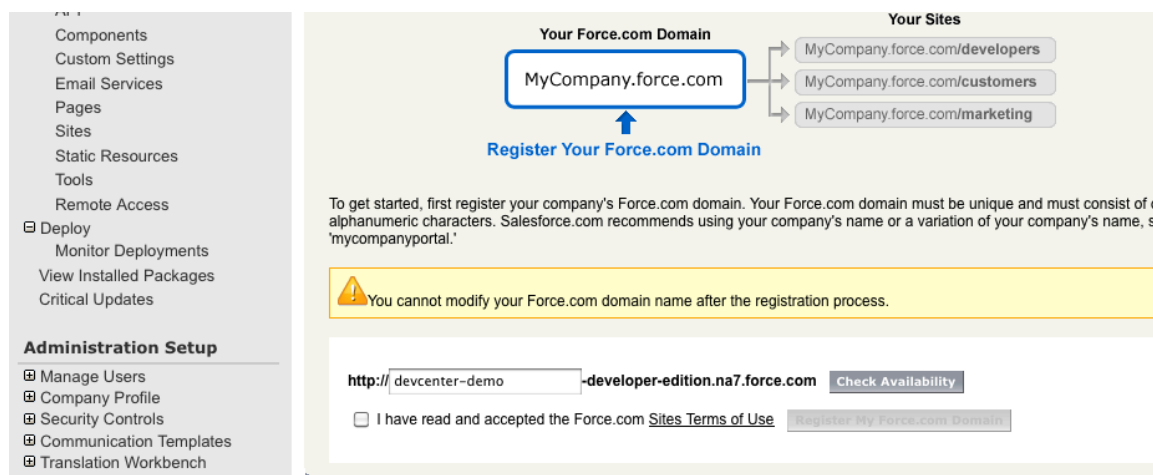


README: Accessing DocuSign API from Salesforce.com to send Contracts for eSignatures.

This readme provides a step-by-step example explaining how to set up a DocuSign ApexCode sample in your Salesforce.com account to send an object with a single click of a button. When you are done with this walkthrough you will be able to send a Salesforce.com Contract object for signing with just one button click.

Before getting started you need to get a free Salesforce.com developer account at <https://developer.force.com> and a free DocuSign developer account at www.docusign.com/devcenter.

1. Start out by adding DocuSign webservice to your authorized endpoints for your Salesforce.com developer account. To do this, go to Setup > Security > Remote Sites and add <https://demo.docusign.net/api/3.0/dsapi.asmx>.
2. Next, set up a Salesforce.com site so you can test rendering pages. There is plenty of documentation on how to get started with VisualForce, so we will leave it up to Salesforce.com to explain the best options there. For the purpose of this walkthrough you just need to have a site for testing. You can get to the screen below by clicking Setup > Develop > Sites.



From there you can create a site label. Once the site is created all you need to do is make it active.

salesforce.com  Setup • Mike Borozdin • System Log • Help • Logout  Sales

Home Campaigns Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products Reports Documents Dashboards

Personal Setup

- My Personal Information
- Email
- Import
- Desktop Integration
- My Chatter Settings *New!*

App Setup

- Customize
- Create
- Develop
 - Apex Classes
 - Apex Triggers
 - API
 - Components
 - Custom Settings
 - Email Services
 - Pages
 - Sites
 - Static Resources
 - Tools

Site Details
DevCenter Demo [Help for this Page](#)

« Back to List: Sites

Site Detail		Edit	Public Access Settings	Login Settings	Deactivate
Site Label	DevCenter Demo			Site Name	DevCenter_Demo
Site Description				Site Contact	Mike Borozdin
Default Web Address	http://devcenter-demo-developer-edition.na7.force.com/ [Preview as Admin]			Active	<input checked="" type="checkbox"/>
Secure Web Address	https://devcenter-demo-developer-edition.na7.force.com/ [Preview as Admin]			Login	Not Allowed
				Active Site Home Page	UnderConstruction [Preview]
Site Favorite Icon				Inactive Site Home Page	InMaintenance [Preview]
Site Robots.txt				Site Template	SiteTemplate [Preview]
Enable Feeds	<input type="checkbox"/>			Analytics Tracking Code	
URL Rewriter Class				Created By	Mike Borozdin, 8/25/2010 3:37 PM
Last Modified By	Mike Borozdin, 8/25/2010 3:38 PM				

Edit Public Access Settings Login Settings Deactivate



3. Now we need to create two pages; one for rendering a contract as a PDF and another that makes a web service call to DocuSign. We will start with the rendering page that we will call **RenderContract** (Note: the second page is set up in step 8). The code for the rendering page is fairly minimal. As you can see below, it just uses the standard Contract controller class and we are just rendering standard contract details with a couple of signature lines.

```
<apex:page renderAs="pdf" standardController="Contract">
  <apex:detail relatedList="true" title="true"/>
  <div style='clear:right;margin-top:50px'>
    <div style='float:right'>_____</div>
    <div style='float:right'>By:</div>
  </div>
  <div style='clear:right;margin-top:50px'>
    <div style='float:right'>_____</div>
    <div style='float:right'>Date Signed:</div>
  </div>
</apex:page>
```

The page needs to be added to the site to ensure that you can access it for testing. You can accomplish that by going to the site detail and selecting Site VisualForce Pages.

Site Visualforce Pages Edit	
Visualforce Page Name	AppExchange Package Name
BandwidthExceeded	
ChangePassword	
Exception	
FileNotFound	
ForgotPassword	
ForgotPasswordConfirm	
InMaintenance	
RenderContract	
SiteLogin	
SiteRegister	
SiteRegisterConfirm	
SiteTemplate	
Unauthorized	
UnderConstruction	

The other setting that has to be adjusted for testing is adding Contracts as something that is accessible from this site. To accomplish this go to Public Access Settings and edit the standard object permission.

	Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All 	Modify All 
Accounts	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Assets	<input type="checkbox"/>	<input type="checkbox"/>				
Campaigns	<input type="checkbox"/>	<input type="checkbox"/>				
Cases	<input type="checkbox"/>	<input type="checkbox"/>				
Contacts	<input type="checkbox"/>	<input type="checkbox"/>				
Contracts	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Documents	<input type="checkbox"/>	<input type="checkbox"/>				

- In order to test the first page and other code in our sample, we will create a test Contract. For simplicity we are going to put a contact in the Customer Signed By field.

At this point in time we have all the test data we need. You can note the Contract object ID in the URL

<https://na7.salesforce.com/800A00000000Q8oI>

And then use it to test the rendering page by plugging in the ID there:

<http://devcenter-demo-developer-edition.na7.force.com/apex/RenderContract?id=800A00000000Q8oI>

At this point you should be getting a PDF file.

5. In this step we are going to create a class to access the DocuSign API. For this walkthrough we are just going to get the sending WSDL, which can be found at <https://demo.docusign.net/api/3.0/Schema/dsapi-send.wsdl>. You should save the WSDL file to your desktop. (Note: For complete set of WSDL files please refer to documentation on DocuSign DevCenter.)

Next go to the Apex Classes and select "Generate from WSDL". When the class generator asks you to supply a class name, we suggest that you overwrite the class name with **DocuSignAPI**.

6. Using the new API endpoint, you can create a controller for the second page. In order to do this you need to go to Apex Classes again and create a new class. The code for the controller class is shown below:

```
public with sharing class SendToDocuSignController {
    private final Contract contract;

    public String envelopeId {get;set;}
    private String accountId = '';
    private String userId = '';
    private String password = '';
    private String integratorsKey = '';
    private String webServiceUrl
        = 'https://demo.docusign.net/api/3.0/dsapi.asmx';

    public SendToDocuSignController(ApexPages.StandardController controller)
    {
        this.contract = [select Id, CustomerSignedId, AccountId, ContractNumber
```

```

        from Contract where id = :controller.getRecord().Id];
        envelopeId = 'Not sent yet';

        SendNow();
    }

    public void SendNow()
    {
        DocuSignAPI.APIServiceSoap dsApiSend
            = new DocuSignAPI.APIServiceSoap();
        dsApiSend.endpoint_x = webServiceUrl;

        //Set Authentication
        String auth = '<DocuSignCredentials><Username>' + userId
            + '</Username><Password>' + password
            + '</Password><IntegratorKey>' + integratorsKey
            + '</IntegratorKey></DocuSignCredentials>';
        System.debug('Setting authentication to: ' + auth);

        dsApiSend.inputHttpHeaders_x = new Map<String, String>();
        dsApiSend.inputHttpHeaders_x.put('X-DocuSign-Authentication',
            auth);

        DocuSignAPI.Envelope envelope = new DocuSignAPI.Envelope();
        envelope.Subject = 'Please Sign this Contract: '
            + contract.ContractNumber;
        envelope.EmailBlurb = 'This is my new eSignature service,' +
            ' it allows me to get your signoff without having to fax, ' +
            'scan, retype, refile and wait forever';
        envelope.AccountId = accountId;

        // Render the contract
        System.debug('Rendering the contract');
        PageReference pageRef = new PageReference('/apex/RenderContract');
        pageRef.getParameters().put('id', contract.Id);
        Blob pdfBlob = pageRef.getContent();

        // Document
        DocuSignAPI.Document document = new DocuSignAPI.Document();
        document.ID = 1;
        document.pdfBytes = EncodingUtil.base64Encode(pdfBlob);
        document.Name = 'Contract';
        document.FileExtension = 'pdf';
        envelope.Documents = new DocuSignAPI.ArrayOfDocument();
        envelope.Documents.Document = new DocuSignAPI.Document[1];
        envelope.Documents.Document[0] = document;

        // Recipient
        System.debug('getting the contact');
        Contact contact = [SELECT email, FirstName, LastName
            from Contact where id = :contract.CustomerSignedId];

        DocuSignAPI.Recipient recipient = new DocuSignAPI.Recipient();
        recipient.ID = 1;
        recipient.Type_x = 'Signer';
        recipient.RoutingOrder = 1;
        recipient.Email = contact.Email;
        recipient.UserName = contact.FirstName + ' ' + contact.LastName;

        // This setting seems required or you see the error:
        // "The string ' ' is not a valid Boolean value.
        // at System.Xml.XmlConvert.ToBoolean(String s)"
    }
}

```

```

recipient.RequireIDLookup = false;

envelope.Recipients = new DocuSignAPI.ArrayOfRecipient();
envelope.Recipients.Recipient = new DocuSignAPI.Recipient[1];
envelope.Recipients.Recipient[0] = recipient;

// Tab
DocuSignAPI.Tab tab1 = new DocuSignAPI.Tab();
tab1.Type_x = 'SignHere';
tab1.RecipientID = 1;
tab1.DocumentID = 1;
tab1.AnchorTabItem = new DocuSignAPI.AnchorTab();
tab1.AnchorTabItem.AnchorTabString = 'By: ';

DocuSignAPI.Tab tab2 = new DocuSignAPI.Tab();
tab2.Type_x = 'DateSigned';
tab2.RecipientID = 1;
tab2.DocumentID = 1;
tab2.AnchorTabItem = new DocuSignAPI.AnchorTab();
tab2.AnchorTabItem.AnchorTabString = 'Date Signed: ';

envelope.Tabs = new DocuSignAPI.ArrayOfTab();
envelope.Tabs.Tab = new DocuSignAPI.Tab[2];
envelope.Tabs.Tab[0] = tab1;
envelope.Tabs.Tab[1] = tab2;

System.debug('Calling the API');
try {
    DocuSignAPI.EnvelopeStatus es
    = dsApiSend.CreateAndSendEnvelope(envelope);
    envelopeId = es.EnvelopeID;
} catch ( CalloutException e) {
    System.debug('Exception - ' + e );
    envelopeId = 'Exception - ' + e;
}
}
}

```

This might seem like a lot of code to understand, but the basics are pretty simple and you can see how a DocuSign envelope is constructed here: <http://wiki.github.com/docusign/DocuSign-eSignature-SDK/code-walkthrough-create-and-send-envelope>

7. In order to use DocuSign API, you need to retrieve your DocuSign API credentials. Go to <https://demo.docusign.net/>, sign into your demo account, and then go to Preferences > API. There you will find these values:

Integrator's Key: **ZORO-a81ec71a-cb17-4af0-b1aa-9513115cbf02**
 API UserName: **0e212ae6-1e12-40c1-8f5a-a57458ccaa63**
 API Password: **<your current password>**
 API Account ID: **736e7948-6861-4ef3-ae71-4c56603dc14f**

Plug these values into the controller class.

8. The next step is to create the second page, which actually calls SendToDocuSign. You will add this page to the site and hook it up to the custom controller we just developed:

```
<apex:page standardcontroller="Contract" extensions="SendToDocuSignController">
<h1>Your eSignature request is being sent to DocuSign API!</h1>
<hr/>
<apex:form >
<apex:commandButton value="Send Again!" action="{!SendNow}" />
</apex:form>
<hr/>
<strong>The DocuSign EnvelopeId:</strong>{!envelopeId}<br/>
</apex:page>
```

Now you should be able to test sending of the contract by calling the page with the same ID you used for testing the rendering page:

<http://devcenter-demo-developer-edition.na7.force.com/SendToDocuSign?id=800A00000000Q8oI>

9. The last step is creating a custom button on the contract object that calls the sending page. Go to Setup > Customize > Contracts > Buttons and Links and create a **New** button.

In the properties, select **Detail Page Button** and then add the URL with the Contract Id:

/apex/SendToDocuSign?id={!Contract.Id}

Add the button to the page layout for the Contract object. This lets your users click on the button to get their Contracts electronically signed! They don't need to learn any other systems or follow complicated steps. The logic behind the button does all the work.

The screenshot shows the Salesforce interface for a Contract record with ID 00000101. The record is in 'Draft' status. The 'Contract Detail' section includes fields for Contract Owner (Mike Borozdin), Contract Number (00000101), Account Name (1st Dev Account), Customer Signed By (Joe Signer), and various dates. A yellow arrow points to the 'One Click Send' button located in the top right of the contract detail section. Below the contract details is a 'Billing Address' section and an 'Items To Approve' section which currently shows 'No approval request specified'.