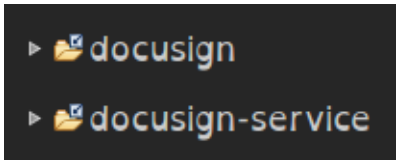
Docusign fio Integration Developer Guide

Table of Contents

Docusign integration components

The docusign integration consists of 2 components.



The **docusign component** is a spring mvc style webapp which consists of the docusign json models, both versions since the original system, with customise and send mode was implemented in the old and the send and sign mode was implemented in the new.

The docusign integration needed to be split in order to deal with tomcat classpath conficts.

The **docusign-service component** is a classic OT style webapp which consists of a controller.xml and various service side json event methods. Also the integration data model is defined in entitydef, there are several tables defined in the enity model along with a couple of views, descriptions of each can be found with the definition.

One table named *DynamicDocumentDownloadTenantCredentials* which is active in the main database might already exist, this mechanism is used in other components that work with documents which are created dynamically. It provides account credentials per tenant for dynamic doc downloads

The table that requires special mention is *DocuSignAuthenticatingUser*, this is the only other entity which is stored in the main database and works to associate a tenant with a docusign account. For testing google 'docusign developer sandbox' sign up, sign in, navagate to admin, choose integrator keys on the left menu, create a integrator key and then copy it into this table in order to enable docusign for a specific tenant, along with login details and *afterSendRedirectUrlBase* which is the base host name of the erp server **eg:** http://localhost:7611/ (this is used by the docusign api in callbacks pass control back to the erp after embedded signing is finished)

Also have a look at the properties in docusign/config/docusign.properties, comments explain properties in file (serverUrl would need to be changed with a move from the sandbox to production, the rest should be left as is)

How is the fio erp user identity provided and used by the integration?

By a call to a server side event located in the docusign-service component
`com.groupfio.docusign.events.DocusignUtilEvents.lookupUidInfo(HttpServletRequest, HttpServletResponse)`

The docusign integration js script import html tag has a 2 data attributes, set as the tenantId and loginUrlKey by the ftl.

This method, using the loginUrlKey or tenantId (it doesn't actually matter which) and the userLoginId as the input (this can also be gotten from the session in some cases unless it's a cross component request), gathers data about the logged in user to pass onto the docusign integration js script. Making it very simple to add a new docusign button to a page with, where loginUrlKey is assigned in the ftl from where ever it's avaiable.

Initialize Docusign On A Page

So to initialize docusign on a page we use the following in the target ftl

```
<div id="spinner"
style="position: fixed;  top: 25%;  left: 50%;z-index:1000"></div>
<script
data-loginUrlKey="${tenantKey?if_exists}"
data-userLoginId="${userLoginId?if_exists}"
src="/docusign/main/dst/docusign-bundle.js" type="text/javascript"></script>
```

Notice we also need to provide a for the spinner progress indicator to attach to.

Details such as tenantKey, userLoginId, name and primary email address for the logged in user, which is used to set up docusign users for the tenant account are provided by event endpoint /docusign-component/control/lookupUidInfo wired to the docusign-service event described above.

Sign Button Implementation

The docusign fio integration is implemented by importing a js script file into the ftl where the anchor (button) will appear. The Docusign object auto initializes and stores itself on the global window.ds, it also adds required css imports to the header on initalization.

In order to implement a clickable target in the context of the page, where data to build the target pdf is naturally avaiable an anchor html element is used with data attrributes used by the docusign script to execute the flow.

There dialogs are used to present setup forms or docusign api embbed views as appropriate. All data attributes except mime are required, mime defaults to application/pdf.
See the example bellow (random example comes from opentaps/warehouse/webapp/warehouse/shipping/submenus/picklistDetailsMenu.ftl):

```
<div class="subSectionHeader">
  <div class="subSectionTitle">${uiLabelMap.WarehousePicklistDetails}</div>
  <div class="subMenuBar">
    <a href="<@ofbizUrl>PicklistReport.pdf?picklistId=$
{picklistInfo.picklistId}</@ofbizUrl>" target="_blank" class="buttontext">$
{uiLabelMap.OpentapsContentType_ApplicationPDF}</a>
    <a data-dynamicDocUrl="<@ofbizUrl>PicklistReport.pdf?picklistId=$
{picklistInfo.picklistId}</@ofbizUrl>" data-title="picklist-$
{picklistInfo.picklistId}" data-emailBody="please sign asap" data-
```

```
emailSubject=”picklist to sign” data-mode="sns" class="signhere buttontext">
      Sign And Send
      </a>
    <#if isPicklistPicked?exists>
      <@submitFormLink form="closePicklistAction" class="subMenuButton"
text=uiLabelMap.WarehouseClosePicklists />
    </#if>
  </div>
</div>
```

In the example we can see how to place a sign and send button next to an existing pdf anchor button.

The dynamicDocUrl is the link (or post) which constructs the document sent to the docusign api and signed either via the custom flow, where tab's can be dragged and dropped onto the document manually and reciepients supplied manually or via the sign and send flow where the logged in user signs the document and then sends it on to a specific recipient.

Notice data attribute data-mode this is how we specify whether the 'customize and send' mode or the 'sign and send' mode should be used, with data-mode=”custom” or data-mode=”sns”.

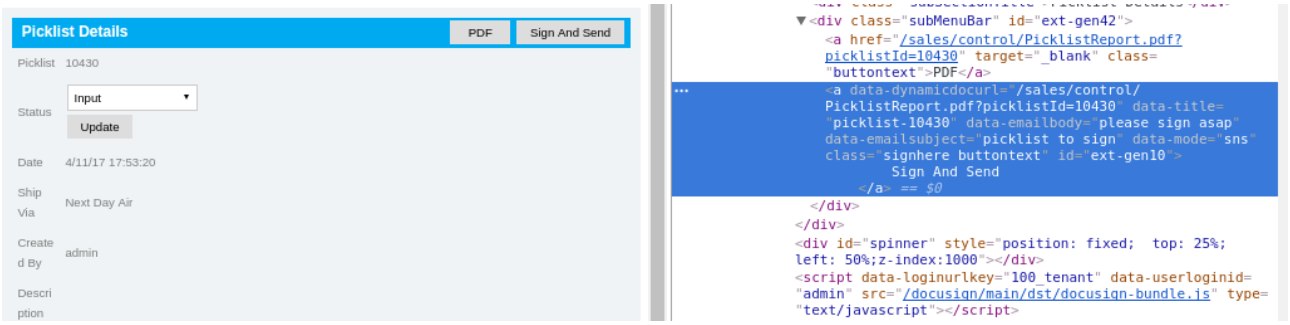
The docusign script opens a modal based flow when a click event occurs on the .signhere class, so having this class name on the anchor is also important.

The .signhere event handler uses this line to locate the correct anchor element (this is why only anchors can be used as buttons at this point)

```
var mydata = $(event.target).closest('a').data();
```

What this means is we can have multiple .sighere buttons on the page and the data atttributes will always to stripped from the anchor element closest to the click, removing the need to have to worry about unique ids for each anchor button.

Other classes can then be added to match the style of the adjacent button.



Providing the Dynamic Document URL

The dynamicDocUrl either needs to be provides as an element attriute named data-dynamicDocUrl where the value is a url with any query paramameters needed to create the document provided also, this is so the document download service can download a copy of the document and then send it to the docusign api.

Or alternatively it needs to be provided as an element attriute named data-dynamicDocFormName where the value is the name of an existing form with hidden inputs holding the details of the documents that needs to be created.

As a single url string which can just be provided as is, the <@ofbizUrl> macro even making it absolute...
data-dynamicDocUrl="<@ofbizUrl>invoice.pdf?invoiceId=${invoice.invoiceId}&reportId=FININVOICE&reportType=application/pdf</@ofbizUrl>"

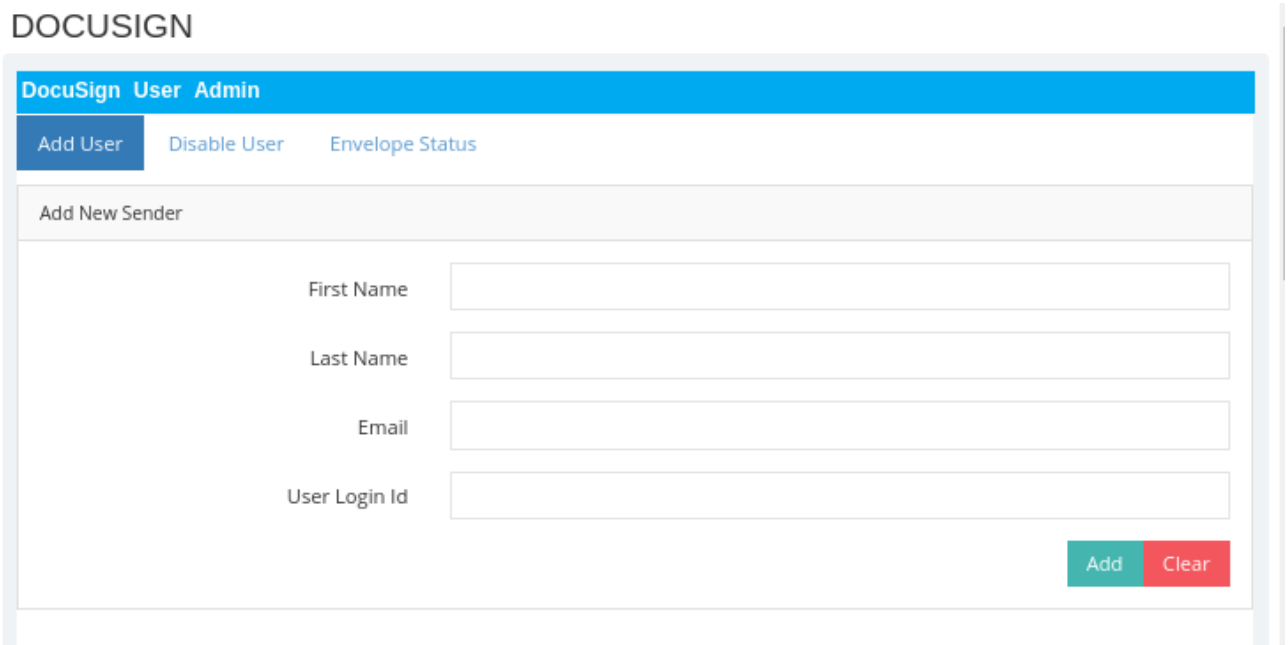
Or a form with hidden type inputs where the post request is created on submit, where the submit is often triggered by a js line somwhere...

```
<form method="get" action="order.pdf" name="orderPdfAction" target="_blank">
  <input type="hidden" name="reportType" value="application/pdf">
  <input type="hidden" name="reportId" value="PRUCHORDER">
  <input type="hidden" name="orderId" value="PO11540">
 </form>
```

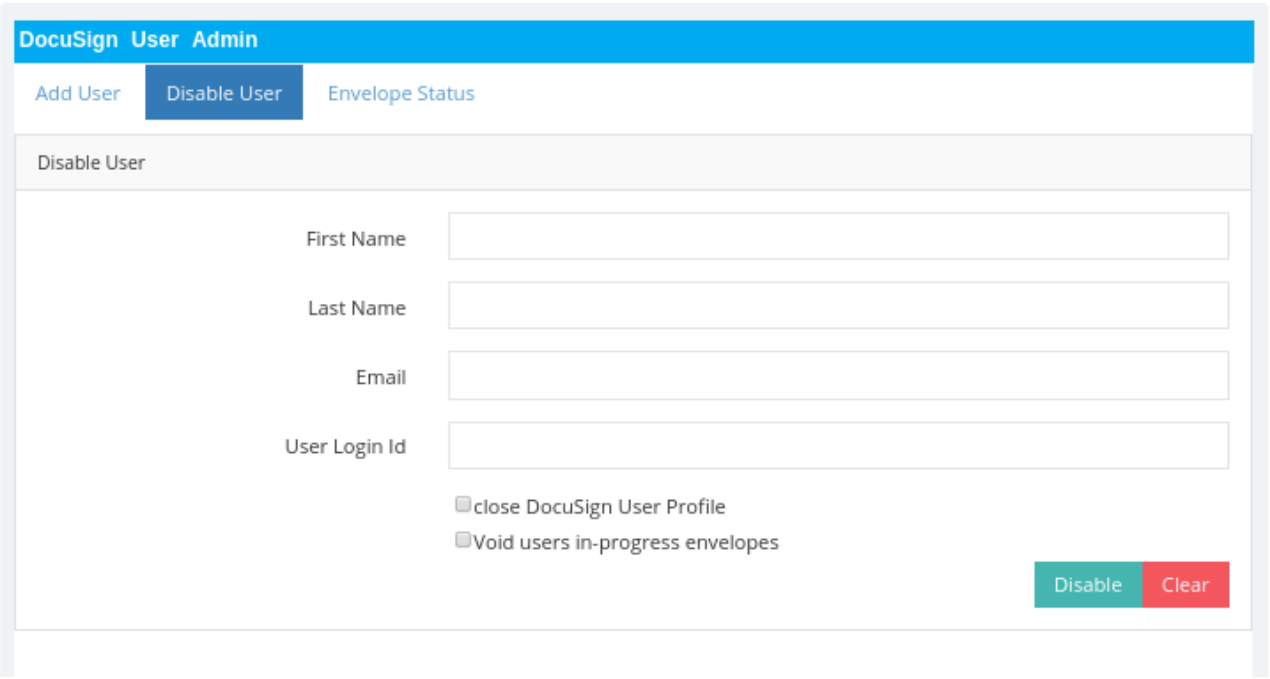
You would use data-dynamicDocFormName="orderPdfAction" and if you need a component url prefix you can use data-ofbizUrlPrefix eg: "/sales/control/". The docusign integration script then grabs the form element and builds the url from it. The data attribute ofbizUrlPrefix provides a way to prefix the built url in cases where this is required.

Docusign Admin Dashboard

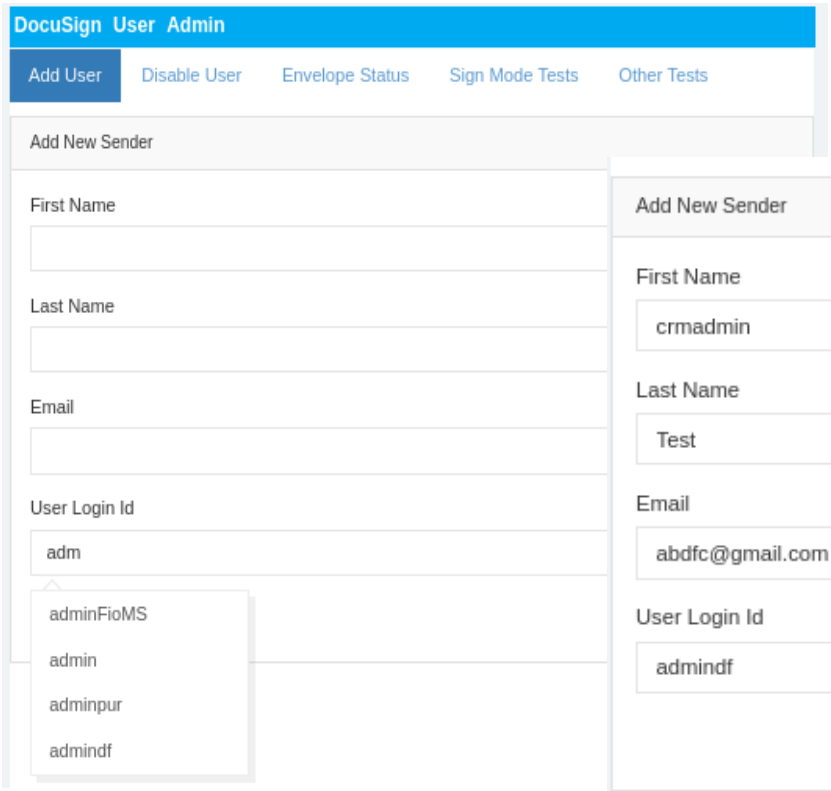
The docusign admin dashboard has 3 tabs



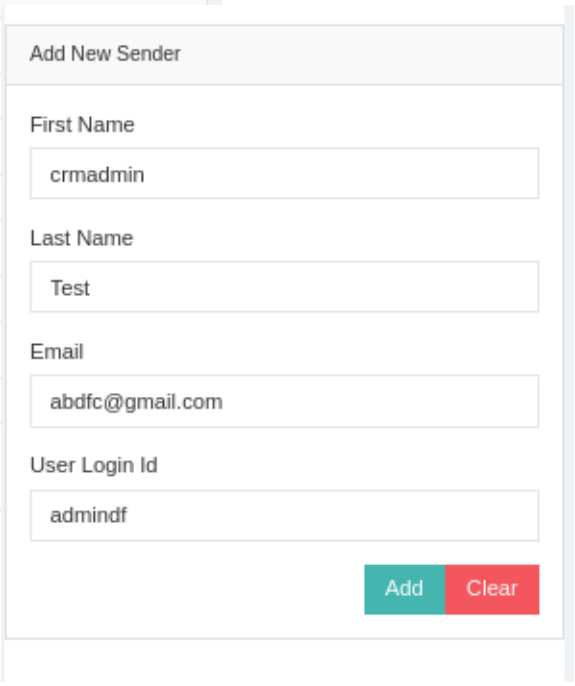
DOCUSIGN



Both Add User and Disable User have suggestions/autocomplete on each field, selection results in the other fields being populated automatically. ***NB: new docusign user needs to accept emailed invitation before they can use docusign with their login on FIO ERP.***
Also when debug is set to true extra testing tabs can be accessed.



Other fields autopopulated...



Lastly the Envelope Status Overview (server side paging table)

Envelope Status

Search

Showing 1 to 10 of 11 rows 10 rows per page

‹ 1 2 ›

user login	subject	envelope	changed	recipient name	status	signed	delivered	recipient name	status	signed	delivered
admin	Please sign asap - test 6	sent	5 Apr 17 16:08:01	FiO Test User	completed	5 Apr 17 16:08:01	5 Apr 17 16:07:46	Ben Seven	sent	-	-
admin	Please sign asap - test 5	sent	5 Apr 17 16:05:34	FiO Test User	completed	5 Apr 17 16:05:34	5 Apr 17 16:05:15	Ben Seven	sent	-	-
admin	Please sign asap - test 4	sent	5 Apr 17 13:27:38	Ben Seven	sent	-	-	Justin Robinsin	sent	-	-
admin	Please sign asap - cust	sent	5 Apr 17 13:30:45	Justin Robinsin	sent	-	-	Ben Seven	created	-	-
admin	Please sign asap - test 4	sent	5 Apr 17 13:22:51	FiO Test User	completed	5 Apr 17 13:22:51	5 Apr 17 13:22:42	Ben Seven	sent	-	-
admin	Please sign asap - test 4	sent	5 Apr 17 13:23:39	FiO Test User	completed	5 Apr 17 13:23:39	5 Apr 17 13:23:30	Ben Seven	sent	-	-
admin	Please sign asap - test 4	sent	5 Apr 17 13:25:34	Ben Seven	sent	-	-	Justin Robinsin	sent	-	-
admin	Please sign asap - test 4	sent	5 Apr 17 13:21:07	FiO Test User	completed	5 Apr 17 13:21:07	5 Apr 17 13:20:58	Ben Seven	sent	-	-
admin	Please sign asap - test 4 (custom)	sent	5 Apr 17 11:18:37	Ben Seven	sent	-	-	-	-	-	-
admin	Please sign asap - test 3	completed	5 Apr 17 09:37:39	FiO Test User	completed	5 Apr 17 08:57:31	5 Apr 17 08:57:16	Ben Seven	completed	5 Apr 17 09:37:39	5 Apr 17 09:15:04