# **DocuSign**<sup>®</sup>

## Covered California streamlines the business of health insurance using DocuSign for Salesforce and the DocuSign eSignature

Covered California is the state-based marketplace where Californians can sign up for health insurance under the Affordable Care Act (ACA). As such, Covered California has a mission to ensure access and enrollment to ACAcompliant plans for all Californians seeking individual and family insurance when they do not receive those benefits through an employer.

Covered California works with 11 insurance companies that provide qualified health plans. To best serve Californians, Covered California engages more than 14,000 certified agents, brokers, and community partners that provide advice and enrollment assistance regarding qualified health plans to California residents. These professionals and agencies are collectively referred to as entities.

Before they can offer services to the public, each of these entities must have a signed agreement – which is reviewed and renewed on an annual basis – with Covered California. These services are governed by federal and state regulations, so processing and finalizing each agreement – which includes a signed agreement, supporting documentation, and certifications – requires many reviews and approvals by Covered California.

The business process for managing the entity agreements was inefficient, time-consuming, and costly. Covered California also was challenged by a lack of visibility into work. Each year, thousands of pages of paper were ushered through various divisions, including sales, finance, and legal – often by being walked from one office to another. Data extracted from the paper documents was manually keyed into spreadsheets, then uploaded into a database for review. The paper, ink, and staff required to process these agreements was significant, and the overall process slow.

Enter digital transformation and the incorporation of DocuSign into Covered California's business process. Covered California already used DocuSign to obtain electronic signatures from contracted agents at the end of the business process. Covered California sought a fully automated, digital solution that would reduce paper and ink, streamline paperwork, speed time to approval for agreements, provide security and confidentiality of entity information and signatures, and permit visibility of the agreement and process status to relevant parties. "The DocuSign digital solution is incredibly flexible. It allows us to include the necessary approvals and workgroups from many departments and to automate workflow. With DocuSign, our processes are much more secure, streamlined, and efficient."

Austin Walls-Barcellos

Project Manager - InformationTechnology Specialist I Covered California

## The Road to Optimal Health

Covered California had a use case for automated templates and electronic signatures. The use case indicated significant potential for automating the multi-stage approval process and electronic signatures. Since Covered California uses **Salesforce** to manage many parts of its business, it turned to **Zennify**, a leading expert in providing custom Salesforce solutions, especially those that integrate the DocuSign eSignature API.

Zennify focused on creating a positive user experience (UX), while also incorporating existing technologies used by Covered California: DocuSign and Salesforce. Covered California decided to implement the Salesforce Community Cloud, sometimes referred to as Salesforce Communities. Salesforce Communities enables companies to create a customer or partner portal that is custom branded. The Covered California portal in the Salesforce Community Cloud provides these benefits:

Enables community partners to use a fully automated process to initiate, sign, and complete their agreements to become Covered California-authorized entities.

Provides digital transformation of existing business processes using **DocuSign for Salesforce**, the **DocuSign eSignature API**, and **Salesforce Custom Buttons**.

Centralized, mobile-enabled website manages the entire process and provides two-way communication between Covered California and its entities.

While DocuSign for Salesforce provides a prebuilt integration that can be used for many business scenarios, Covered California needed a highly flexible integration with Salesforce Communities. To create this flexibility, Covered California extended DocuSign for Salesforce functionality by implementing a Salesforce custom button that calls the DocuSign eSignature API. This enabled Covered California to bridge the gap between Salesforce Communities and DocuSign. Behind the custom button functionality is a coding language called **APEX**, which is similar to Java. The entity agreement process involves providing key business information and is completed sequentially, as follows and as shown in Figure 1 (below):

Sections 1 – 5 – Entity provides key business information as shown in Figure 1 (below).

**Section 6** – Entity provides three types of documentation required by Covered California. This documentation can be provided in any order:

**Entity Agreement** – This step compiles all the data entered in sections 1 through 5 and generates an agreement document that is sent to the entity for eSignature. (See the **How It Works** section below for more details on the technical implementation.)

**Proof of Business Status Documentation** – Documentation that provides a Tax ID Number on letterhead to prove eligibility to be or remain an entity.

**Proof of Insurance** – Documentation that proves the entity carries a minimum of \$1,000,000 of liability insurance.

Section 7 - Qualifying Attestations - Series of five statements certifying eligibility.

The entity is prompted in each of the seven sections to either enter data or upload documents for review by Covered California, so those steps are not covered in this API success story. However, from a DocuSign perspective, the compelling part is when the entity clicks the **DocuSign CAE Agreement** custom button shown in Figure 1. This custom button was created within the **Salesforce Lightning Platform** to initiate the eSignature process for the Certified Application Entity (CAE) Agreement. For more information about how the custom button was implemented, see **How It Works** later in this API success story.



Figure 1. Salesforce Communities page showing completed data entry and custom button to send the agreement for eSignature.

Covered California created these **DocuSign templates**, each of which is used for a specific purpose:

#### **Certified Application Entity (CAE) Agreement**

#### Medi-Cal Managed Care Plan (MMCP) Agreement

The APEX code automatically selects the correct template based on the information the entity has provided in Sections 1 – 5. For example, DocuSign templates are determined by the type of entity (e.g., Certified Application Entity, Certified Enrollment Entity), how an entity provides services (e.g., at a private office, through a community clinic), and whether an entity is compensated by Covered California or by the plan-based provider.

After selecting the correct template (and looking up the corresponding Template ID), Covered California calls the DocuSign eSignature API to specify recipients and substituted Salesforce Communities field values (from Sections 1-5) for the preconfigured **DocuSign Tab** values in the selected template. The result is a recipient receiving the populated agreement document via email, which is shown in Figure 2.

		FINISH	OTHER ACTIONS +	
	Q Q ± = 0	D		
Doudign Envelope ID: 3850DCBA-EFE9-4092- STATE OF CALFORNIA STATE OF CALFORNIA ST2213 (New 5003) 1. This Agreement is entered into State address walk California Health Benefit Contractores walk Agreement is: 	ABP-13BC4045A26B between the State Agency and the Control Exchangethrough June 30,2018 t the terms and conditions of the followir	AGREEMENT NUMBER REDISTIVATION NUMBER actor named below:	ade a	
IN WITNESS WHEREOF, this Agroo	greement nent has been executed by the parties her CONTRACTOR (and antifer acceptation; particular), etc.)	eto. Cultivniu Dynamasu of General Ser Only	nico Ese	
BY (Auth Signature)	DATE SIGNED/D/ 11/9/2017	s not type)		Figure 2. Sample

After the DocuSign agreement is signed, the Salesforce Communities page shows a checkmark in the Entity Agreement area of Section 6. The supporting documentation (proof of business status, liability insurance, etc.) can be uploaded at any time by the entity, but all sections must be completed for Covered California to begin reviewing the documents. All signed and uploaded documents are stored in the Salesforce Communities record for the entity (displayed as **Account**) in the **Required Documentation** section, as shown in Figure 3.

Searc	ch Search	
♠ MYENTITY COUNSEL	ORS COVEREDCA.COM CONTACT SUPPORT	
Account Acme Inc. (TEST)	+	Follow Edit
Program Type Certified Application Entity	Phone Website Primary Location Address 1234 Test Ave., Sacramento, CA, 95834	
Entity Information	Required Documentation (1)	New
Account Name	Entity Status REQUIRE FILE TYPE STATUS EXPIRA	TION
Acme Inc. (TEST)	Draft CAEAgre CAEAgree DocuSign P	-
Parent Account	Program Type Certified Application Entity	View All

**Figure 3.** Salesforce Communities showing which required documents have been received.

As shown in Figure 4, when each document is signed or uploaded, it is added to the Salesforce Communities record, the corresponding Salesforce record, and the **Notes & Attachments** section.

COVERED.	Q Search Sal	esforce		★ • ■ ? 犂 ♣	Ö
Covered CA EAPA	Home Ac	counts 🗸 Entity Application	s 🗸 Contacts	✔ Locations ✔ Reports ✔ Bugs ✔ More	• •
Required Documentati CAEAgreement	on			Edit Delete Clone	•
Required Doc Name CAEAgreement		Entity		Notes & Attachments (2) Envelope 1a747c32-5e74-48e5-bd7b-a7746bd	▼ d
File Type  CAEAgreement		File Details	1	<ul> <li>Aug 28, 2017 - Attachment</li> <li>Envelope 1a747c32-5e74-48e5-bd7b-a7746bd</li> <li>Aug 28, 2017 - Attachment</li> </ul>	d
Status DocuSign Pending		Expiration Date 0			
✓ Analyst					
Analyst Assigned				View All	
Analyst Assigned Date				Recipient Status (3+)	•
Analyst Comments		Accounts V Entity Applications V Contacts V Locations V Reports V Bugs V More V Edit Delete Clone Entity File Details • Expiration Date • DocuSign Status Sent Completed Last Modified By View All View All			
✓ DocuSign Related Fi	elds			Molly	¥
DocuSign Envelope Id 1A747C32-5E74-48E5-BD7 A7746BDD7654	7B- /	DocuSign Status Sent		Docusign Re Docusign Ro 3 Recipient St Created	¥
DocuSign Sent 8/28/2017 3:29 PM		Completed		Docusign Re 2 Recipient St Sent	
Created By 8/28/2017 3:29 PM		Last Modified By 8/28/2017 3:31 PM		View All	

**Figure 4.** Salesforce record showing received documents and application status.

Figure 4 contains detailed information that helps Covered California understand what the status of a given application is in the overall process. Some key points are that Figure 4 shows the overall DocuSign status as **Pending**, indicating that the review process is not yet complete. The **Notes & Attachments** section shows a list of all DocuSign documents that are complete (signed), as well as all uploaded documents. The **Recipient Status** section shows the status of each recipient's signature on the envelope. The recipients are listed in reverse signing order with the earliest recipient on the bottom and the most recent recipient on top. Each recipient will have a recipient status of **Created** until the envelope is routed to a given recipient. When the envelope is routed to a specific recipient, it will have a recipient status of **Sent** and when the recipient shave signed the agreement, the overall **DocuSign Status** field is changed to **Complete** and the **Completed** field will reflect the corresponding date.

### **How It Works**

To extend the functionality of DocuSign for Salesforce, an APEX API wrapper was created around the DocuSign eSignature API that is called when the entity clicks the custom button shown in Figure 1. Clicking the custom button initiates the agreement review process between Covered California and its entities.

The APEX code, shown in Figure 5, performs the following functions:

Selects the correct template based on the criteria entered by the entity filling out the agreement.

Calls the DocuSign API to **create the envelope** using the selected template ID. The template is pre-created by Covered California with the correct recipients and routing order, but some values are substituted at run-time based on user input into the various Salesforce Communities sections. The values include entity name, email address, etc. Standard DocuSign functionality handles the routing based on selected routing order.

Checks DocuSign envelope status by implementing a **DocuSign Connect webhook**. This enables a custom web listener to receive notifications when any status change is made to an envelope. Therefore, the Salesforce status is updated when any recipient in the routing order signs the agreement, and also when the envelope status is complete (all recipients have signed).

Checks whether additional required documentation has been uploaded. This is not a DocuSign piece, but is required as part of the overall process. When all required documentation has been uploaded and the agreement has been signed, Salesforce notifies Covered California so they can assign an analyst to review all documents.

<pre>DeckCompacture.up = Concerning = Compacture.up = Compactu</pre>	Lonpoe	Fire can navigate Search Project Rull W	less (slasses (Dess) (Circo)		inen Alennel		9
<pre>PocuSignRecipient CALS &amp; restEffection Long &amp;</pre>	•	Force.com -	(src/classes/DocuSignRi	ecipient.cls - Eci	ipse - /Users/	Invorkspace	
<pre>SocuSignRecipient.ds II  exateRecipient.doo.ds  exateRecipient.</pre>	리 왜 😂 🕫	💊 🗶 🖹 🗳 🚳 🖓 🏇 र 🔾 र 🏊 र 🗁 🔗 र 🖗 र 🤞 र	• <			Q Quick Access 🛛 🖸 🔀 Force.com 🛃 Jazz Administration 💿	Worl
<pre>public with sharing class DocuSignRecipient {     #uroEnabled     Public kiths haring class DocuSignRecipient {     #uroEnabled     Public kiths Haring class DocuSignRecipient {         System.debug('docuSignMp ====&gt;'+docuSignMpp);         User oLirrentUser = [select Id, Nome, email, Tile from User where id=:docuSignMpp.get('userId')];         MppString, Strings mopToReturn = new MppString, String&gt; docuSignMpp.get('userId')];         MppToReturn.put('sel', ');         mpToReturn.put('sel', ');         mpToReturn.put('sel', ');         string signer.comelser.loa;         string signer.comelser.loa;         string signer.comelser.loa;         string signer.comelser.loa;         string signer.comelser.loa;         string signer.comelser.loa;         string signer.coleMem;         string signer.coleMem;         string signer.coleMem;         string signer.coleMem;         string docuSign.comelser.loa;         string signer.coleMem;         string docuSignExplote = DocuSignSetting_cc.getValues('DocuSignSetting');         if(DocuSignTempDate = null )         {             mpToReturns, ut('fron', 'No DocuSign Settings Found');             return mpToReturns;         }         DocuSignTempDate _Dota_c DocuSignTempDate = [select id, nome,Object_API_Nome_c, Role_Nome_c, Template_Idc, Text_Tabs_with_API_Nome_c, Save_to_Object_API_c, Parent_Field         if(DocuSignTempDate = null )         // mpToReturn;         Npp -String, Strings template Found');         return mpToReturn;         Npp -String, Strings S</pre>	🗶 DocuSignR	ecipient.cls 🕱 🗷 createRecipientJson.cls 🛛 🗶 createB	nvelopeJson.cls 🛛 🗶 clsDo	cuSignUtility.cls			. 4
<pre>public with sharing class DocuSignReciptent {     #uroInabled     Fublic static Map.String, String&gt; processDocuSignViem(Map.String&gt; docuSignMap){     System.debug('docuSignMap):     User GurrentUser = [salect 14, Mane, email. Title from User where id=:docuSignMap.get('userId')];     Map.String, string- mopDoleturn = new Map.String, String&gt; (j);     mapToReturn.put('Error', ');     mapToReturn.put('Urror', 'No DocuSign.Setting, etc''neturnRL');     string docuSignConfigSettings = DocuSign.Setting_c.getValues('DocuSignSetting');     if(DocuSignConfigSettings = null)     forter.m.put('Error', 'No DocuSign Settings Found');     return mapToReturn;     forter.m.put('Error', 'No DocuSign Settings Found');     return mapToReturn;     forting stepre.exale.docuSignMap.get('neturnRL');     string docuSignConfigSettings = DocuSign.Setting_c.getValues('DocuSignSetting');     if(DocuSignConfigSettings = null)     fortion = null ('focuSignConfigSettings = DocuSign Setting_c.getValues('DocuSignSetting');     if(CocuSignConfigSettings = null)     fortion = null ('focuSignPepDate = [select id, nome,Object_API_Name_c, Nole_Name_c, Template_Idc, Text_Tobs_with_API_Name_c, Sove_to_Object_API_c, Parent_Field     if(OocuSignTempDate = null )     map Ostion = null )     map Ostion = null ('focuSignTempDate = 1exlect id, nome,Object_API_Name_c, Nole_Name_c, Template_Idc, Text_Tobs_with_API_Name_c, Sove_to_Object_API_c, Parent_Field     if(OocuSignTempDate = null )     map Ostion_nut('Error', 'No DocuSign Template Found');     return mapToReturn;     }     Map ostring, String, Strings terms of String, String-('interviend');     return mapToReturn;     }     Map ostring, String, Strings Template Found');     return mapToReturn;     }     Map ostring, String, String, String, String-('interviend');     return mapToReturn;     }     Map ostring, String, String, String, String-('interviend');     return mapToReturn;     Map ostring, String, String, String, String-('interviend');     return mapToReturn;     Map ostring, String, Stri</pre>	1.11						
<pre>Public Wich Summing Closs DucusignMeetplein {     #urdinabled     #urdinabled     #urdinabled     Fullic static MapString, Strings processBocuSignViem(MapString, Strings dacuSignMap.jet('userId')];     User oLimentUser = [select Id, Name, email, Tille from User where id=:docuSignMap.get('userId')];     User oLimentUser = [select Id, Name, email, Tille from User where id=:docuSignMap.get('userId')];     MapString, Strings motification = new MapString, Strings-O;     mapToNeturn.put('Ise', ',');     string signer.real = oCurrentUser.Nam;     string offecturn.put('Iron', 'No DocuSign.Setting.c.getValues('DocuSignSetting');     if(DocuSignTeepDate = null )     moToNeturn;     preturn spToNeturn;     fortowSignTeepDate = null )     //    moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;     // moToNeturn;</pre>	8	e with charing class DecuSicsBeciniant (					
<pre>#urclabled Public static MpoString, String&gt; processBocuSignVien(MpoString, String&gt; docuSignMup)[  System.debug('docuSignMup =========''+'.docuSignMup);  User CurrentUser = [select d+'.Mone, menil, Title from User where id=:docuSignMup.get('userId')];  Map-String, String=mpOROturn = new MopString, String&gt;(); mpToReturn.put('Error', '); mpToReturn.put('User', '); string sign=r.mome = CurrentUser.emeil; string sign=r.mome = CoursignConfigSettings = DocuSignSettingc.getValues('DocuSignSetting'); i(fOcuSignSetToBeturn; } former_rentUserDeturn; formutUserDeturn; formutUserDetu</pre>	9 publi	c with sharing class bocasignkeciptent (					
<pre>Public Static MpcString, String&gt; processDocuSignVien(MopeString, String&gt; docuSignMop){</pre>	11						
<pre>Public static MapString, String&gt; processDouSignVien(Map-String, String&gt; docuSignMap){     System.dbug('docUSignMap);     MapString, String&gt; reprocessDouSignVien(Map-String, String&gt; docuSignMap,get('userId')];     MapString, String&gt; reprotection = new MapString, String&gt;();     mapToReturn.put('Error', ');     mapToReturn.put('Irror', 'No DocUSignSetting_ c.getValues('DocuSignSetting');     string docuSignConfigSettings = DocuSign_Setting_c.getValues('DocuSignSetting');     if('DocuSignConfigSettings = mull )     fortor', 'No DocUSign Settings = DocuSign Setting_s C.getValues('DocuSignSetting');     if('DocuSignConfigSettings = mull )     fortor', 'No DocUSign Settings Found');     return mapToReturn;     fortor', 'No DocUSign Settings Found');     fortor', 'No DocUSign Settings Found');     fortor', 'No DocUSign Settings Found');     return mapToReturn;     fortor', 'No DocUSign Settings Found');     fortor', 'No DocUSign Settings Found');     fortor', 'No DocUSign TempDate = [select id, name,Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_to_Object_API_c, Parent_Field     if(CocUSignTempDate = mull )     fortor', 'No DocUSign Template Found');     return mapToReturn;     fortor', 'No DocUSign Template = [select id, name,Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_to_Object_API_c, Parent_Field     if(CocUSignTempDate = mull )     fortor mapToReturn;     fortor mapToReturn;</pre>	12 @	AuraEnabled					
<pre>System.dbug('docudignNup ====s'+docudignNup); User Guter</pre>	13 P	ublic static Map <string, string=""> processDocuSignV</string,>	iew(Map <string, string=""> a</string,>	docuSignMap){			
<pre>System.dbug('douGignMpg'douGignMpg): Description:</pre>	14						
<pre>System.debug('docUs(gnMop&gt;'+docUs(gnMop); User cCurrentUser [select 14, Mews, email.] Title from User where [d=:docuSignMop.get('userId')]; MeprOstime, put('emp', ''); meprOsterum, put('Use', ''); meprOsterum, put('Ise', ''); meprOsterum</pre>	15						
<pre>bude outpretuber = [statch is, nome, deadL, fitte from UseP name: i=:docUsignMop.get('useris'); mepToReturn.put('Error', '); mepToReturn.put('Error', '); mepToReturn.put('Error', '); string signer.tile = ofurentUser.email; string signer.tile = ofurentUser.tile; string signer.role = ofurentUser.tile; string signer.role = ofurentUser.tile; string signer.role = ofusignMop.get('userid'); string signer.roleName; string totUser.id = occusignSetting.c.getValues('DocuSignSetting'); i(fOocuSignConfigSettings = null) f nefOrdeturn.put('Error', 'No DocuSign Settings Found'); return nepToReturn; f pocusignTempDate = null) f nefOrdeturn.put('Error', 'No DocuSignTempDate = [select id, name_Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_to_Object_API_c, Parent_Field i(fOocuSignTempDate = null) f meturn nepToReturn; put for the option = null) f meturn nepToReturn; put for the option = null) f meturn nepToReturn; put for the option = null) f meturn nepToReturn; meturn nepToReturn; f meturn nepToReturn; meturn nepToReturn; f meturn nepToReturn; f me</pre>	16	System.debug('docuSignMap>'+docuSignMap)	1				
<pre>http://www.communication.</pre>	17	User ocurrentUser = [select 1d, Name, email, I	itle from User where id=:	aocusignMap.ge	et('userid')];		
<pre>motoSeturn.put('trop', '); motoSeturn.put('trop', '); motoSeturn.put('trop', '); motoSeturn.put('trop', 'No DocuSignTemplate Found'); string signer_times - docuSignTemplate - Seturn Seturn string signer_times - docuSignTemplate - Seturn string signer_times - docuSignTemplate - Seturn string signer_times - docuSignTemplate - Seturn string string seturn.put - docuSignTemplate - Seturn motoSeturn.put - Seturn string seturn seturn string seturn seturn string seturn seturn string seturn</pre>	10	mapToReturn nut('msa', '');	ng, scrings(),				
<pre>mcpToRturn.put('URL', ''); string docusign.terplate_ld; string signer_mome = ofurrentUser.email; string signer_mome = of consignMop.get('uterrURL'); string signer_most_= of docusignMop.get('uterrURL'); string signer_most_id = docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string signer_uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string docusign.get('uterrURL'); string signer_uterrURL'); string signer_uterrURL'); string signer_uterrURL'); string signer_uterrURL'); string signer_uterrURL'); string signer_uterrURL'; string signer_uterrURL'; string signer_uterrURL'; string signer_uterrURL'); string signer_uterrURL'; string s</pre>	20	mapToReturn.put('Error', ');					
<pre>string docusign.template_ld; string signer_email = of_remetilser_email; string signer_enable = of_remetilser.email; string signer_enables = of_remetilser.email; string signer_user_id = docuSignMop.get(returnURL'); string signer_user_devices string string motions string string motions string string motions string string motions string string motions string string string motions string string string motions string motions string string motions string string motions string string string motions string string motions string string motions string string motions string string motions string string motions string motions string string motions string string motions string string motions string string motions string string string motions string string strin</pre>	21	mapToReturn.put('URL', '');					
<pre>string docusign_teeplote.jd; string sign=r_mome = ofurrentUser.email; string sign=r_mome = ofurrentUser.id= string sign=r_mome = ofurrentUser.id= string sign=r_user.id= docuSignMpp.get('userId'); string sign=r_user.id= docuSignMpp.get('userId'); string sign=r_user.id= docuSignMpp.get('userId'); string sign=r_user.id= docuSignMpp.get('userId'); string sign=r_user.id= docuSignMpp.get('userId'); string sign=r_user.id= docuSignConfigSettings = DocuSign_Settingc.getValues('DocuSignSetting'); if(CocuSign_Settingc DocuSignConfigSettings = DocuSign_Setting_c.getValues('DocuSignSetting'); if(CocuSignConfigSettings = mull) apToReturn.put('irror', 'No DocuSign Settings Found'); return mptToReturn; } DocuSignTempDate = mull) f mpToReturn.put('irror', 'No DocuSignTempDate = [select id, name_Object_API_Namec, Role_Namec, Template_Idc, Text_Tabs_with_API_Namec, Save_to_Object_APIc, Parent_Field if(CocuSignTempDate = null) f mpToReturn.put('irror', 'No DocuSignTempDate = [select found'); return mptToReturn; } d f mpToReturn.put('irror', 'No DocuSignTempDate = new MprString, String&gt;(); mpToReturn.put('irror', 'No DocuSignTempDate Found'); return mptToReturn; } f f f f f f f f f f f f</pre>	22						
<pre>string signer_email = oLurentUser.email; string signer_meme = oCurrentUser.Neme; string signer_neme = oCurrentUser.Neme; string signer_neme = oCurrentUser.Neme; string signer_neme string signer_neme string signer_neme string viewBit; string viewBit; string viewBit; string docuSignTempDb; bocuSign_textTodSWithAplKeme; string docuSignTempDb;</pre>	23	string docusign_template_id;					
<pre>string signer_nome = OurrentUser.Nome; string signer_nome. = OurrentUser.Titis: string signer_nome.Titis: string signer_nome.Titis: string signer_nome.User.Id = docuSignMop.get('userId'); string string string string string string string string string string string string string string string.c.getValues('DocuSignSetting'); if(DocuSign_Setting</pre>	24	<pre>string signer_email = oCurrentUser.email;</pre>					
<pre>string sign=r.trie = ourmentuser.ittie =</pre>	25	string signer_name = oCurrentUser.Name;					
<pre>string signer_user.id = douSignMepr.get('userId'); string textIbaSitthAptIone; string viewId; DouSign_Settingc_DouSignConfigSettings = DouSign_Settingc.getValues('DouSignSetting'); if(CouSignConfigSetting = mull ) if nepToReturn.put('irror', 'No DouSign Settings Found'); return mepToReturn; } DouSign_Template_Dota_c DouSignTempDate = [select id, name,Object_API_Name_c, Role_Name_c, Template_Idc, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field if(CouSignTempDate = mull ) if nepToReturn.put('irror', 'No DouSign Template Found'); return mepToReturn; } DouSignTempDate = new MepAstring, String&gt;(); } Mep Astring, String mepRedeff = new MepAstring, String&gt;(); </pre>	26	string signer_title = oturrentuser.litle;	tumpli RL 1 N				
<pre>string signer_roleWame; string string inveloperation string ovelopeId; string viewaRk; string divelopWithpulkame; string div</pre>	28	string signer user id = docuSignMap.get('userI	d'):				
<pre>string envilopEld; string viewEld; string viewEld; string viewEld; string textTobsWithApiNome; string docs[infemDb]; bocusign_Settingc_DocuSignConfigSettings = DocuSign_Setting_c.getValues('DocuSignSetting'); if(CocuSignConfigSettings == mull) impToReturn.put('Error', 'No DocuSign Settings Found'); return mpToReturn; bocusign_Template_Data_c DocuSignTempData = [select id, name_Object_API_Name_c, Role_Name_c, Template_Idc, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field if(CocuSignTempData == mull) four particular in the DocuSign Template Found'); return mpToReturn; } def mpToReturn.put('Error', 'No DocuSign Template Found'); return mpToReturn; } def mpToReturn; four mpToReturn; } def mpToReturn; four mpToReturn; } def mpToReturn; four mpToReturn; } def mpToReturn; four m</pre>	29	string signer_roleName:	- )1				
<pre>string viewURL; string textDoWithapName; string docuSignTempDb; string docuSignTempDb; docuSignTempDateLings = null ) f ngpToReturn_put('Error', 'No DocuSign Settings Found'); return mgpToReturn; dupTopLate_Data_c DocuSignTempData = [select id, name,Object_API_Name_c, Role_Name_c, TempLate_Id_c, Text_Tobs_with_API_Name_c, Sove_to_Object_API_c, Parent_Field t('DocuSignTempData == null) dupTopLate_nut('Error', 'No DocuSignTempData = [select id, name,Object_API_Name_c, Role_Name_c, TempLate_Id_c, Text_Tobs_with_API_Name_c, Sove_to_Object_API_c, Parent_Field t('DocuSignTempData == null) f ngpToReturn_mapToReturn; hep_String_String_NameEcder = new Map<string, string="">(); dup_string_String_NameEcder = new Map<string, string="">(); f new String_String_NameEcder = new Map<string_string>(); f new String_String_NameEcder = new Map<string_string_string>(); f new String_String_NameEcder = new Map<string_string>(); f new String_String_String_String_String_String&gt;(); f new String_S</string_string></string_string></string_string></string_string></string_string></string_string></string_string></string_string_string></string_string></string,></string,></string,></string,></string,></string,></string,></string,></pre>	30	string envelopeId;					
<pre>string textTabsWithApIName; string textTabsWithApINapIName; string textTabsWithApIName; string textTabsWithAp</pre>	31	string viewURL;					
<pre>string docuSignTempOb; stringcDocuSignConfigSettings = DocuSign_Settingc.getValues('DocuSignSetting'); if(DocuSignConfigSettings = null) f nopToReturn.gut('Error', 'No DocuSign Settings Found'); return mapToReturn; DocuSign_Template_Data_c DocuSignTempData = [select id, name_Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field if(DocuSignTempData = null) f nopToReturn napToReturn;</pre>	32	string textTabsWithApiName;					
<pre>Bit DocuSign_Settingc DocuSignConfigSettings = DocuSign_Settingc.getValues('DocuSignSetting'); i('CocuSignConfigSettings = null ) { f nepToReturn.put('Error', 'No DocuSign Settings Found'); return mepToReturn); d Docusign_TempTote_Dota_c DocuSignTempDota = [select id, name_Object_API_Namec, Role_Namec, TempTote_Idc, Text_Tobs_with_API_Namec, Save_to_Object_API_c, Parent_Field i('CocuSignTempTote_Dota_c DocuSignTempDota = [select id, name,Object_API_Namec, Role_Namec, TempTote_Idc, Text_Tobs_with_API_Namec, Save_to_Object_API_c, Parent_Field i('CocuSignTempTote_Dota_c DocuSignTempTote = [select id, name,Object_API_Namec, Role_Namec, TempTote_Idc, Text_Tobs_with_API_Namec, Save_to_Object_API_c, Parent_Field i('CocuSignTempTote = null) } } Mop -String, Strings mapHeader = new Map<string, string="">(); } } Description = DocuSignTempTote = new Map<string, string="">(); } } } } } } } } } } } } } } } } } } }</string,></string,></pre>	33	string docuSignTempObj;					
<pre>BocuSign_ConfigSettings = DocuSign_Settings = DocuSign_Settingc.getValues('DocuSignSetting'); if(DocuSignConfigSettings = null) if nopToReturn_sqt'Error', 'No DocuSign Settings Found'); return mapToReturn; } Docusign_Template_Data_c DocuSignTempData = [select id, name_Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field if(DocuSignTempData = null) if appToReturn_sqt'Error', 'No DocuSignTemplate Found'); return mapToReturn; } Map <strings_trings_trings_mapreturn; <="" pre="" }=""></strings_trings_trings_mapreturn;></pre>	34						
<pre>if(DocuSignConfigSettings == null) if(DocuSignConfigSettings == null) if(DocuSignConfigSettings == null) if(DocuSignTerplate_Data_c DocuSignTerpData = [select id, name,Object_API_Name_c, Role_Name_c, Tamplate_Id_c, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field if(CocuSignTerpData == null) if(DocuSignTerpData == n</pre>	35	DecuSion Setting DecuSionConficSettings - D	oculion Satting c cotVal		atting 1)		
<pre>status getuins getuins getuins getuins found'); neturn mapToReturn; } function for the status for the stat</pre>	37	if(DocuSignConfigSettings == null )	ocustgn_secctingc.gecva	tuest bocustign.	eccing ),		
<pre>impToReturn.put('Error', 'No DocuSign Settings Found');     return mpToReturn;     return mpToReturn;     return mpToReturn;     return mpToReturn;     return mpToReturn;     return mpToReturn;</pre>	38	{					
<pre>40 return mapToReturn; 41 } 42 43 44 44 45 45 46 46 46 47 47 47 47 47 47 47 47 48 47 48 48 48 49 49 49 49 49 49 49 49 49 49 49 49 49</pre>	39	mapToReturn.put('Error', 'No DocuSign Sett	ings Found');				
<pre>41 } 42 43 Docusign_Template_Data_c DocuSignTempData = [select id, name,Object_API_Name_c, Role_Name_c, Template_Id_c, Text_Tabs_with_API_Name_c, Save_ta_Object_API_c, Parent_Field 45 (f(DocuSignTempData == null) 45 { 46 naploReturn.put('Error', 'No DocuSign Template Found'); 47 return maploReturn; 48 } 49 Map <string. map<string,="" mapheader="new" string="" string.="">(); 49 Amp <string. map<string.="" napleader="new" string="" string.="">(); 40 Amp <string. string="" string.="">(); 41 Amp </string.></string.></string.></pre>	40	return mapToReturn;					
<pre>42 43 44 44 45 46 46 47 47 47 47 47 47 47 47 47 48 47 47 47 48 47 47 48 48 49 49 49 49 49 49 49 49 49 49 49 49 49</pre>	41	}					
<pre>43 UOCUSIgn:emplote_uota_c uota_sign:emplota = [select to, nome,uoject_AFL_nome_c, Role_Nome_c, Templote_Id_c, Text_Tabs_with_AFL_Nome_c, Save_to_Object_AFL_c, Parent_Field 45 { 46 napToReturn.put('Error', 'No DocuSign Template Found'); 47 return mapToReturn; 48 } 49 Map <string. map<string,="" mapheader="new" string="" string.="">(); 49 Amp <string.string.string.mapheader =="" map<string.string="" new="">(); 40 Amp <string.string.string.string.string.string.string>(); 41 Amp(string.str</string.string.string.string.string.string.string></string.string.string.mapheader></string.></pre>	42						
<pre>st (coccusingneepouts == net ) {     f agrOReturn sepToReturn, No DocuSign Template Found');     return mepToReturn;     f     Mop <string, mep<string,="" mepheeder="new" string="" strings="">();     f }</string,></pre>	43	<pre>vocusign_iemplate_uatac DocuSignTempData = [ if(DesuSignTempData =)] }</pre>	select la, name,Object_AF	'1_Namec, Ro	e_Namec, remplate_Idc, Te	<pre>xt_iaps_witn_API_Namec, Save_to_Object_APIc, Parent_Field</pre>	a_A
<pre>mapToReturn.put('Error', 'No DocuSign Template Found'); return mapToReturn; } Map <string, map<string,="" mapheader="new" string="" strings="">(); </string,></pre>	44	r coocorginempouru == nurr j					
47 return mapToReturn; 48 } 49 Map <string, map<string,="" mapheader="new" string="" strings="">();</string,>	46	mapToReturn.put('Error', 'No DocuSian Temp	late Found'):				
<pre>48 } 49 Map <string, string=""> mapHeader = new Map<string, string="">(); </string,></string,></pre>	47	return mapToReturn;	,1				
49 Map -String, String-appleader = new Map-String, String-O;	48	}					
	49	Map <string, string=""> mapHeader = new Map<strin< td=""><td>g, String&gt;();</td><td></td><td></td><td></td><td></td></strin<></string,>	g, String>();				
Source Properties Docusigneecipient.cis-meta.xmi	Source Prop	erties DocuSignRecipient.cls-meta.xml					

Figure 5. APEX code using Eclipse as the editor.

In only a matter of three weeks, the prototype was created, development was completed and the custom solution was rolled-out, with the DocuSign development piece taking only a little more than one week. They were able to find most of the resources they needed in the DocuSign Developer Center, but also consulted the DocuSign Professional Services team to learn best practices for integrating with Salesforce custom buttons.

## **An Excellent Prognosis**

By using the eSignature API and DocuSign for Salesforce to create a custom solution that integrates with Salesforce Communities, Covered California now has an automated process for reviewing and approving agreements from the entities that provide Californians with access to ACA qualified health plans.

Agreements now move seamlessly through the approval process, which is trackable, secure, transparent, and 100 percent automated. If entities need to amend their information, the new solution is nimble enough to handle changes immediately. The time to process agreements has been reduced now that agreements don't need to be walked from one division to the next and information doesn't need to be manually keyed into spreadsheets. Covered California benefits from complete visibility into the work ow through the Salesforce dashboard and DocuSign. Now, Covered California can readily track where exactly agreements are in the approval process and which divisions still need to sign agreements. In addition to the highly efficient new process, Covered California is saving money and eliminating wasted resources of paper and ink. This efficiency saves time and money, while benefiting from enhanced services offered by Covered California.

You can try the power and flexibility of the DocuSign eSignature API for yourself with a free developer sandbox – just visit the **DocuSign Developer Center**, and see for yourself why Covered California chose the DocuSign API.

#### About DocuSign

DocuSign helps organizations connect and automate how they prepare, sign, act on, and manage agreements. As part of the DocuSign Agreement Cloud, DocuSign offers eSignature: the world's #1 way to sign electronically on practically any device, from almost anywhere, at any time. Today, more than 537,000 customers and hundreds of millions of users in over 180 countries use DocuSign to accelerate the process of doing business and to simplify people's lives.

**DocuSign, Inc.** 221 Main Street, Suite 1550 San Francisco, CA 94105 For more information sales@docusign.com +1-877-720-2040

docusign.com