREFIX}Phone.value`));

cdHelper.setData('ROLE1\_Country','US');

cdHelper.setData('ROLE1\_AuthenticationMethod','6');

cdHelper.setData('ROLE1\_IDCheck','TRUE');

let continueFail = cdHelper.getValue('ROLE1\_ContinueOnFail');

if (continueFail === '' || continueFail === 'FALSE')

{

continueFail = '0'

}

else if (continueFail === 'TRUE')

{

continueFail = '1';

}

cdHelper.setData('ROLE1\_ContinueOnFail',continueFail);

cdHelper.setData('ROLE1\_AllowedAttempts','3');

cdHelper.setData('ROLE1\_SSNType', '9');

cdHelper.setData('ROLE1\_DisplayLevel',cdHelper.setData('KBADisplayType'));

//N-Always show it

//Y-By Pass the first time, but show if there are errors processing the In-Wallet data

//X-Never show it

}

This is only meant to be a sample. In the example you see options for continue on fail tied to a UI object. You don’t need that, you’ll either hard code it, or write some business logic around when to set that. The same goes for the KBADisplayType.

To test this below are three “Fake” people you can use to test the KBA screen.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test PI #1 | Test PI #2 | Test PI #3 |
| PI First Name | Mary | Cheryl | Kenny |
| PI Last Name | Bbegonia | Ssugarmaple | Rraspberry |
| PI Test SSN | 292-78-7965 | 601-03-7919 | 600-65-9534 |
| PI Address Line 1 | 6847 Jaguar Road | 3948 Pug Road | 2323 Kangaroo Road |
| PI City | Concord | Concord | Daly City |
| PI State | CA | CA | CA |
| PI Zip Code | 94521 | 94521 | 94014 |

# Forms.Config Setup

The new setup for forms config removes the concept of signing packages. You only need to create the main package, all others will be controlled via the “FORMS” section of the forms config file. An example is this

<Form Name="MainApp-US" Pdf="MainApp-US" Signatures="SIG\_X\_PI,SIG\_D\_PI,SIG\_X\_OWN1,SIG\_D\_OWN1,VIEW\_V\_PAYOR,SIG\_X\_AGENT,SIG\_D\_AGENT"/>

Let’s dig into this example. Here we have the typically defined form name and pdf name, nothing new there. But, we also have the newly created Signatures node. This is used to house all of the signatures that are mapped to this particular PDF, plus, anyone who doesn’t have a signature that may need to view this. **NOTE: Any PDF that does not have a Signatures node will be seen by all signing parties. So not all forms need a Signatures node.**

In this form it will be signed and dated by the PI, OWN1 if it exists and the AGENT. But, also, if the PAYOR exists they will be able to see it in their “forms package” even though they aren’t signing it.

You set what signature are visible in the JSCA.

There are also two other keys that can be set for each form. (BW and DPI)

BW (can be set to True or False, recommended to always be set to True). This will set to form to black and white, thus decreasing the size. Ex<Form Name="MainApp-US" Pdf="MainApp-US" BW=”True” Signatures="SIG\_X\_PI,SIG\_D\_PI,SIG\_X\_OWN1,SIG\_D\_OWN1,VIEW\_V\_PAYOR,SIG\_X\_AGENT,SIG\_D\_AGENT"/>

DPI refers to the Dots Per Inch, the resolution of the PDF. This is recommended to be set whenever a form doesn’t need to be signed. EX <Form Name="MainApp-US" DPI=”54” Pdf="MainApp-US" Signatures="SIG\_X\_PI,SIG\_D\_PI,SIG\_X\_OWN1,SIG\_D\_OWN1,VIEW\_V\_PAYOR,SIG\_X\_AGENT,SIG\_D\_AGENT"/>

There is also another value that can be set called ESigGroup. (See Workflows for use cases)

EX. <Form Name="HIPAA-ICC" Pdf="HIPAA-ICC" ESigGroup="HIPAA" Signatures="SIG\_X\_PIHIPAA,SIG\_D\_PIHIPAA"/>

# Sub Company Keys

There are only 3 sub co keys that need to be added to your GAID. Make sure you find your supplier ID from iService to enter below.

<!-- eSig Subco Keys -->

<Key Name="USE\_ESIG\_COMMON\_SIGNATURE\_COMPONENT">true</Key>

<!-- iService/AdminPipe CarrierID -->

<Key Name="ESIG\_COMMON\_SIGNATURE\_SUPPLIER\_ID">[Your Carrier ID]</Key>

<!-- 1=DocuSign 2=AlphaTrust -->

<Key Name="ESIG\_COMMON\_SIGNATURE\_PROVIDER\_ID">2</Key>

If you plan on using the ability to review all the forms in AT, Acknowledgements will be generated for all signers for forms have no signatures (which is the recommended approach)

<Key Name="ESIG\_GENERATE\_ACK\_FOR\_ALL\_SIGNERS">TRUE</Key>

OR

If you plan on using the ability to review all the forms in AT, Acknowledgements will be generated for the roles specified with View Only Signature tags in the CreatePackage Requests​ (which is the recommended approach)

<Key Name="ESIG\_GENERATE\_ACK\_FOR\_VIEW\_ONLY">TRUE</Key>

If you do not want to support singer signing at all you can set this key to force the user to always have to click to sign.

Key Name="ESIG\_COMMON\_SIGNATURE\_APPLICATION\_TYPE">FontStamped</Key>

If you want Alpha Trust to land on the top of the page before starting signing, please add the following config key

<Key Name="ESIG\_COMMON\_SIGNATURE\_START\_SIGNING\_ON\_PAGE\_ONE">true</Key>

Set this key if you have forms that need to be merged, such as overflow to main app

<Key Name="MERGE\_FORMS\_BEFORE\_SENDING\_TO\_ALPHA\_TRUST">TRUE</Key>

Set this key if you want to have all signers acknowledge any form they don’t sign

<Key Name="ESIG\_GENERATE\_ACK\_FOR\_ALL\_SIGNERS">TRUE</Key>

Set this key if you want to specify what forms signers will see that have no signatures

<Key Name="ESIG\_GENERATE\_ACK\_FOR\_VIEW\_ONLY">TRUE</Key>

#### For Your Local (add these to the defaults section)

<Key Name="ESIG\_COMMON\_SIGNATURE\_LOG\_LEVEL">debug</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_UI\_EXPIRE\_IN\_SECONDS">20</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_UI\_CLIENT\_ID">esig-client-igo-browser</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_UI\_CLIENT\_SECRET">gRrsAseZHsaYo9kb2uAFSYcw2i4qB8FSctZbvjMJPxDaPedJYlhv6l3e6gCzc2lb</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_GET\_PING\_TOKEN\_API\_ENDPOINT">http://127.0.0.1/CossEnterpriseSuite/(S([SESSION\_ID]))/api/eSigToken</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_COMPONENT\_EXTERNAL\_ORIGIN">https://esig-qd2.ipipeline.com</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_APP\_ID">1</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_COMPONENT\_URL">https://esig-core-qd2.dv.ipipenet.com/api/Package</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_ENVIRONMENT\_CODE">QD2</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_RETURN\_URL">http://www.google.com</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_ISSUE\_URL">https://federate-qa.ipipeline.com/as/token.oauth2</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_CLIENT\_ID">esig-client-igo</Key>

<Key Name="ESIG\_COMMON\_SIGNATURE\_PING\_FEDERATED\_TOKEN\_CLIENT\_SECRET">atuNhCsI3Uhv0jzEPWylQ9F8D1sXfsXCwJVRnbg9TqaH5AUR1TMTLzcwjSVr5nlc</Key>

<Key Name="SAVE\_ESIG\_COMMON\_API\_REQUESTS\_AND\_RESPONSES">TRUE</Key>

**NOTE: This key probably exists on your local in the defaults section, overwrite it.**

<Key Name="EXCLUDED\_FIELDS">Submitted\_CultureName,Submitted\_Date,Submitted\_Guid,Submitted\_Names,Submitted\_Time,Submitted\_Utc,myScreenHistory,CaseStatus,CaseStatus\_fmt,ProducersNotes,cossIsClientShared,cossIsClientLocked,AIS\_STATE,AIS\_STATE\_fmt,UpdatedDateTime,AIS\_RESULT,AIS\_RESULT\_fmt,ESIG\_LastApiRequestFileName,ESIG\_LastApiResponse,ESIG\_LastApiResponseFileName,ESIG\_Completed,ESIG\_Created,ESIG\_StatusDescription,ESIG\_StatusName,ROLE1\_Completed,ROLE1\_Created,ROLE1\_Description,ROLE1\_ESIGDocumentDownloadUrl,ROLE1\_ESIGDocumentViewUrl,ROLE1\_ESIGViewingUrl,ROLE1\_SignedDate,ROLE1\_Status,ROLE1\_StatusDate,ROLE1\_StatusDescription,ROLE2\_Completed,ROLE2\_Created,ROLE2\_Description,ROLE2\_ESIGDocumentDownloadUrl,ROLE2\_ESIGDocumentViewUrl,ROLE2\_ESIGViewingUrl,ROLE2\_SignedDate,ROLE2\_Status,ROLE2\_StatusDate,ROLE2\_StatusDescription,ROLE3\_Completed,ROLE3\_Created,ROLE3\_Description,ROLE3\_ESIGDocumentDownloadUrl,ROLE3\_ESIGDocumentViewUrl,ROLE3\_ESIGViewingUrl,ROLE3\_SignedDate,ROLE3\_Status,ROLE3\_StatusDate,ROLE3\_StatusDescription,ROLE4\_Completed,ROLE4\_Created,ROLE4\_Description,ROLE4\_ESIGDocumentDownloadUrl,ROLE4\_ESIGDocumentViewUrl,ROLE4\_ESIGViewingUrl,ROLE4\_SignedDate,ROLE4\_Status,ROLE4\_StatusDate,ROLE4\_StatusDescription,ROLE99\_Completed,ROLE99\_Created,ROLE99\_Description,ROLE99\_ESIGDocumentDownloadUrl,ROLE99\_ESIGDocumentViewUrl,ROLE99\_ESIGViewingUrl,ROLE99\_SignedDate,ROLE99\_Status,ROLE99\_StatusDate,ROLE99\_StatusDescription,CWEmailText\_1,CWEmailText2,CWEmailText3,CWEmailText4,CWEmailText70,CWEmailText71,CWEmailTitle1,CWEmailTitle2,CWEmailTitle3,CWEmailTitle4,CWEmailTitle70,CWEmailTitle71,CWRole99,CWPassword99,CWPasswordEncrypted99,CWUserID99,CWLoginURL\_99,ESB\_JSON\_LISTENER\_REQUEST\_1,ESB\_JSON\_LISTENER\_REQUEST\_2,ESB\_JSON\_LISTENER\_REQUEST\_3,ESB\_JSON\_LISTENER\_REQUEST\_4,ESB\_JSON\_LISTENER\_REQUEST\_5,ESB\_JSON\_LISTENER\_REQUEST\_6,ESB\_JSON\_LISTENER\_REQUEST\_7,ESB\_JSON\_LISTENER\_REQUEST\_8,ESB\_JSON\_LISTENER\_REQUEST\_9,ESB\_JSON\_LISTENER\_REQUEST\_10,ESB\_JSON\_LISTENER\_REQUEST\_11,ESB\_JSON\_LISTENER\_REQUEST\_12,ESB\_JSON\_LISTENER\_REQUEST\_13,ESB\_JSON\_LISTENER\_REQUEST\_14,ESB\_JSON\_LISTENER\_REQUEST\_15,ESB\_JSON\_LISTENER\_REQUEST\_16,ESB\_JSON\_LISTENER\_REQUEST\_17,ESB\_JSON\_LISTENER\_REQUEST\_18,ESB\_JSON\_LISTENER\_REQUEST\_19,ESB\_JSON\_LISTENER\_REQUEST\_20,ESB\_JSON\_LISTENER\_REQUEST\_21,ESB\_JSON\_LISTENER\_REQUEST\_22,ESB\_JSON\_LISTENER\_REQUEST\_23,ESB\_JSON\_LISTENER\_REQUEST\_24,ESB\_JSON\_LISTENER\_REQUEST\_25,ESB\_JSON\_LISTENER\_REQUEST\_26,ESB\_JSON\_LISTENER\_REQUEST\_27,ESB\_JSON\_LISTENER\_REQUEST\_28,ESB\_JSON\_LISTENER\_REQUEST\_29,ESB\_JSON\_LISTENER\_REQUEST\_30</Key>

# Server Procedure Configs

Please refer to the CossEnterpriseSuiteServerProcedures.config file in Sharepoint for direct examples. The procedures you’ll want to copy over to your project are:

**NGSD\_SendCWEmails**, this check if your signer is present or not. If not, it calls a custom assembly (change the name of the custom assembly to your CA name. Also make not that an AUDIT item is written to client data. Any item that starts with AUDIT, will be available via a project specific audit report if you want it. If not, you can remove the AUDIT pieces of data.

**NGSD\_ResendClickWrap,** will be used to resend any CW emails, please make sure to change the custom assembly name in the example.

**ProcessSignatureListenerEvent,** this procedure is used as a backup in case the signer closes their browser before the modal signing window completes. You can review the casestatus and casestatushover items and indicate the text you’d like displayed.

**NGSD\_LockApplication**, you’ll call this when the app is locked on the button click event. Review the LockUnlock .NET CA call.

**NGSD\_UnlockApplication**, you will call this on the unlock event of the LockAndValidate screen.

# .NET Custom Assembly

For the .NET CA you’ll need to pull in three items. SetupCWValues.vb, LockUnlock.vb and the TriggerForms.vb (You may already have some of these).

The entry point into these items is within your FormsScreenServer.vb

Private Shared \_eventHandlerMap As Dictionary(Of String, AbstractEventHandler) = CreateDictionary(Of String, AbstractEventHandler) \_

(

"ViewForms".ToUpper, New EventHandlers.TriggerForms,

"LockUnlock".ToUpper, New EventHandlers.LockUnlock,

"SetupCWValues".ToUpper, New EventHandlers.SetupCWValues

)

# PDF Mapping

When mapping the PDFs you must keep in mind a few things.

You should never need to overlay any signature mapping variable. If you think you do, you are probably misunderstanding the technical workflow on how to set these variables dynamically.

You cannot map the same signature to the same form more than once. So for example you can’t have a SIG\_X\_PI tag reserved for the primary insured twice on any single PDF. You could do SIG\_X\_PI and SIG\_X\_PICOPY as an example.

You should always follow the best practice naming convention when mapping these signatures. See [HERE](#SigVariables) to the standard prefixes.

Your logic to the used party suffixes should follow the Data Dictionary as best you can. So PI, OI1, OWN1, etc. There may be times when you have the role and needs to sign the form in one place or another. Let’s say you have a HIV form where the PI has to initial Yes if they accept or No if they decline. In that case you’d have to create something like SIG\_I\_PIHIVYES and SIG\_I\_PIHIVNO. Notice how we still keep the standard prefix of SIG\_I, then the standard suffix of PI, then add something custom to the rest of the prefix. From a coding standpoint you could then set which tag is required to be initialed. Also ensure you are not using extra underscores in your naming. So SIG\_X\_PI\_HIVNO will not work.

# Workflows/Use Cases

## HIPAA Workflow

This workflow is not necessarily HIPAA specific, but is geared around the typical HIPAA use case. The first item you’ll need to understand is the client data variable of ESIG\_IsHIPAA. This variable controls two different calls in iGO.

1. View Forms, if ESIG\_IsHIPAA is True, the View Forms mechanism will trigger and display the forms currently triggered in iGO. If False, it will display all of the forms you’ve sent up to the eSignature vendor.
2. Create Package (Update Package), if True, when you call the GenerateDocuSign or GenerateDocuSignAsynch it will trigger an Update Package call.
   1. This call is used to add signers and document to an already existing signing package.

#### How This works

As a user of iGO, you’d fill out a few screens, then you’d “HIPAA Lock” the application (meaning you’ve locked down those specific screens). When locking you’d call a new server proc called GenerateDocuSignHIPAAAsynch. (You can find this server proc in the base level server procedures). The one item to call out in this procedure is that it sets a Session value of ESigGroup to HIPAA. (Note: You can copy this and change the name of the server proc and session value to manipulate your project with additional calls if need be)

*After this call is made it is expected that this signer or set of signers will be signing right away BEFORE any other update or append calls are made. If not, please follow the Append Package workflow to add to an existing signer.*

The next step is that the base process will find each form in the forms config file that has an ESigGroup set to the session value (ESigGroup) that has been set in the server proc. This results in an example of creating ROLE80 (PI HIPAA Signer), being able to sign only the HIPAA form, as that is the only form in the 9801 example that has an ESigGroup of HIPAA set.

Then the signing process is complete, it is suggested that you call the Server Procedure (RetrieveDocuSignDocuments). This will download the signed HIPAA form to iGO. (Note, the reason we can do this here, is we’ve sent up a very limited number of documents. If you were sending the entire application package for example, this call would be a significant performance hit). Then set the client data variable ESIG\_IsHIPAA to True. This now allows the View Forms process to render the forms within the iGO app, which thanks to the above call, will now also include the HIPAA form. If this variable is not set, the View Forms process will show ONLY the HIPAA form in this example.

At this point you can proceed to the rest of your iGO screens and fill out the info. You can then lock the application as you normally would. This lock will call the GenerateDocuSign or GenerateDocuSignAsynch Server Procedures, which will now call the update process because you’ve set the ESIG\_IsHIPAA value to True. However, once the next signer signs, or even on the load of the Signature Method screen, you’ll want to delete this value, or set it to False. The reason is, you now want to see all the forms on the eSignature vendor’s site, not iGOs.

At this point, you can continue with the rest of your signature process…. Unless………

## APPEND Package Workflow

This workflow is used to add documents to an existing signer. The process here is very similar to the HIPAA workflow, the difference is that the HIPAA workflow expects new signers and documents, while the Append workflow is adding documents to an existing signer.

#### HOW THIS WORKS

The best use case for this if for the agent. The agent may want to sign an attestation after other signers have signed. You’ll also want to map data to this form. So once the data has been captured normally, and before the agent signers, you’d call the Server Procedure AppendPackge or AppendPackageAsynch. This follows suit with the HIPAA workflow in the fact that it will look for an ESigGroup in the forms config as well. This Session value is set to Agent. So following this example it sends all the forms with ESigGroup, and have been triggered, up to the eSignature vendor, Agent-US in this sample.

After this call is complete, the agent can now sign, and the new form will be available to sign.

#### Overflow signatures on each page

Esignature can support a signature on each page. To enable this you need to log into Launch Darkly. Then choose igo as the project and QA as the environment from the dropdown in the upper left. Once you’ve done that, fine the key called esig-sign-all-remark-pages and click on the key. Under the Target individual users go to the “true” section and add a new user. Your user will be SUB\_[Your GAID]. For example, SUB\_9800. Then click save. You’re all set!

# Audit Reports

Audit reports are always pulled when a submission is triggered, if you don’t want this pulled, you can set BYPASS\_SAVING\_AUDIT\_XML\_DATA to True in your client data.

. The audit report contains the following:

* An AlphaTrust PDF per PDF sent and acted upon (Sign, Initial, Acknowledge).
* AlphaTrust XML saved into client data per PDF sent and acted upon
* User audit data that coresponds to the screen actions a user takes (Ex. AUDIT\_ROLE1\_Consent\_chkAgree and the time stamp as a value)

# Troubleshooting

There are multiple areas you can look within the eSignature. CES\Logs\CossEnterpriseSuite\_[date].log if any internal 500 errors are thrown or you don’t see any of the following info.

The browser Network tab for any client errors that may be thrown.

The DB by looking at the ESIG\_LastAPIResponse and the ESIG\_LastApiResponseFileName then looking into that file for the error message thrown back from the ESignature service.

You can also set the following key under your GAID

<Key Name="SAVE\_ESIG\_COMMON\_API\_REQUESTS\_AND\_RESPONSES">TRUE</Key>

And logs will be written to the dump folder per process.