What is DocXpert?

DocXpert lets you use merge templates and simple SOQL queries to create custom documents in your Salesforce org. Quotes, Contracts, Invoices, or other documents related to your Salesforce data can be generated in just one click using a custom button on any object. The document is attached automatically to your record, making it available to send via email straight from Salesforce. Fully customizable and easy for your users!

Generates documents such as:

- Quotes
- Proposals
- Contracts
- Invoices
- Receipts
- Work Orders
- Renewals

Compatible with: Sales, Service and Marketing Clouds
Can be used in Salesforce Lightning only

The DocXpert package contains the following components:

Package Components

Action	Name	Parent Object	Type
	All	Document Template	List View
	spinner		Lightning Web Component Bundle
	All	Document Data Provider	List View
	documentGenerator		Lightning Web Component Bundle
	docxtemplates		Static Resource
	docxGeneratorCompressed		Static Resource
	Document Template Layout	Document Template	Page Layout
	Document Data Layout	Document Data Provider	Page Layout
	Document Template Asset Layout	Content Version	Page Layout
	DocumentGeneratorTemplates		Apex Class
	DocumentGeneratorDataProvider_T		Apex Class
	DocXpert		App
	Document Data Provider		Custom Metadata Type
	Document Template		Custom Object
	RecordIdExtractor		Apex Class
	RecordFieldValueExtractor		Apex Class
	Document Template		Tab

QueryMethodExtractor		Apex Class
MergeFieldExtractor		Apex Class
DocumentGeneratorTemplates_T		Apex Class
Image Height (cm)	Content Version	Custom Field
Document Identifier	Content Version	Custom Field
Document_Generator_UtilityBar		Lightning Page
Document Identifier	Document Template	Custom Field
Document Base Name	Document Template	Custom Field
Type	Document Data Provider	Custom Field
SOQL Query	Document Data Provider	Custom Field
PDF Link	Content Version	Custom Field
Image Width (cm)	Content Version	Custom Field
SObject Names	Document Template	Custom Field
Active	Document Template	Custom Field
DocumentGenerator		Visualforce Page
docXpertDocGenAction		Aura Component Bundle
pfblueicon		Asset File
DocumentGeneratorDataProvider		Apex Class
DocumentGeneratorController		Apex Class

You also receive the following for a quick start:

- Standard Document Data Providers (see corresponding section in the Administration Guide) with an Excel Reference File for easy copy and paste (see corresponding section in the Administration Guide)

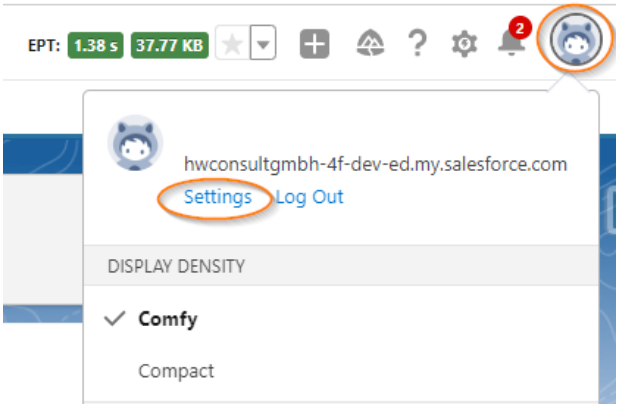
Troubleshooting / Support

This documentation is a complete guide to setting up DocXpert on your own.

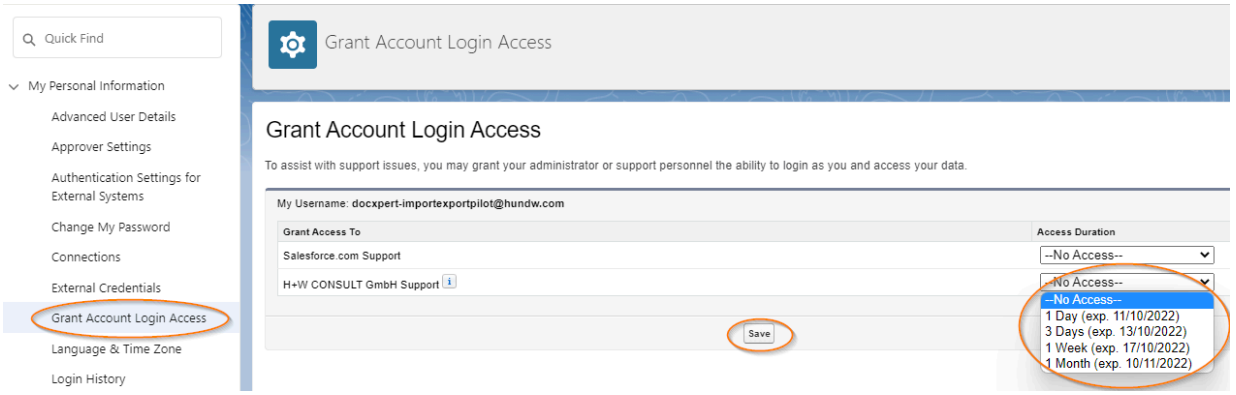
However, if you have a specific question and cannot find your answer in this DocXpert Handbook or in the DocXpert Troubleshooting/FAQs documentation, you can contact DocXpert Support at: DocXpert.Support@hundw.com.

Note: For quicker troubleshooting, DocXpert Support may ask you to give them access to your org. If this is the case, please do the following:

1. Click on your User Icon in the upper-right corner of your screen and click the “Settings” link to enter your Personal Settings.



2. In the menu, click “Grant Account Login Access”. Then next to “H+W CONSULT GmbH Support” give access for the appropriate duration and click “Save”.



If you would like to have a DocXpert Support Agent help you with your configuration, please contact us at DocXpert.Support@hundw.com to purchase a support contract.

Installation Guide

Technical Information

Technical Requirements

Salesforce Requirements

Available in: Lightning Experience

Available for: **Enterprise**, **Performance**, and **Unlimited** Editions.

Supported Browsers

	MICROSOFT® INTERNET EXPLORER®	MICROSOFT® EDGE	GOOGLE CHROME™ **Recommended**	MOZILLA® FIREFOX®	APPLE® SAFARI®
Lightning Experience	NO	Latest	Latest	Latest	Latest*

* Please configure your browser based on the settings recommended by Salesforce:
https://help.salesforce.com/articleView?id=getstart_browser_considerations_safari.htm&type=5

Supported Programs for Merge Templates

DocXpert is based on Office_Open_XML standard:
https://de.wikipedia.org/wiki/Office_Open_XML

We recommend using the latest version of Microsoft Word for Mac or Windows, and creating your templates in .docx formats. Only this format can be used with DocXpert!

Installation

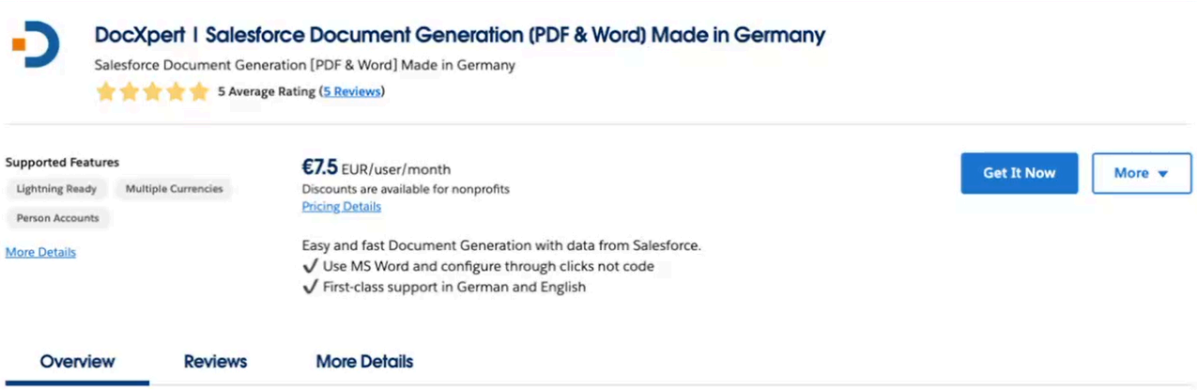
Follow the steps below to install the 30-day free trial version of DocXpert for you to install in your sandbox. The trial version ends automatically after 30 days. If you enjoy using DocXpert and purchase it during the trial period, licenses will be activated in your org, so you can use the functionalities you tested for the duration of your contract. There will be no need to re-install the package after purchase.

If you would like to purchase DocXpert, please contact DocXpert@hundw.com within the 30-day trial Period. Please provide all your org IDs (Live and all Sandboxes) for activation.

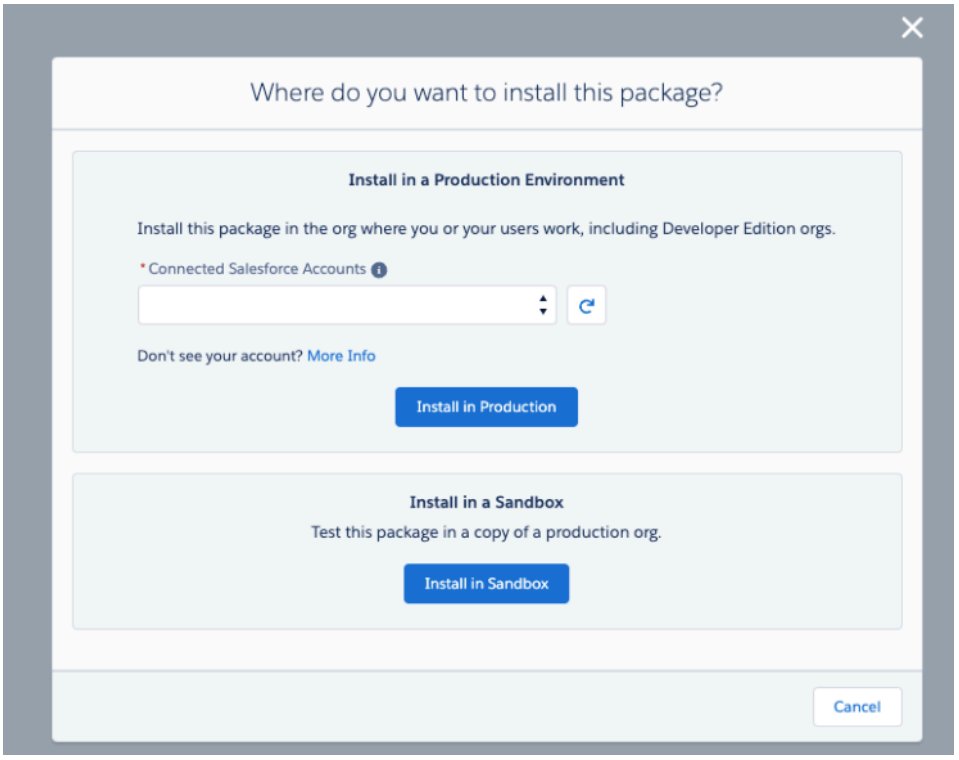
NOTE: If you create a new Sandbox in the future, and wish to install DocXpert in the new sandbox, please remember to contact DocXpert@hundw.com to have the functionality activated. Otherwise, it will run out after 30 days.

Follow these steps for installation:

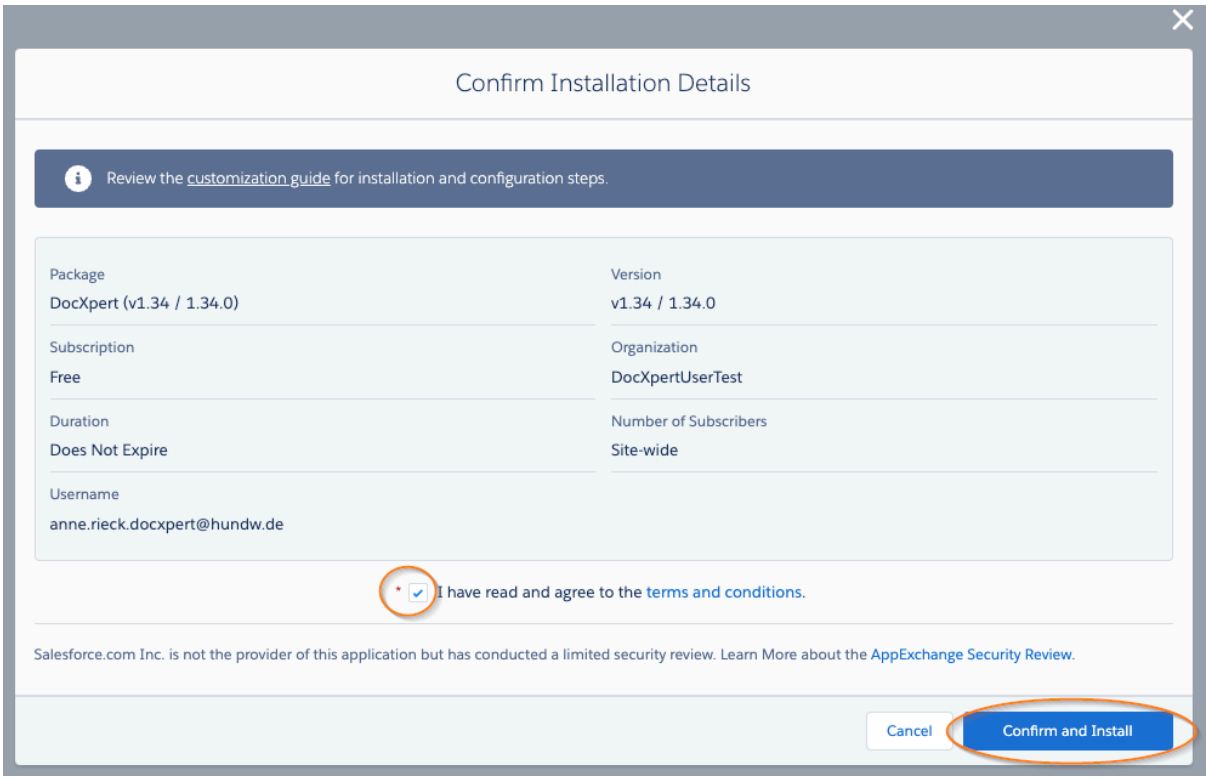
1. After finding DocXpert on the AppExchange (https://appexchange.salesforce.com), click on the “Get it Now” button.



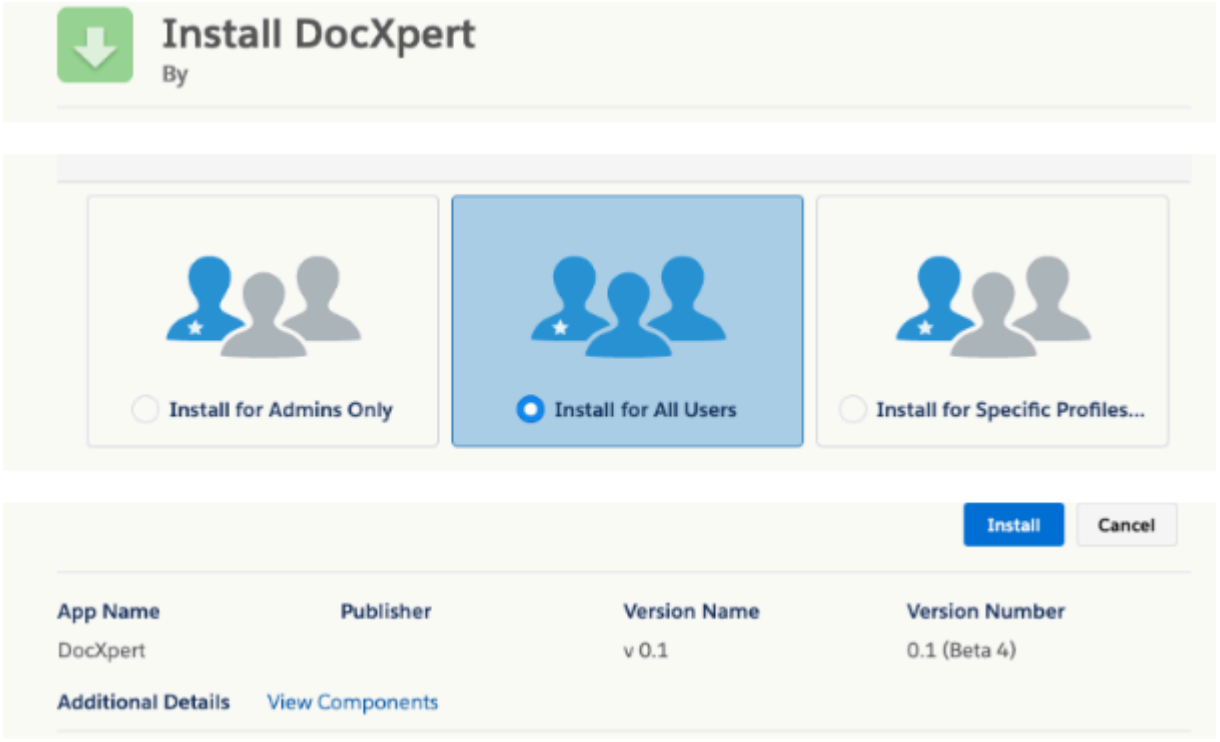
2. You will then be asked to log into your org and where you would like to install DocXpert. We recommend *always* installing DocXpert in your Sandbox first to test!



3. Read and accept the terms and condition, then click on “Confirm and Install” to begin the installation process. This will install the 30-day free trial version of DocXpert.



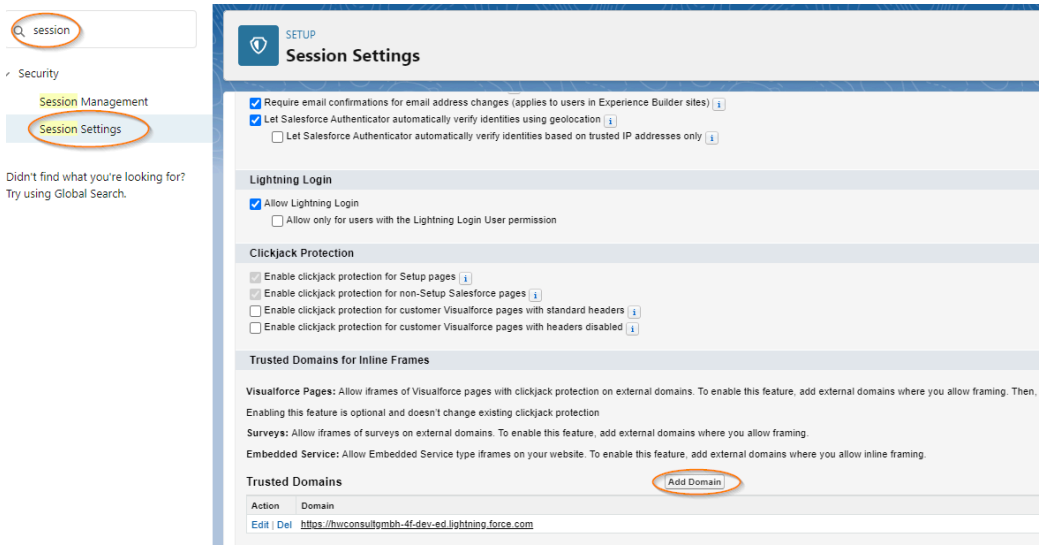
4. Select “Install for All Users”. Press “Install” and wait until the package finishes the installation.



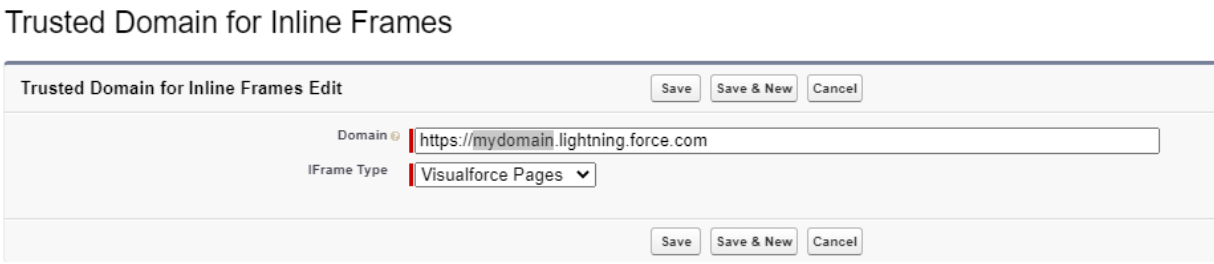
NOTE: Within the trial period, you can test DocXpert with all users in the org. After the trial period, you will need to assign the users the DocXpert license and Permission Sets. Please see the Licences & Permission Set section of this documentation.

5. **Only if your org does NOT yet use Enhanced Domains or HAS BEEN migrated to Hyperforce (rolled out in Release Update Winter ‘23 / Spring ‘23):**
Immediately after installation, whitelist the visualforce site to disable clickjack protection by following these steps:

a. Go to Setup and search for “Session”. Click on “Session Settings”.



b. Click on “Add Domain” Button and enter the following Domain:

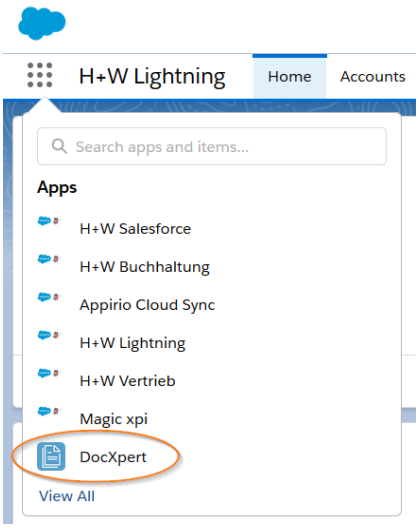


c. Make sure to enter your domain (found in your Salesforce URL): https://yourdomain.lightning.force.com

d. Press “Save”.

6. Because the package only includes a “Recently Viewed” list view, create an “All” list view for the Document Templates object. To do this:

a. Click on the App Manager and select the DocXpert App.



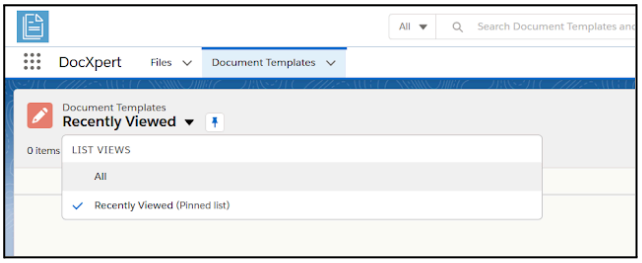
Note: If you cannot find DocXpert in this list, click instead on “View All” and find DocXpert under the “All Items” section, or simply enter “DocXpert” in the search field.

b. Click on the Document Templates tab.

c. Click on the gear icon and “New” in the drop-down menu.

d. Enter “All” under List Name and List API Name. Choose who should see this list view. Then click Save.

e. Now you will be able to view all Document Template records by switching to the “All” list view. (Without this list view, you can only view recently viewed templates.)



Note: If only installing for System Administrator or certain profiles, be sure your own user is assigned one of these profiles. Otherwise, the package will be installed, but you, yourself, will not be able to see the DocXpert App or Document Templates Object. They are hidden by default for all other users when first installed.

For access to the DocXpert App (in the App menu):

1. Go to Setup → App Manager → DocXpert App. Click “Edit” in the drop-down menu to the right of the app’s name.
2. Click “User Profiles” in the menu on the left of the screen.
3. Find your profile under “Available Profiles” and move it to the “Selected Profiles” columns using the arrow button.

For access to the Document Templates Object:

1. Go to Setup → Profiles → choose your profile in the list by clicking on its name in the “Profile Name” column
2. Search for and click on “document templates” (Object Settings) in the search box, or click on Object Settings, then Document Templates in the list.
3. Click the “Edit” button at the top of the page.
4. Enable the appropriate Object permissions by clicking on the checkboxes in the “enabled” column. Click save.

Licenses & Permission Sets

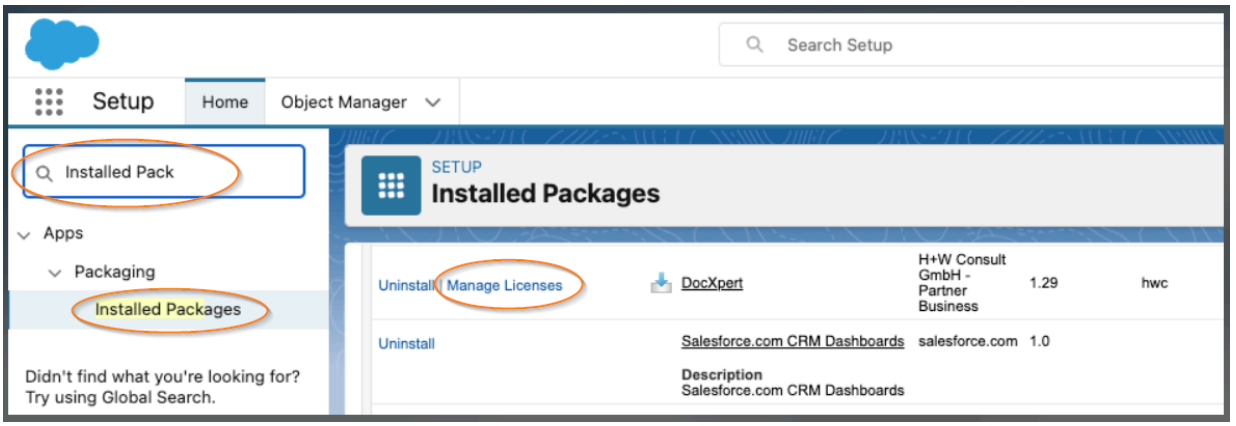
When installing DocXpert for the first time, please make sure you assign Users the DocXpert User License and appropriate Permission Sets. Otherwise, the users will not be able to administer or use DocXpert.

NOTE: Even when using the Option “Install for all Users” you will need to make sure DocXpert Licenses and Permission Sets are assigned to users.

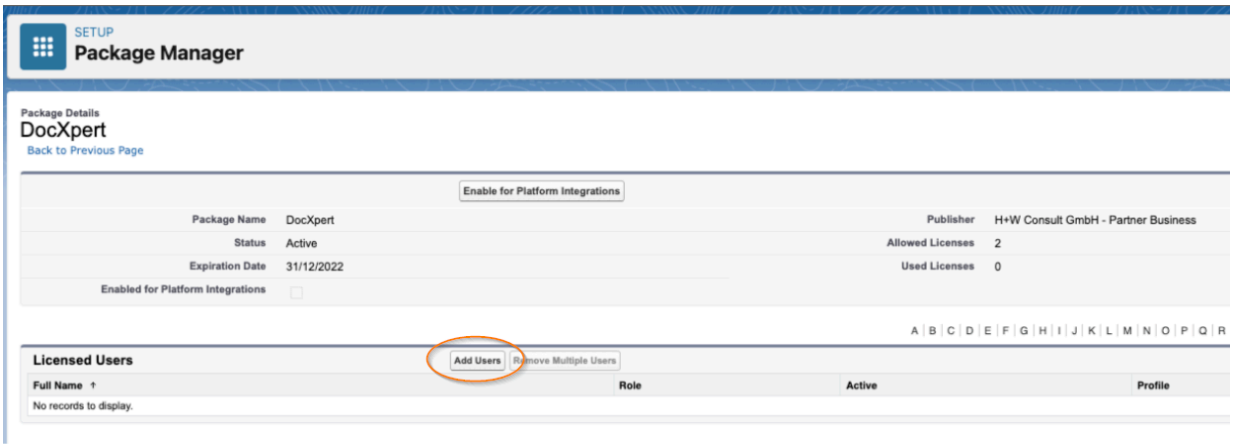
DocXpert licenses:

As of package version V1.30, all administrators and users need a DocXpert license to use the package. For this, a Salesforce Administrator needs to assign the License “DocXpert” in the Permission Set License Assignments section of the User’s record.

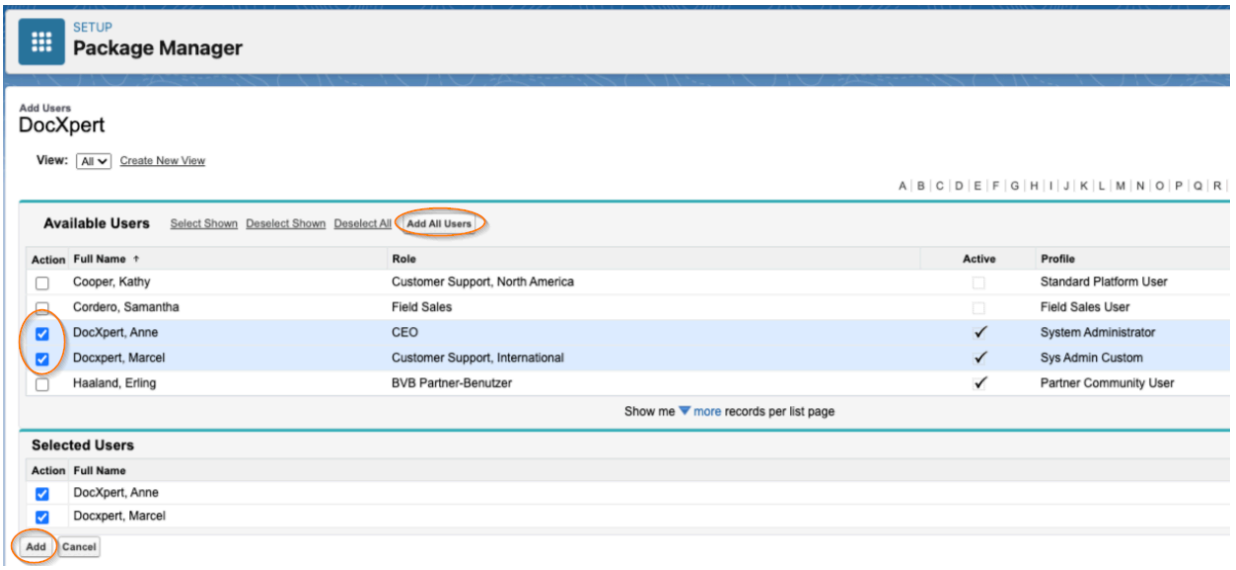
1. Navigate to Setup → Installed Packages → click on the “Manage Licenses” link next to the Package “DocXpert”:



2. Click on the “Add User” button:

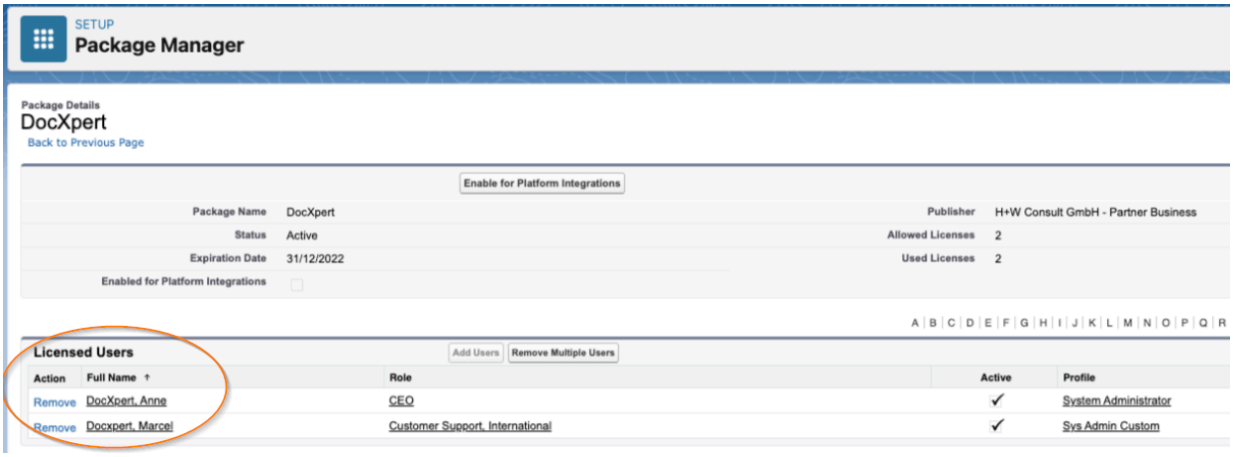


3. In the list of Available Users, activate the checkbox next to the users who should have a DocXpert license and click “Add” at the bottom of the screen.

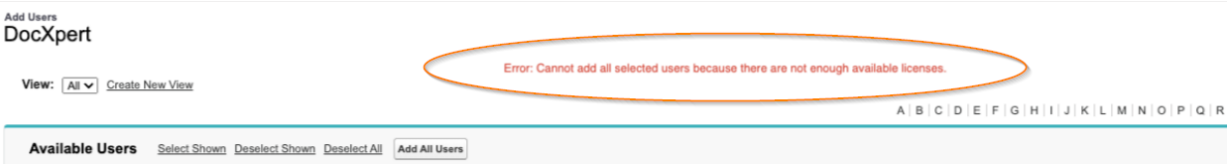


Note: if you would like to add the DocXpert license to all users in your org, click on the “Add All Users” button instead of activating each individual user.

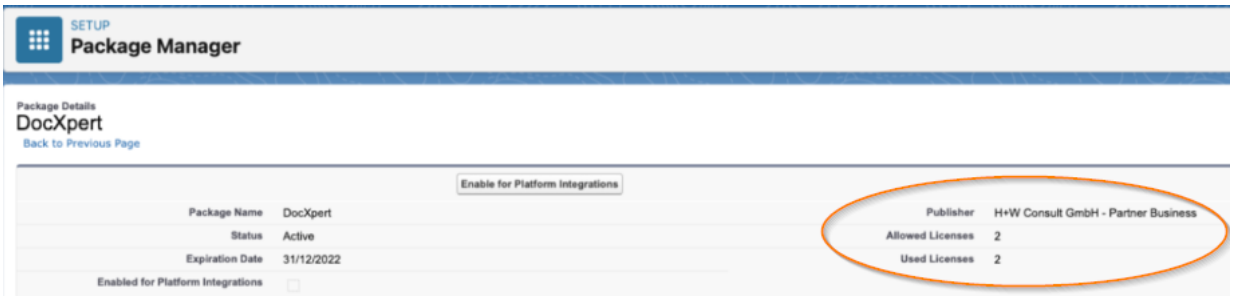
4. After clicking “Add”, you should see the selected users in the Licensed Users list:



If you cannot add the licenses, you will receive an error message:



Please check under Setup → Installed Packages → DocXpert (Link: Manage Licenses) that you have enough free licenses to use. You will see them here:



Permission Sets:

In addition to the license, permissions sets must be assigned to set up (for Admins) and use the product successfully (all Users).

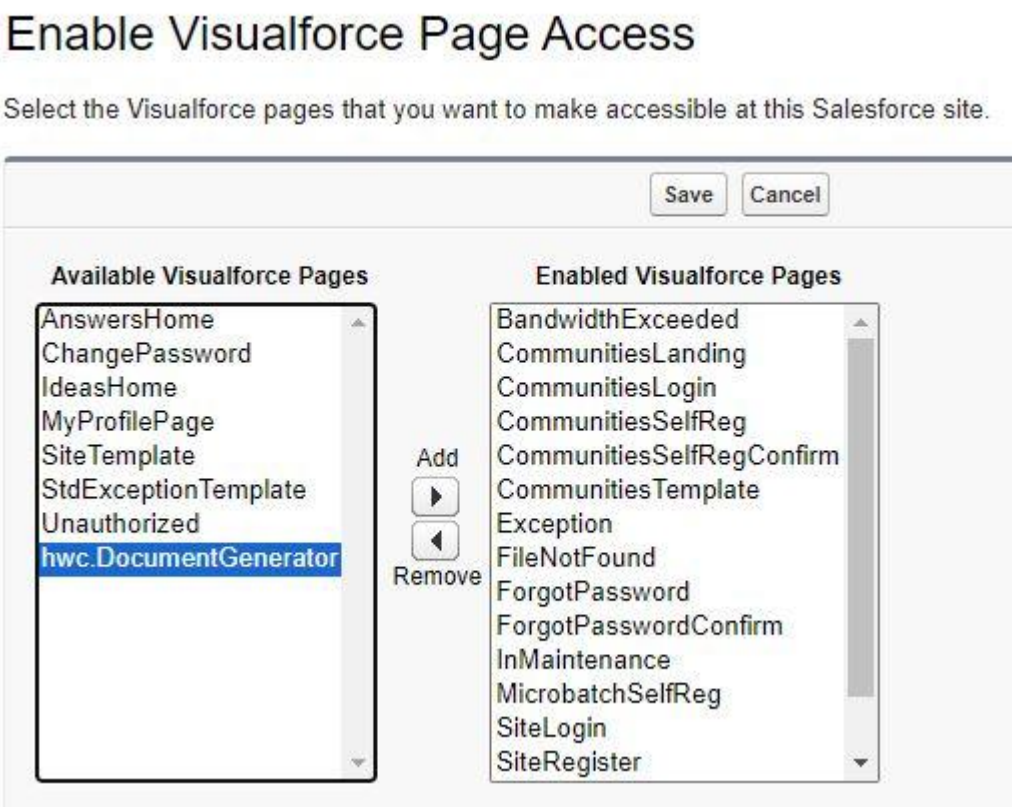
DocXpert Admin permission set:
Assign this permission to users who will set up and manage DocXpert.

DocXpert User permission set:
Assign this permission to any user who will be generating documents using DocXpert, including Admins. If Admins do not have this permission set, they cannot generate documents!

Digital Experience (Communities)

If you use DocXpert in Digital Experience and you would like to generate and save documents in your community you have to enable the visualforce page of DocXpert. Please follow these steps to do so.

1. Go to Setup → Digital Experience → Choose “All Sites”
2. Click on “Workspaces” on the left of your Community Site
3. In the Workspace click on the Administration Button
4. Choose “Pages” in the Menu
5. Choose “Go to force.com” under Advanced Customization
6. Scroll down to the Section “Site Visualforce Pages” and click on “edit”
7. You can now add the Visualforce Page “hwcDocumentGenerator” to the list of enabled Visualforce Pages and save.



Technical Limitations & Considerations

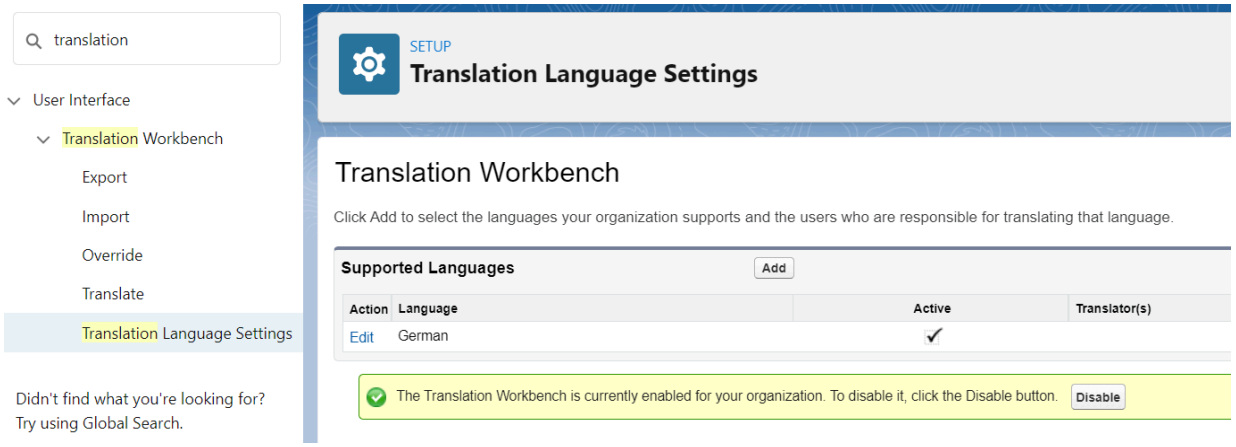
- Max Images 1 MB
- Max Lines when using a Loop 500
- Max Templates File Size 4 MB
- 200 Pages Limit
- For in For Limit in Total Lines 500
- Generation Time depends on client Hardware
- Rich Text fields are not supported in DocXpert

Languages

DocXpert is available in English and German.

The language use depends on the language setting in each User's personal settings. If the user's language is set to English, English will be used. If the language is set to German, German will be used. Other languages are not currently supported out-of-the box, but can be entered manually.

Note: If using any other language than English (including German), please make sure that the Translation Workbench has been activated, and that you have added the language(s) you intend to use to the list of "Supported Languages", as shown here as an example when using German:



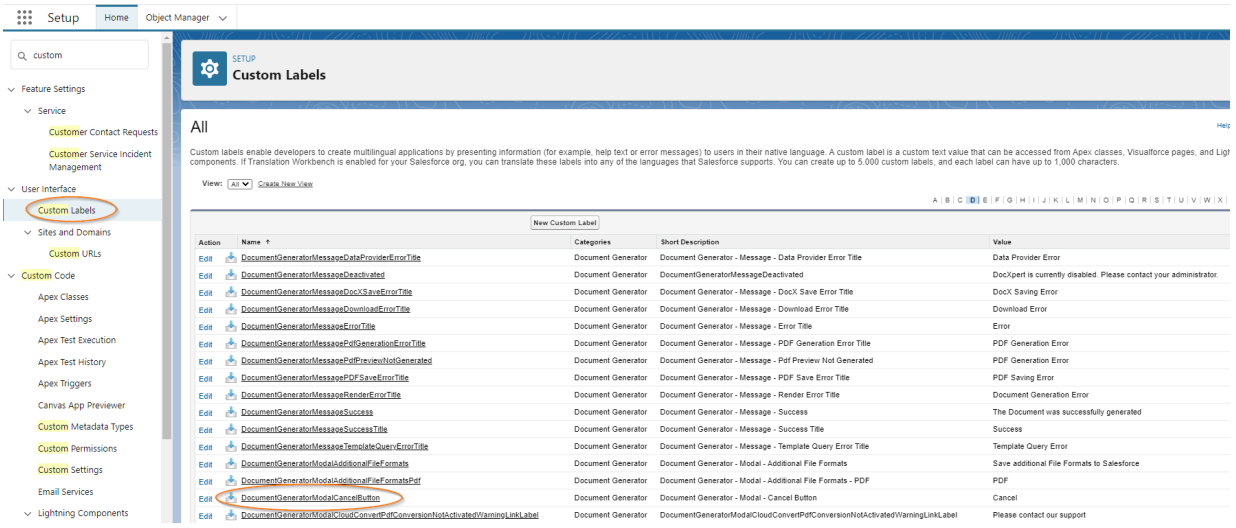
The DocXpert labels to be translated can be found under:
Setup → Custom Labels → click the letter "D" to see all DocXpert Labels.

These include:

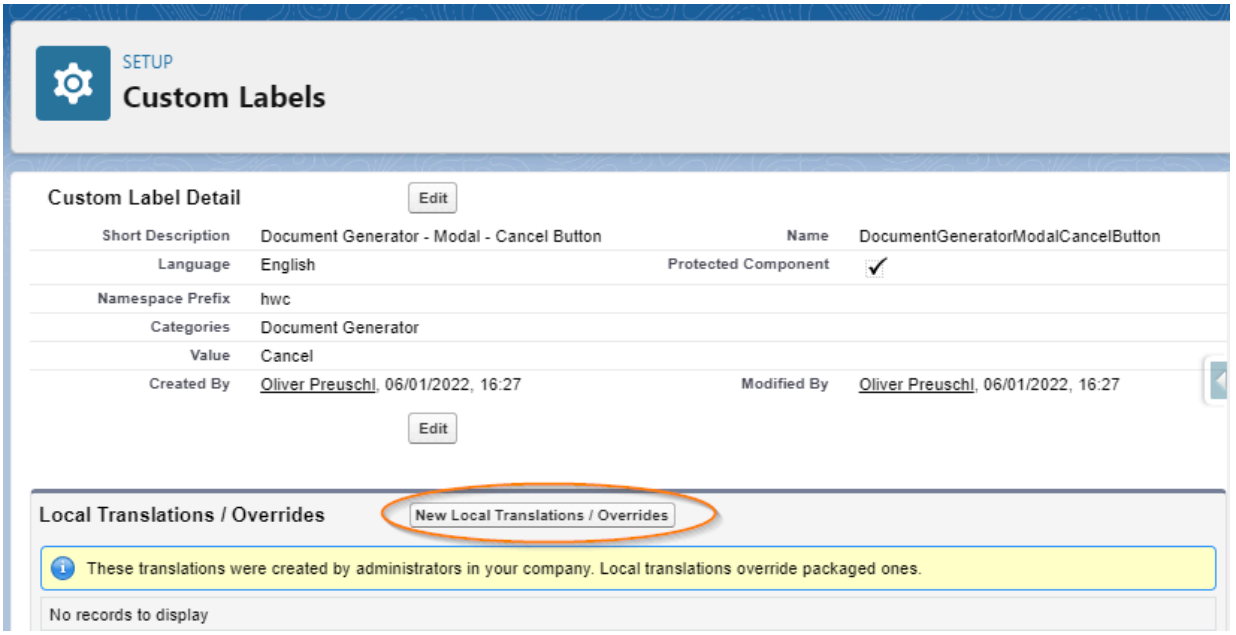
- The labels shown in the pop-up Document Generation window, when a user clicks on the Document Generation button, including the "cancel" and "save" buttons
- Error messages shown, if the document generation does not work

Example: Translating the “Cancel” button in the pop-up window

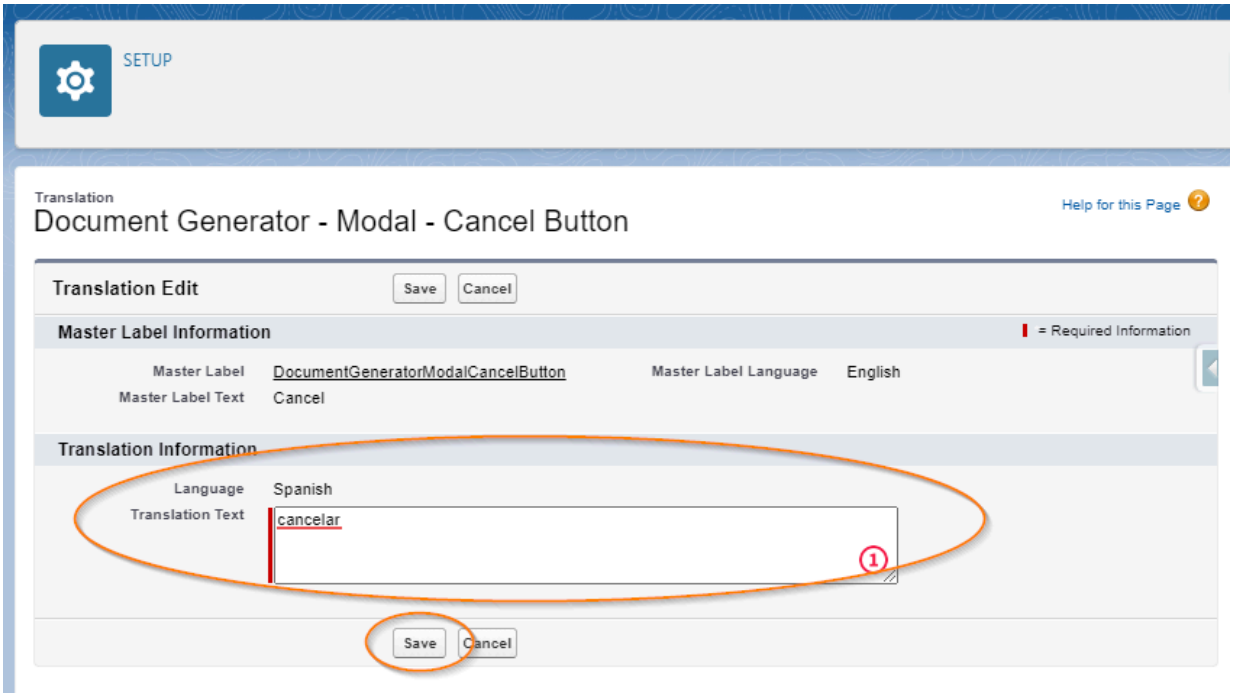
1. Find the correct component. For this example, the Name is “DocumentGeneratorModalCancelButton”. Click on the Name.



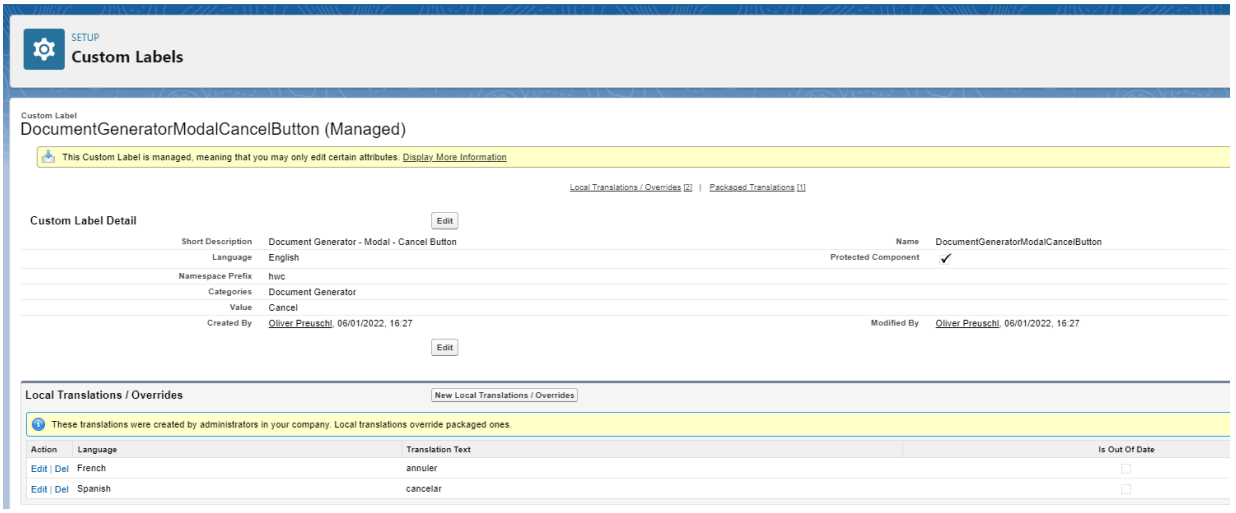
2. Click on the button “New Local Translations / Overrides”.



3. Choose the language and enter the corresponding Translation Text and click “save”.



4. Repeat these steps for any additional languages. At the end, you should see the list of languages & translations under the Custom Label, as in this screenshot:



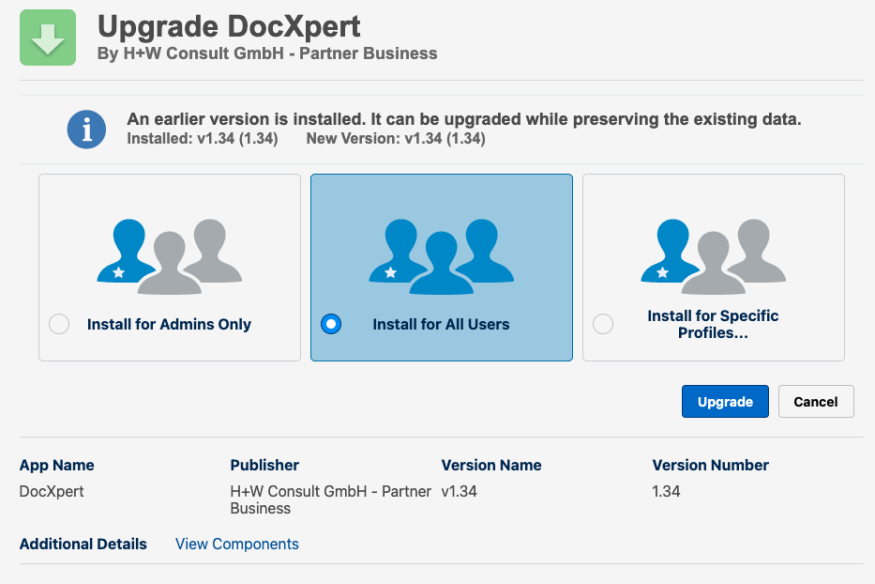
5. Now, when a user sets his user to one of the languages listed for this component (the “cancel” button), they should see the custom translation, as shown here:

Upgrade DocXpert

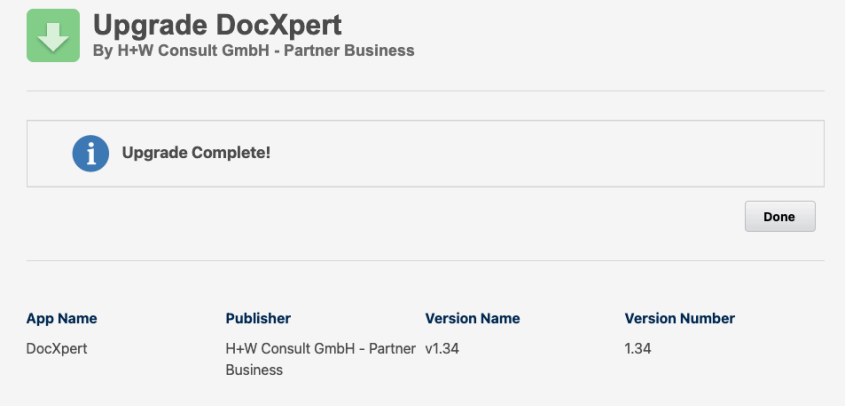
To upgrade your DocXpert to a newer version, simply find DocXpert on the AppExchange (https://appexchange.salesforce.com), and follow the sames steps as for the Installation.

NOTE: When upgrading from V1.16 to V1.34, please contact docxpert.support@hundw.com to get the new add-on and have your license extended for the time period of your contract, otherwise it will run out in 30 days and the PDF functionality will be disabled. Please also remember to assign the Licenses and Permission Sets are assigned to users. How to do this is described in the Licences and Permission Sets section of this documentation.

We still recommend doing this for your sandbox org first, not production, and installing it for all users as shown here:



When finished, you will see this screen:

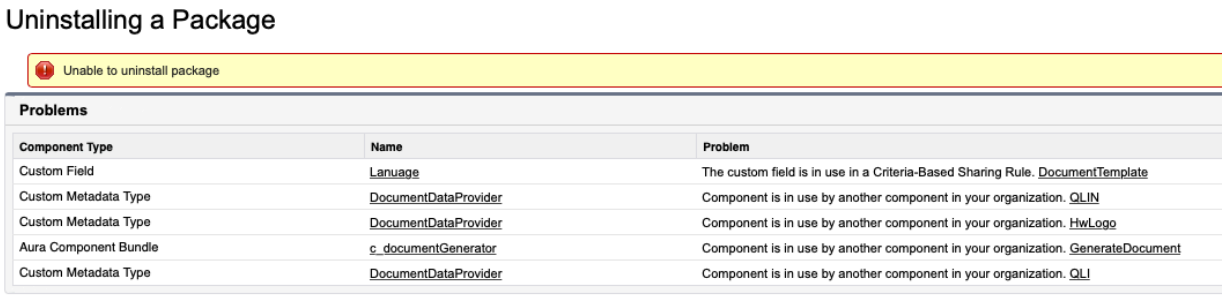


Uninstall DocXpert

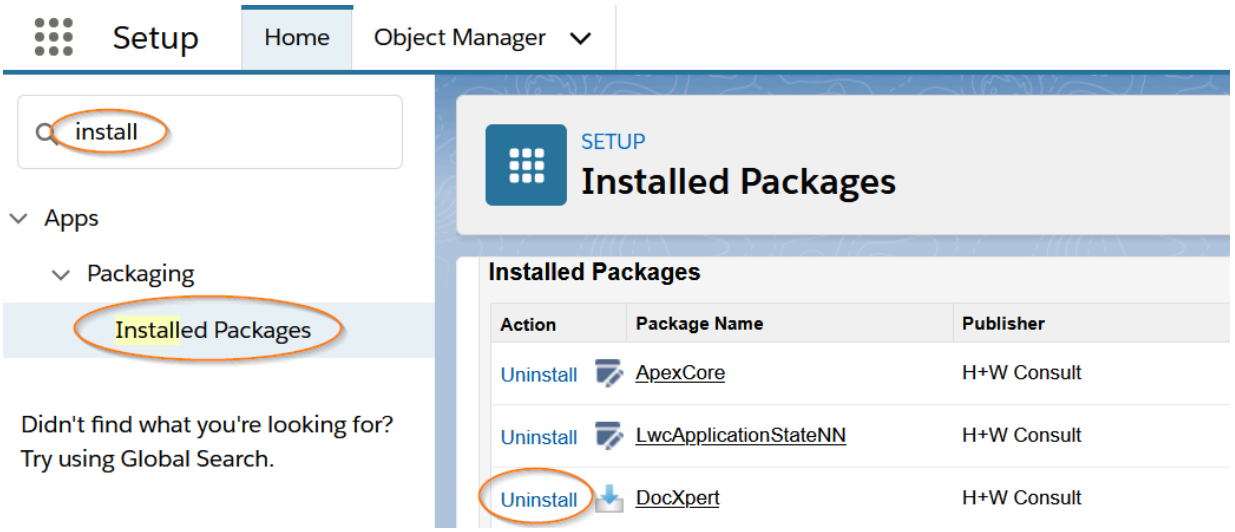
Please delete or clean all related objects:

- Created Actions
- Added Buttons or Fields to Page-Layouts
- Metadata in Data Provider

Otherwise, you can not uninstall the package and will receive the following error:



After this clean-up, you can uninstall the package just as you would with any other Salesforce package.



Press “Yes” and Uninstall Button.

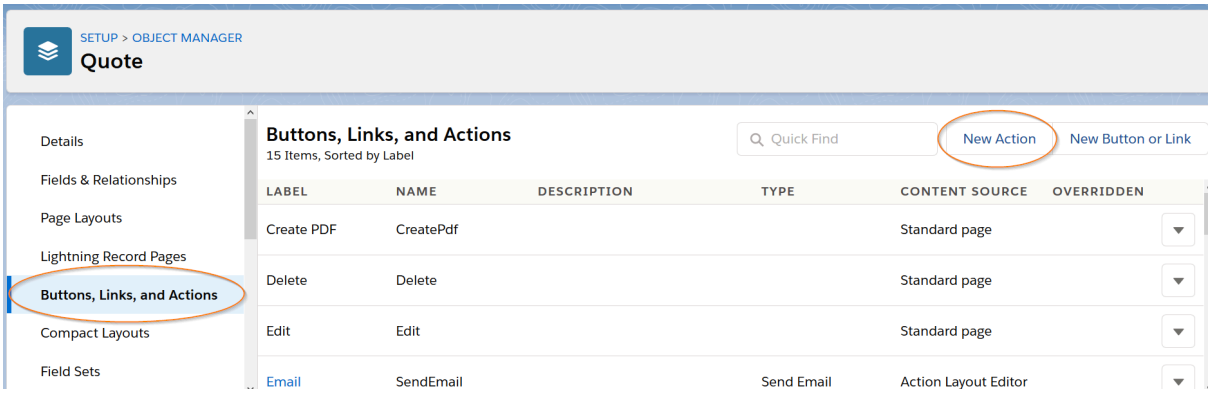
Note: Consider backing up your Document Data Providers and/or other metadata, if you plan to re-install the package again in the future.

How to Create the Button

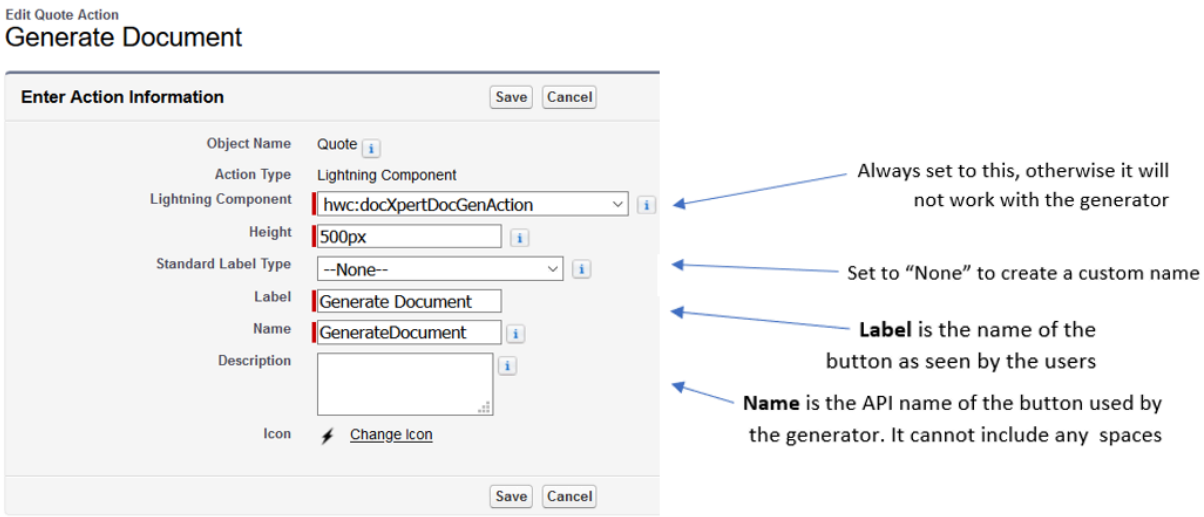
1. Navigate to the object by clicking on
   Setup → Object Manager → *Your Object (ex. Opportunity)*

Note: You may need to activate your object before continuing to the next step.

2. In the menu on the left side of your screen, click “Buttons, Links, and Actions” then the “New Action” button in the upper right hand corner of the window.

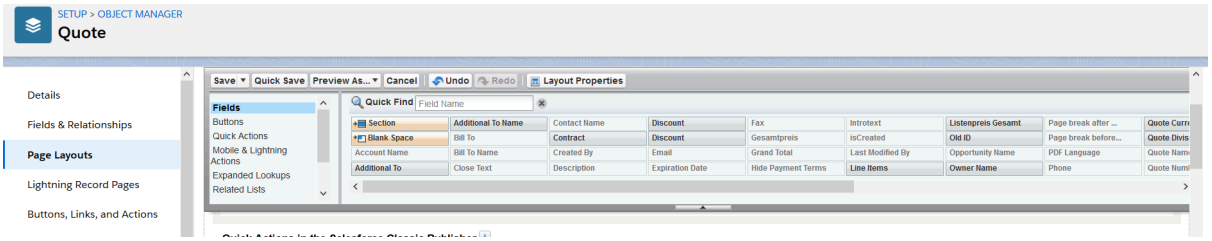


3. Enter the necessary information and click “Save”.



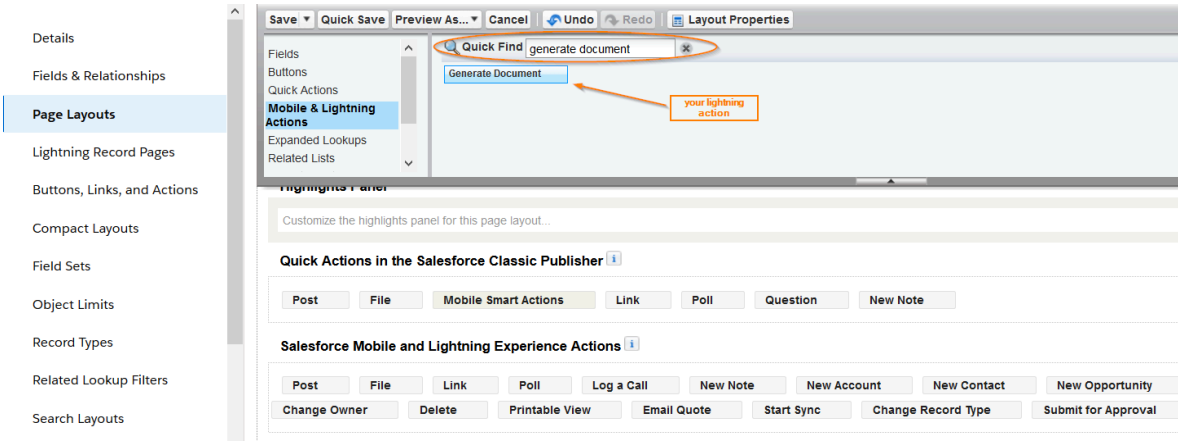
4. Put the newly created lightning action on your page layout(s).

   a. On your object in the Object Manager (the window you see after clicking “Save” in Step 3), click the “Page Layouts” link on the left side of the screen.

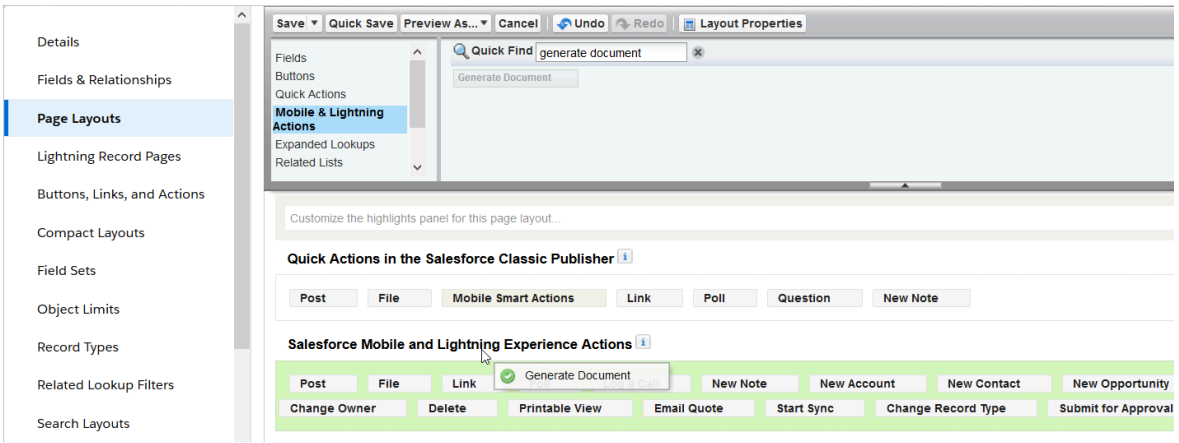


Note: If you clicked out of your object, simply go back by clicking Setup → Object Manager → *Your Object*

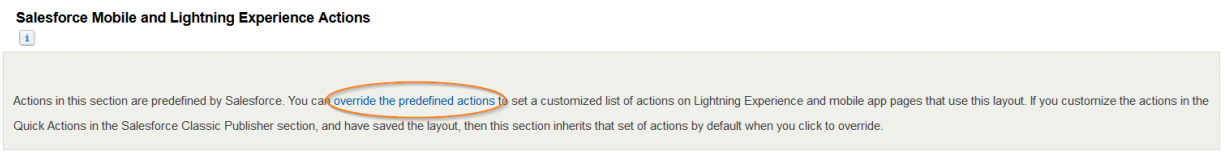
b. Click Mobile & Lightning Actions and search for your newly created lightning actions in the Quick Find box



c. Click and drag the action down to the “Salesforce Mobile and Lightning Experience Actions” section on the page layout.



Note: It can only be added to this section! If the Salesforce Mobile and Lightning Experience Actions sections looks as it does below, first click the “override the predefined actions” link. Then you can add your newly created action as shown.

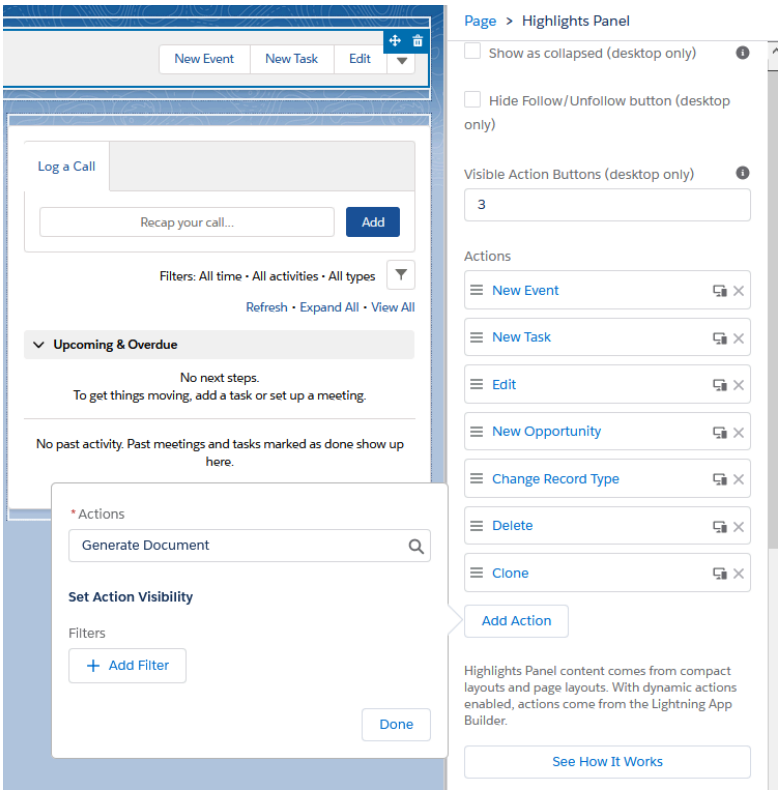


d. Repeat these steps for all layouts on which you would like to have the Document Creator Button.

Note: If you are using Dynamic Forms (available as of the Salesforce Summer '20 Release), an additional step is necessary to bring the button onto your Lightning Record Page:

e. Navigate to your Lightning Record Page, for example:
Setup → Lightning App Builder → Click "Edit" by the page you wish to modify.

Click at the top of the page by the buttons.
Click "Add Actions" in the side panel to the right.
Then, in the pop-up window, search for and enter your button under "Actions"
Adjust the position of the button via drag-and-drop in the side panel.
Remember to click "Save".



5. Set up your sharing settings on your Document Template object to make the appropriate template(s) available for your users. Please see the Set Up User Access section for more information.

Document Generation Options

When using DocXpert, you can generate .docx and/or PDFs. Before using DocXpert, you need to configure your company's preferred options. Below are the options to choose from. For configuration, please read: Configure Document Generation

DOCX Generation Options

Choose whether to generate .docx documents or not (i.e. when only PDFs should be generated):

--None--	Docx function is deactivated
None	Docx function is deactivated
Client-Side	Docx function is activated

PDF Generation Options

None	PDF function is deactivated. The PDF options will be hidden in the document generation window: 
Server-Side (CloudConvert)	CloudConvert generates the PDF. For this option, please contact us at docxpert@hundw.com to purchase the PDF Add-On and receive a key for the CloudConvert license.

For **Orgs who have Upgraded** from a V1.24 or lower:
The "Server-Side (Salesforce)" PDF Generation option is no longer available. The option is still listed in your setup, but does not allow you to generate PDFs. If chosen, you can generate Word Documents, but the PDF options in the document generation window are still hidden. (See above screenshot by "None")

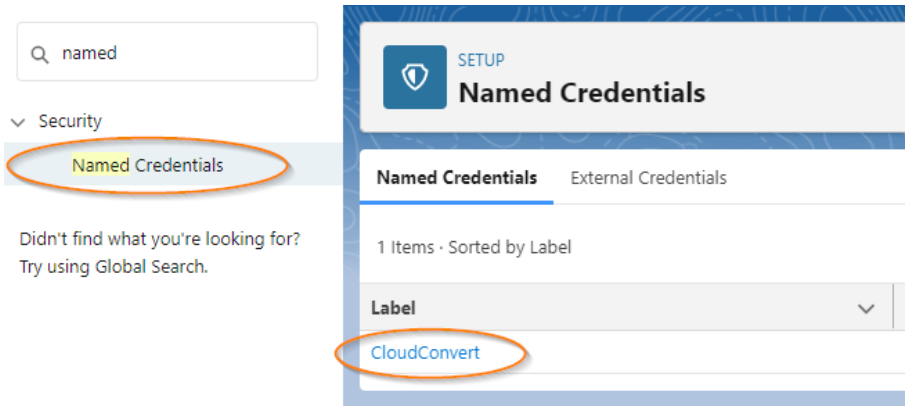
PDF Configuration

If you would like to generate PDFs using the third-party service CloudConvert, please contact us at docxpert@hundw.com to purchase the PDF Add-On and receive a key for the CloudConvert license.

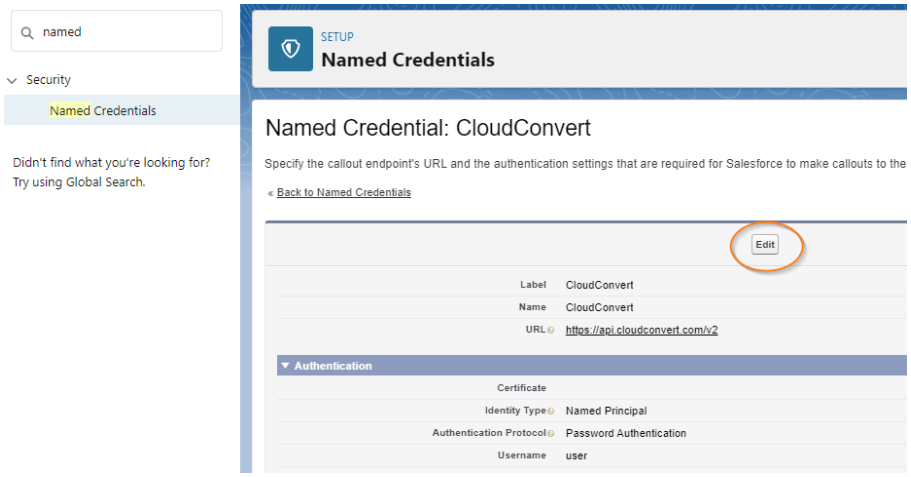
After purchasing the Add-On, you will need to enter the CloudConvert key you receive in your Org settings. To do this, use the following steps.

In your Salesforce instance:

1. Go to Setup → Named Credentials → click on “CloudConvert”

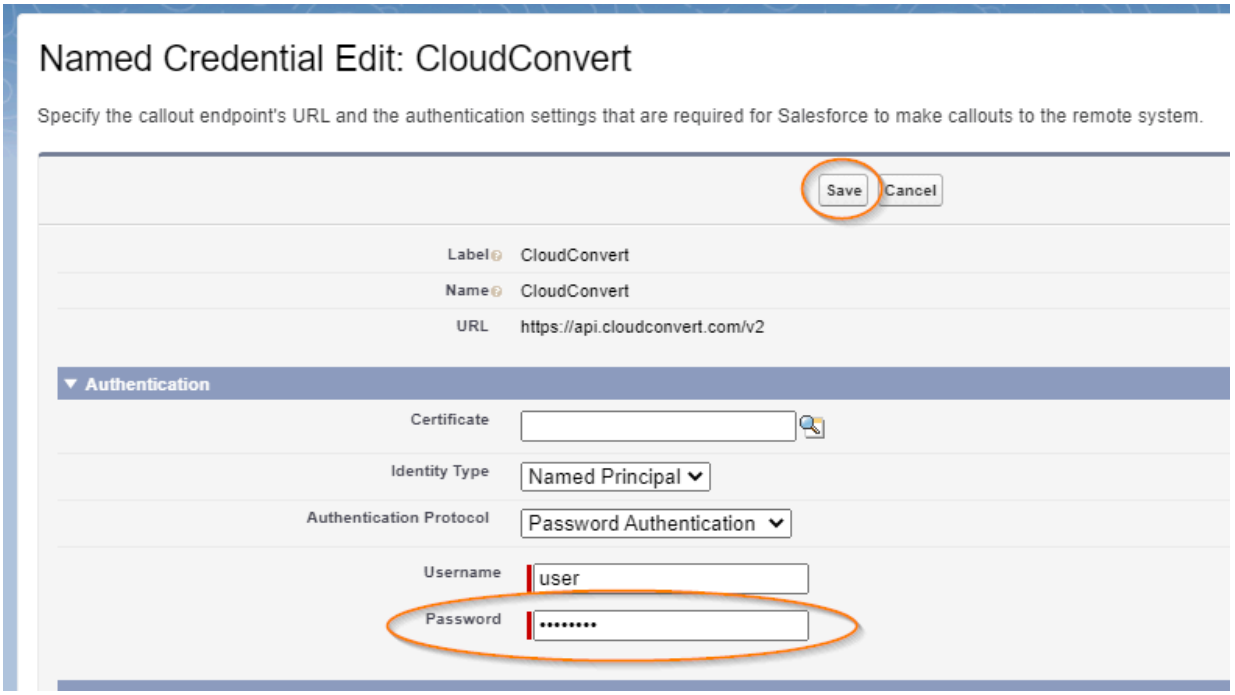


2. Click the “Edit” button on the CloudConvert Named Creditals page:



3. Enter the API key you received from CloudConvert in the field “Password” and click “Save”.

Note: Even though the key you enter is very long, it will still appear after saving as though the password is quite short. This is normal. The key should still be saved correctly.



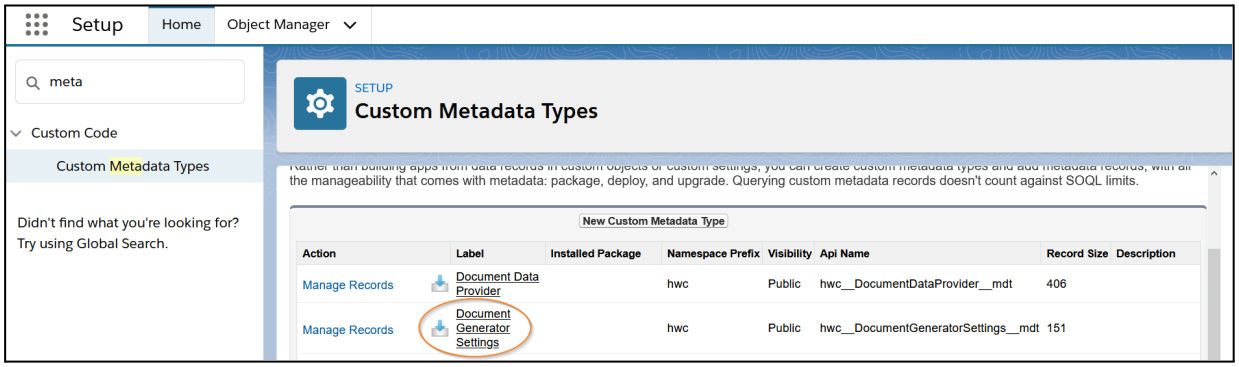
4. Test. You should now be able to generate PDFs.

Note: If you do not see the PDF Generation Options field, please follow the steps in the next section, Configure Document Generation.

Configure Document Generation

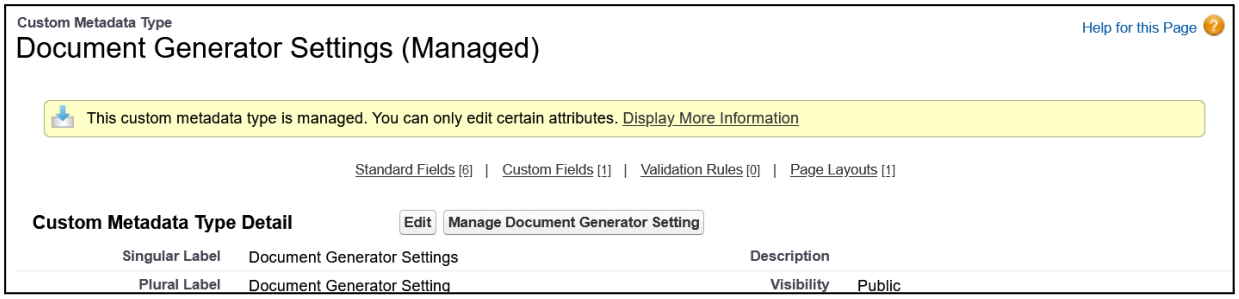
To activate the PDF/.docx generation options, follow these steps in the org:

1. Click in Setup → Custom Metadata Types → Document Generator Settings

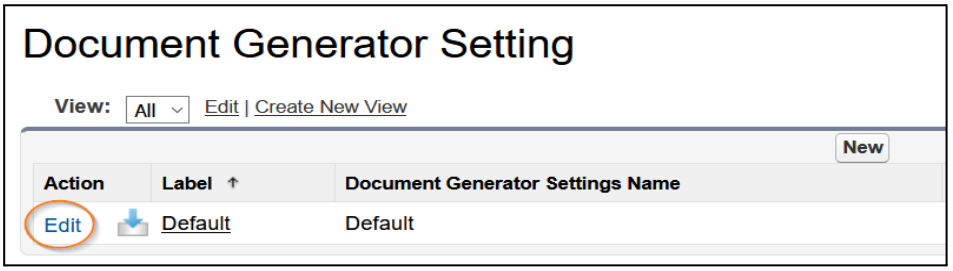


Note: If you would like to generate PDFs, follow the additional steps 1a & 1b below to modify the page layout with the “PDF Generation Options” picklist, otherwise go straight to Step 2.

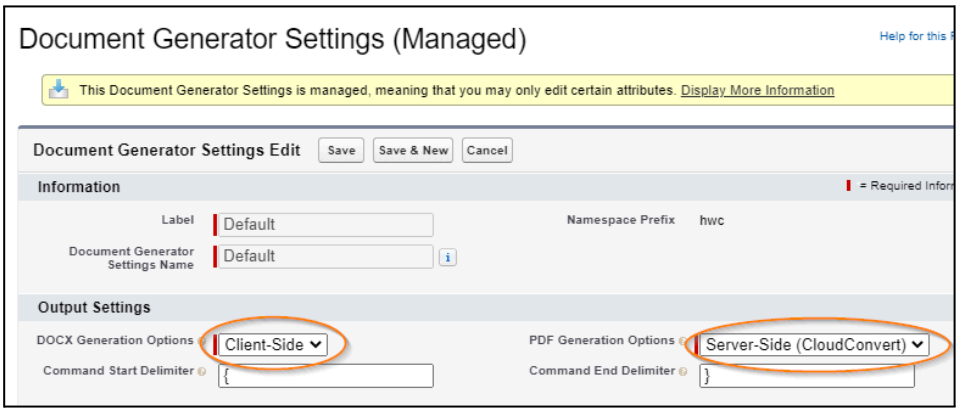
2. Click the “Manage Document Generator Settings” button.



3. Click “Edit” next to the “Default” Document Generator Settings Name.

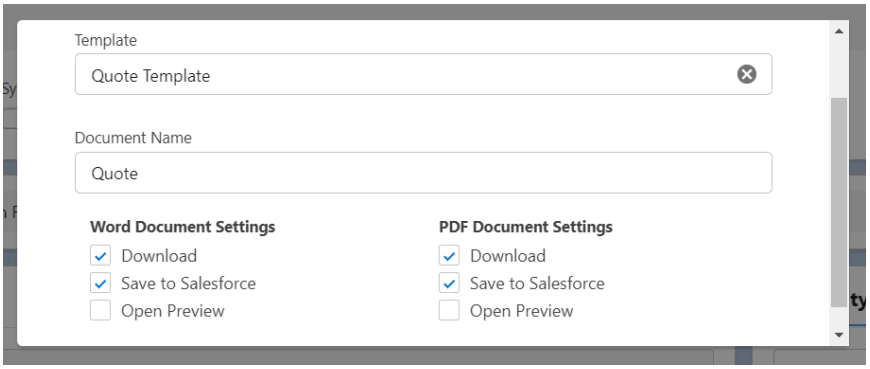


4. In the picklists “PDF Generation Options” and/or “DOCX Generation Options”, select your preference(s). For an explanation of the options, please refer to [Document Generation Options]

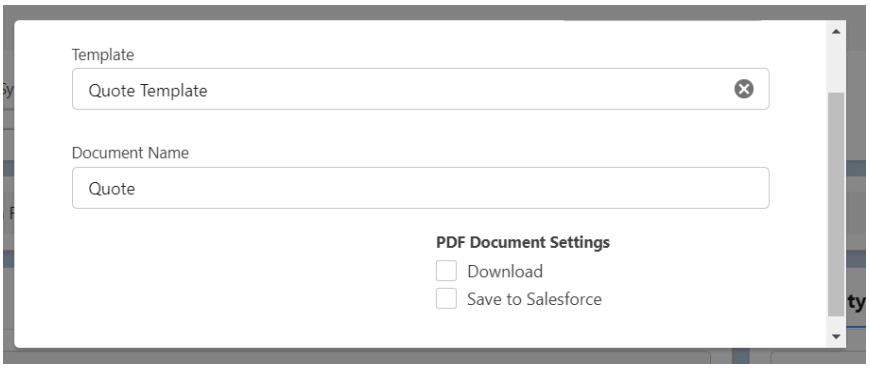


User Experience Example 1: PDF und .docx :

In this case, the “DOCX Generation Option” picklist is set to “Client-Side” (activated) and the “PDF Option” picklist is set to “Server-side (CloudConvert)” (also activated).

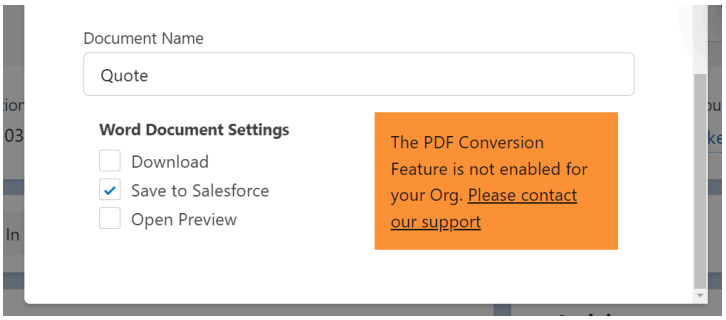


User Experience Example 2: PDF only



In this case, the “DOCX Generation Option” picklist is set to “None” (deactivated) and therefore not selectable. The “PDF Option” picklist is set to “Server-side (CloudConvert)” (activated).

User Experience Example 3: PDF not enabled message



If you see this message, it means you do not have an active CloudConvert account in Salesforce. To purchase the Add-On contact DocXpert@hundw.com. Also contact CloudConvert directly to purchase a package. See PDF Generation Options for more information.

E-Signing

DocXpert itself *does not* create e-signing forms, but it generates PDF & Word files which can be used with most third-party apps for the creation of E-Signing documents. It usually works as follows:

1. Choose and install a third-party tool to create your dynamic e-signing documents. Please inform yourself of the various options on the Salesforce AppExchange.

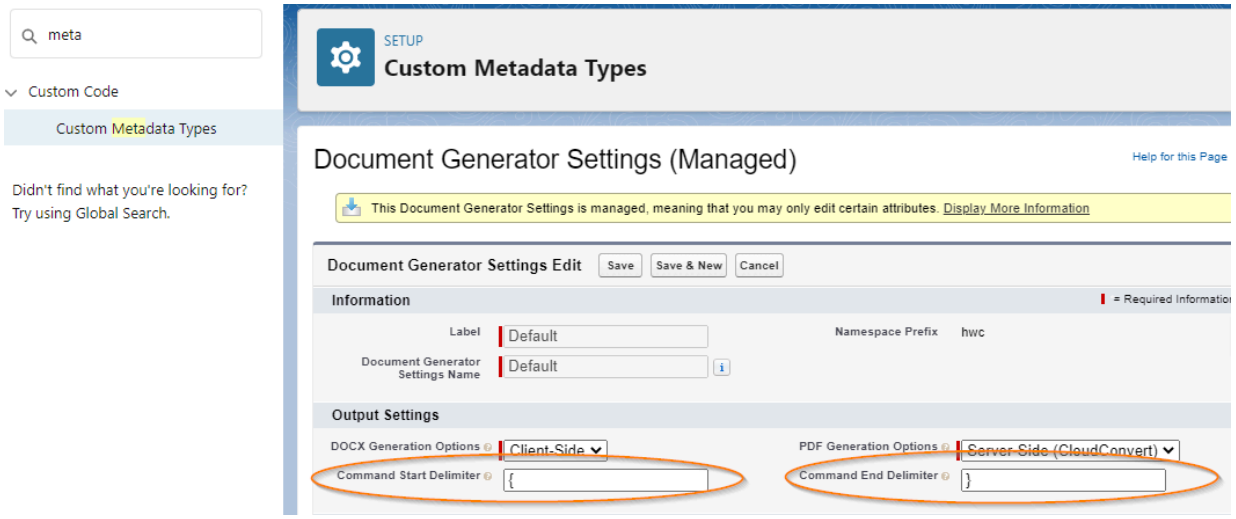
   Note: There are also further third party tools that are not found on the AppExchange. If you use such a tool, please make sure it can connect to Salesforce.

2. First, a PDF or Word document is generated with DocXpert. This document is then used in the third party e-signing tool to generate and possibly send the e-signing document itself.

3. DocXpert gives you the possibility to have a seamless workflow between the two tools. Instead of having to map each field from Salesforce to the e-signing tool, you can generate a PDF or Word document with DocXpert which already contains the syntax needed to create the dynamic e-signing functionality. For this, it may be necessary to adjust the syntax in DocXpert (see example below).

   For example, this functionality has already been tested and implemented by our team with the e-signing tool Adobe Sign.

When generating documents with DocXpert, { } parenthesis are used as default delimiters for the template merge fields. If this causes problems with the E-Signing application (i.e. when the 3rd party tool uses similar delimiters), DocXpert gives you the option to change the command start and end delimiters to suit your needs. You can find this in the Document Generator Settings:



Example:

You use a third-party app to generate e-signing documents. Your template will have merge fields to generate through DocXpert as your first generation step, and it will also contain dynamic e-signing fields which are generated using the e-signing tool as your second step.

If the delimiters for both tools are the same, the tools will not know which commands to read. Therefore, you may need to adjust the delimiters for DocXpert, so there is a difference between the two.

Take the following case:

- The e-singing tool uses {{ }}
- DocXpert delimiters are therefore changed to use [[ ]]
- You would like to generate a dynamic signature field for the e-signing document (e-signing functionality) which shows the name of the Quote in Salesforce (DocXpert merge field)

The solution could look similar to this in your template:
{{([[Quote.Id]])*Quote_es_:signer1}}

Note: The actual solution depends on how your 3rd-party e-signing app works and what DocXpert delimiters you use. Please check your e-signing tool to create your own individual solution. If you would like to use our consulting services to help set up your e-signing, please contact DocXpert.Support@hundw.com.

Administration Guide

Configuration in Salesforce

Documents are generated in DocXpert using a Word document merge template in a **.docx** format, Document Data Providers to fill in the merge fields on the templates with Salesforce data, and a custom button for the users to generate the documents.

The steps are as follows:

1. Install DocXpert
2. Decide what data from your Salesforce org to use in your document
3. Set up your Document Data Provider(s)
4. Create and upload your merge template
5. Create a custom button and place it on your page layout
6. Set up user access
7. Test
8. Deploy

The steps in this guide are to create an example Quote document. To demonstrate the DocXpert functionalities, some fields are based on custom fields and custom Data Providers, which will need to be created in your org. Alternatively, you can adapt the steps and instructions to create your own custom documents based on objects and fields in your own org.

If you would like to create the exact Quote document as described in this documentation, the following functionalities are custom:

- Custom field: Additional Information (text field on Quote object)
- Custom field: Product Picture (text field on Product2 object)
- Company Logo “HwLogo” (upload your own company logo)
- Custom Document Data Provider “HwLogo” (for your company’s logo)
- Custom Document Data Provider “Quote” (to include the Additional Information field)
- Custom Document Data Provider “QuoteLineItems” (to include the Product Picture field)

How to upload images and create data providers are included in this documentation.
For help on creating custom fields in Salesforce, refer to this Create Custom Fields article.

Note: We recommend building and testing all functionality in your organization's sandbox before deploying it to your live production environment.

Before You Begin

The configuration of DocXpert involves a few main parts:

1. **Setting up the Document Data Providers**
   Administration Guide: Document Data Providers

   These are the queries that get the data from Salesforce records to insert in the generated document. In most cases, you will need to create your own custom Document Data Providers.

2. **Creating a merge template and uploading it to Salesforce**
   Template Creator Guide: Merge Templates

   This is the document you create in Word as a template. It uses merge fields to tell the generator where to place the Salesforce data on the page. The templates are then uploaded as records on the Document Template object.

3. **Configuring user access**
   Administration Guide: Set Up User Access

   User access to the template(s) depends on the sharing configurations on the Document Template object and must be configured.

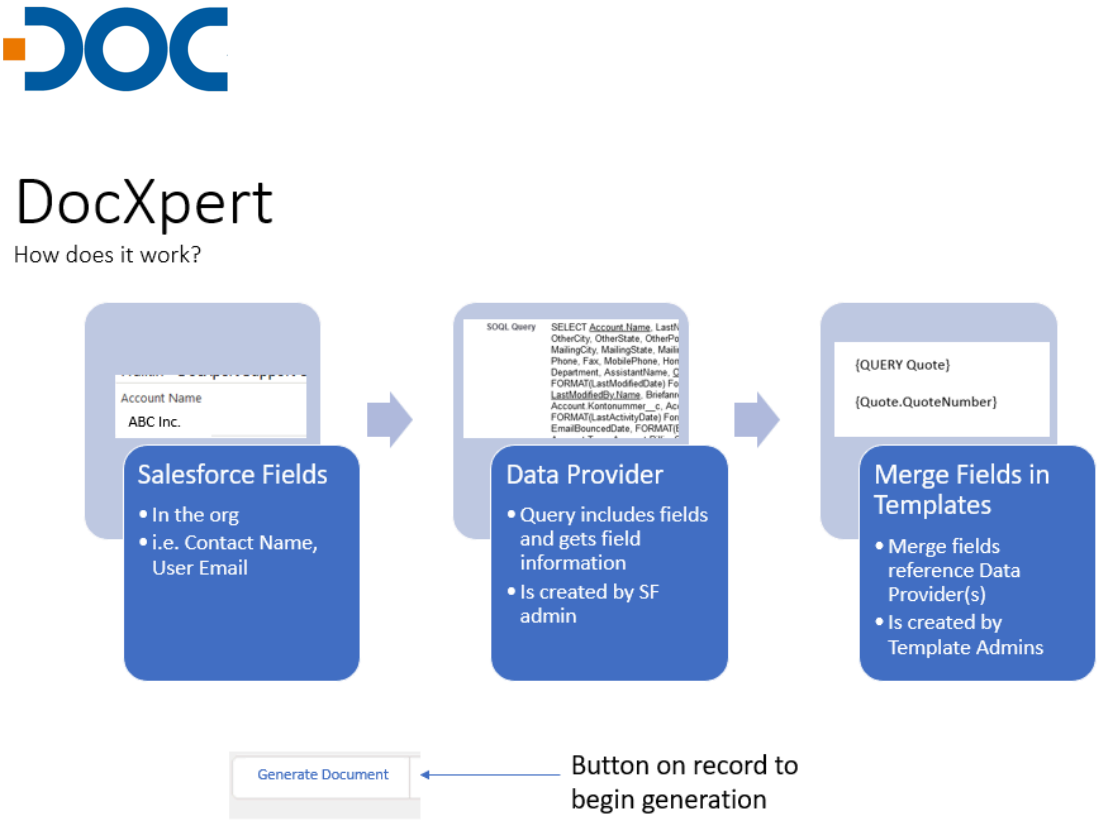
4. **Configuring the document generator button**
   Administration Guide: How to Create the Button

   This is the button the user clicks to generate the document, bringing the data from Salesforce onto the template page, creating a finished Word document.

5. **Testing & Deployment**
   Administration Guide: Testing / Deployment

   We recommend always configuring DocXpert in your sandbox, testing, then deploying it to your live system and users. This involves various deployment steps.

Please see the following diagram to understand how the parts of DocXpert work together. Note there are parts that need to be created by Salesforce (SF) system admins, and parts that can be done by Template Admins who do not have Salesforce admin permissions:



Before beginning any configuration, be sure to draft your document and define the information you would like to auto-generate. Only data saved in Salesforce can be auto-generated in the document.

Activate Quotes

This handbook shows an example of document generation on the Quote object. In new Salesforce Orgs, the Quote object first needs to be enabled to see it. To do this, navigate to:

Setup → Quote Settings

and click on the “Enable” button.

Fields

It is important to know which fields you would like to reference and where they are located in Salesforce. For example, when sending an invoice, you need to know whether the address from the Quote, Opportunity, Contact, or Account object should be used for the generated document.

Note: If a field is not filled out on the Salesforce record, the document will be generated, but a blank will be inserted for this field on the document. Be sure to check the document after generation for spelling errors and blanks!

Example:
When generating a quote document, you may want to generate the following:

- Your company's logo
- The customer's address
- The sales representative's contact information
- A list of products and product information from the quote

This is what your draft may look like (italic text is the text to auto-generate):

Company Logo

10 Billing Street
12345 City
State,
Country

Ref: Quote #*0000000123*

Dear *John Smith*,

Thank you for your interest in our products!

Below you will find our quotation for you to review. It includes all products as discussed.
Please review it at your earliest convenience and let us know when we can place your order.

Best regards,
Sales Rep
Sales Rep Telephone
Sales Rep Email

Product	Description	Product Code	List Price	Quantity	Total Price
Product Ex1 *Product Image, if there is an associated image*	*Description 1 Text*	*XX00001*	*ListPrice* €	x	*TotalPrice* €
Product Ex2 *Product Image, if there is an associated image*	*Description 2 Text*	*XX00002*	*ListPrice* €	x	*TotalPrice* €
Product Ex3 *Product Image, if there is an associated image*	*Description 3 Text*	*XX00003*	*ListPrice* €	x	*TotalPrice* €

Additional Information Text (if customer is located in USA)

To get this result, first find where the information is located. You can write this in a table. The table shown below is for the above example Quote. This information will then be referenced when creating your Document Data Providers and Merge Templates to make configuration easier.

Information Needed on Document	Field API-Name(s) in in Salesforce	Field is on which object in Salesforce?	Field is on which record in Salesforce?
company logo	MyCompanyLogo	Files	
customer address	BillingStreet BillingPostalCode BillingCity BillingState BillingCountry	Quote	quote record on which document is generated
quote number	QuoteNumber	Quote	quote record on which document is generated
customer name	Name	Contact (through standard Contact lookup field on quote)	contact record related to quote record on which document is generated
sales rep info: 1. sales rep name 2. phone number 3. email	1. Name 2. Phone 3. Email	User (through standard Owner field on quote)	user record of Owner of the quote on which document is generated
product information: 1. name 2. description 3. product code 4. product image	1. Name 2. Description 3. ProductCode 4. ProductPicture__c	Product2	product2 records associated with quote line items attached to the quote record on which document is generated
product information: 1. list price 2. quantity 3. total price	1. ListPrice 2. Quantity 4. TotalPrice	Quote Line Item	quote line items related to quote record on which document is generated

additional information text	AdditionalInformation__c	Quote	quote record on which document is generated

As seen in the table, some fields may not be stored on the Quote record itself, but on a related object record. For example,

- The sales rep information such as the phone number is not on the quote itself, but found on the Quote Owner's User record.
- The product information is not on the Quote itself but on the related Quote Line Item records or even the Product records related to the Quote Line Item records
- You may also wish to reference the address of the Contact or Account record, instead of using the fields directly on the Quote record.

Once you know exactly where the information you need is located, you can create your Document Data Providers and merge templates.

Some relationships between objects in Salesforce are direct and easy to find:

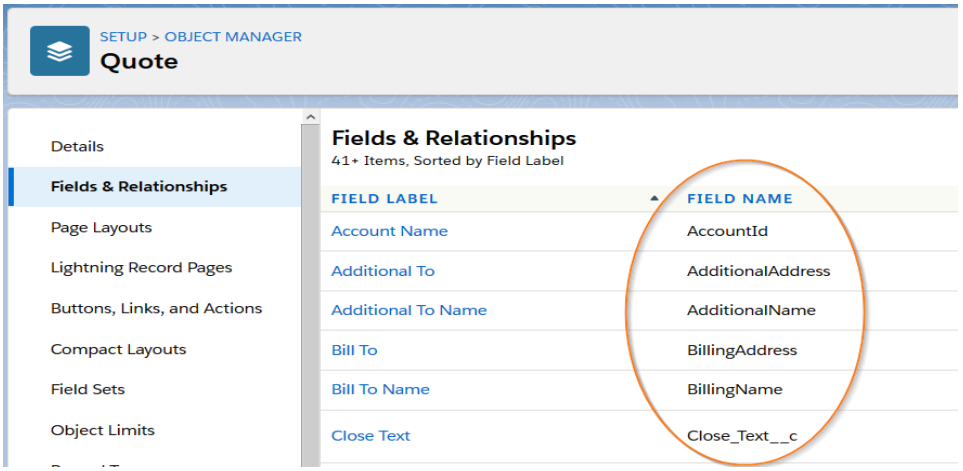
- Contacts are related to an Account. This means you can easily reference Account information from a Contact's record, or a list of contacts from your Account record.
- Quote Line Items are directly related to a Quote. This means you can easily create a list of quote line items from your Quote record.

Other objects are not directly related to each other in Salesforce and therefore usually require a workaround:

- Contacts are not directly related to Opportunities. They are related through a junction object called Contact Roles. Therefore, you cannot directly get Contact information from an Opportunity record, but rather through a Data Provider that queries the Contact Roles Object.
- Campaigns are not directly related to Contacts. They are related through a junction object called Campaign Members.

Please reference the Sales Objects data model to see all relationships between related objects.

Note: To create your Document Data Providers and Merge Templates, you will need to use the Field Names (API names) from the setup, not the Field Label shown on the records. Field Names are found by clicking on:
Setup → Object Manager → *Your Object* → Fields & Relationships



Images

Images you wish to insert when generating a document need to be uploaded and stored in the “Files” tab. If using an existing image, check to make sure it can be located in Files:

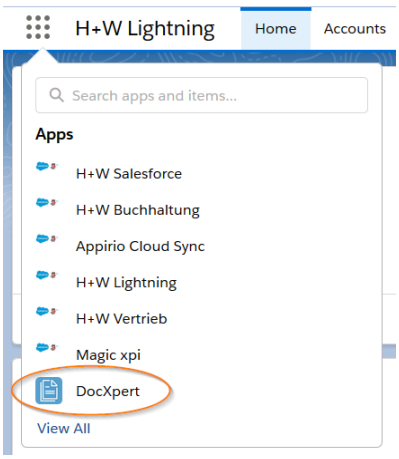
Click the “Files” tab → Click the “Owned by Me” or “Shared with Me” link → check image

Note: You can put an image directly into the header or footer of your Word template without needing a DocXpert data provider or query. Such images *directly in your Word template* do not need to be uploaded in Salesforce as Files.

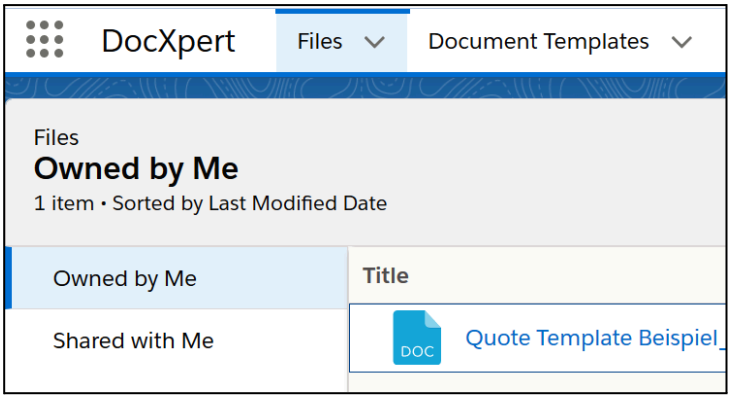
If you cannot find the Files tab, navigate through the App Finder to the Document Generator App. The Files tab should be shown in the tab bar by default.

If you need to upload a new image, follow these steps:

1. Navigate to the App Manager and click on the Document Generator App.

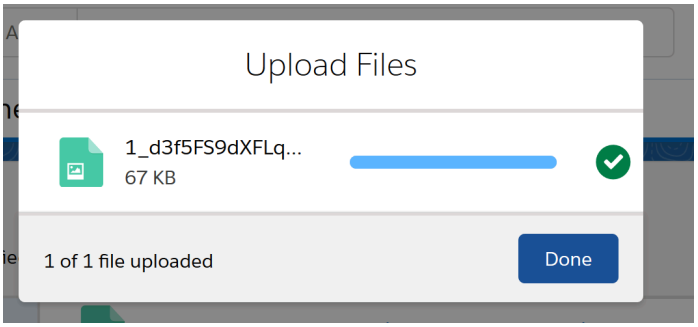


2. The “Files” Tab should open. If not, click on the “Files” tab.



3. Click on the “Upload Files” button on the right-hand side of the screen.

4. Choose the file from your computer and click “Open”.
5. This window will open when your file is uploaded:



6. Click “Done”. You can now use this image in the document.

Note: You will need the name of the image to reference it in your configurations.

Document Data Provider

In DocXpert, a Document Data Provider holds the information you would like to insert in your generated document. It uses SOQL queries to get the Salesforce data from the record(s) and insert it into your merge template, creating your finished document.

Examples:

- A “MyCompanyLogo” document data provider finds your logo to insert into the document.
  Use: universally on any document

- A “Quote” document data provider finds fields from the Quote and related records such as Contact or Opportunity.
  Use: for generating an offer based on a single Quote record

- “QuoteLineItems” finds data from a list of the Quote Line Item records, including Quote Line item and related Product information.
  Use: to generate a record list of Quote Line Items on a generated document

Although standard Document Data Providers are included in the DocXpert package, in most cases, you will need to **create your own customized Document Data Provider**. An easy way to do this is to create a new Document Data Provider, copy & paste the standard Document Data Provider into the SOQL query field, and add your org’s custom field(s) and/or adjust the query logic to your needs. **Only fields in your Document Data Providers can be referenced on your merge templates! If you use custom fields in your org, you must create your own providers.**

To create your own custom Document Data Providers, read the following information carefully to understand how Document Data Providers work, then follow the steps in this handbook.

Each data provider has a name, label, SOQL Query and the protected component option.

Label

- Will be shown in the UI for Admins.

Example: Quote

Document Data Provider Name

- This is the name used in the merge templates as a reference. This name must be included exactly in the template, for the document generation to work correctly.

Type
There are three “Type” choices:

- *Single Record*: used when the provider queries data for one record, such as an Account, Contact, Quote or Opportunity used for an offer or invoice.

  Note: When using this option, documents are generated, even if an image on the generated document fail. A placeholder of transparent pixels will be generated on the document where the image would have been. Therefore, documents should be checked thoroughly before sending them to customers!

- *Single Record (Image required)*: Same as “single record” with one important difference. This choice ensures that if an image on the merge document cannot be generated, an error occurs and the document generation as a whole fails. This ensures company standards when, for example, the logo or another image is required on a customer-facing document.

- *Record List:* used when generating a list of related records in a document, such as a list of Quote Line Items (products included on a Quote record) on a quote/offer document, or Order Line Items (products included on an Order record) on an order confirmation document.

Protected Component
If a developer releases protected custom metadata records in a managed package, access to them is limited. When limited, it can only be accessed through the package namespace. Click here for more information.

SOQL Query

Document Data Providers are written using an SOQL query. This is the most important part of the Document Data Provider, because this is what locates the exact information for your document. Queries can be used for records, images, users, or your company's information.

A query contains a few basic parts:

1. `SELECT` specifies what data (fields) are being referenced
2. `FROM` specifies the object the fields are on
3. `WHERE` specifies which exact record is being referenced

The query can be customized for your own document. Simply follow this format:

`SELECT` ***Fields*** `FROM` ***ObjectName*** `WHERE` ***Condition***

See the FROM ObjectName section of this handbook to understand which Document Data Providers you need for your custom document.

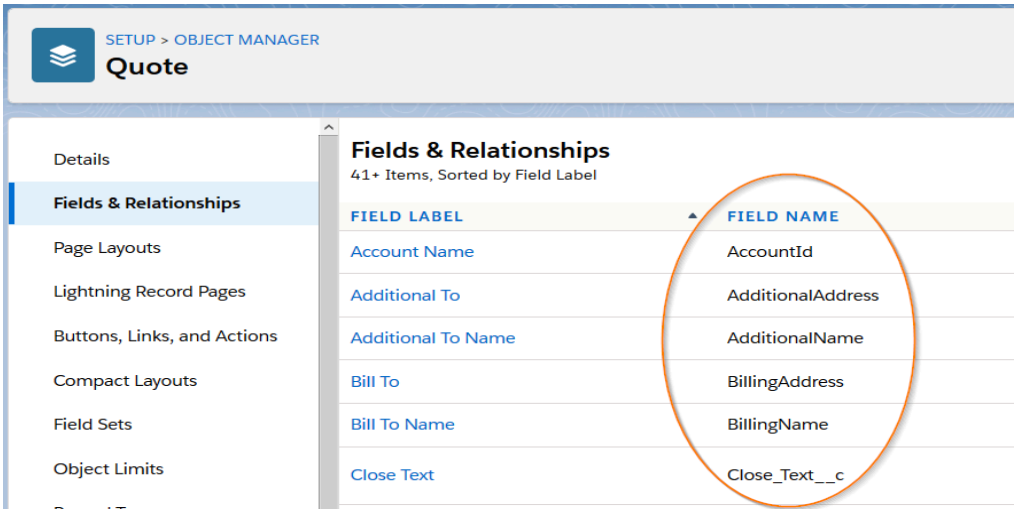
`SELECT` *Fields*

Add any fields you may need in the query after `SELECT` to the "***fields***" section. Be sure to separate the fields with a comma, like this:

```
SELECT QuoteNumber, Additional_Information__c, Contact.Name
```

Make sure the Field Name of the field is used in the query, not the Field Label, otherwise the query will fail. You can find the field name by clicking on:

Setup → Object Manager → *Your Object* → Fields & Relationships



More information on the syntax for simple fields, complex fields, custom fields, and custom relationships in a query, please read the Adding Fields section of this handbook.

For our Quote document example, we found the fields we need in the Before You Begin section of this handbook and wrote them in the “Field API-Name(s) in Salesforce” table.

These fields would then be written in our queries.

`FROM` *ObjectName*

In the `FROM` part of the query, you will need to determine which object level you should use for the query. You can do this by referencing the table in the Before You Begin section of this handbook. For our Quote document example, we created this table:

Information Needed on Document	Field API-Name(s) in Salesforce	Field is on which object in Salesforce?	Field is on which record in Salesforce?
company logo	MyCompanyLogo	Files	
customer address	BillingStreet BillingPostalCode BillingCity BillingState BillingCountry	Quote	quote record on which document is generated
quote number	QuoteNumber	Quote	quote record on which document is generated
customer name	Name	Contact (through standard Contact lookup field on quote)	contact record related to quote record on which document is generated
sales rep info: 1. sales rep name 2. phone number 3. email	 1. Name 2. Phone 3. Email	User (through standard Owner field on quote)	user record of Owner of the quote on which document is generated
product information: 1. name 2. description 3. product code 4. product image	 1. Name 2. Description 3. ProductCode 4. ProductPicture__c	Product2	product2 records associated with quote line items attached to the quote record on which

			document is generated
product information: 1. list price 2. quantity 3. total price	1. ListPrice 2. Quantity 3. TotalPrice	Quote Line Item	quote line items related to quote record on which document is generated
additional information text	AdditionalInformation__c	Quote	quote record on which document is generated

In the “Field is on which object in Salesforce?” column, we see that we have information on the Quote, Contact, User (Quote Owner), Quote Line Item and Product2 objects we would like to use, as well as a logo saved in the Files object.

Then we need to understand the relationships. The Quote record is related to the appropriate Contact record and User (Quote Owner) record through a lookup field on the Quote. To make sure the correct Contact and User records are referenced, we include the fields from these related records in a “Quote” Document Data Provider. The query would look like this:

```
SELECT BillingStreet, BillingPostalCode, BillingCity,
BillingState, BillingCountry, Number,
Additional_Information__c, Contact.Name, Owner.Name,
Owner.Phone, Owner.Email FROM Quote
```

The same is true for the Quote Line Items and Product2 objects. Because each Quote Line Item record is related to a specific Product2 record, we include the related product fields in a “Quote Line Item” Document Data Provider. The query would look like this:

```
SELECT ListPrice, Quantity, TotalPrice, Product2.Name,
Product2.Description, Product2.ProductCode,
Product2.ProductPicture__c FROM Quote Line Item
```

The syntax for simple fields, complex fields, custom fields, and custom relationships in a query is explained in the Adding Fields section of this handbook.

The company logo image is saved in the Files object, which is not directly related to the Quote or Quote Line Items objects, therefore we need a third Document Data provider for our logo. Please read the Image Example section of this handbook to find out how to query images.

Please reference the Sales Objects Data Model to view the relationships between objects in Salesforce.

`WHERE` *Condition*

In DocXpert, the WHERE condition often refers to the current record and is written like this:

```
WHERE Id=:recordId
```

":" means current. In the above case, it means current record id and refers to the record on which the document generation button is clicked.

Based on the Quote document example, the main object is the quote, which is why we place our document generation button on the Quote object. When the button is clicked, the information from that specific Quote record should be entered into the document.

For example, if you click a "generate document" button on the record "Quote #00012345", the field values of that specific record and its related records (in this case the Contact and Owner records) would be inserted into the generated document. Therefore, our query would reference the current record in the `WHERE` part of the "Quote" Document Data Provider query, as shown above.

The Quote Line Items Document Data Provider would also use the current record function in the `WHERE` clause, but would be written as follows:

```
WHERE QuoteId=:recordId
```

Because the button is clicked on the Quote record, not the Quote Line Items record, the Quote Line Item is not, itself, the current record. The Quote record is the current record. The Quote Line Item records are, however, related to the current record through a lookup field to the Quote. This lookup field stores the Id of the Quote. Therefore, we set `QuoteId` from this lookup field as the current record.

This means, that when the generate document button is clicked on "Quote #00012345", the information on the Quote Line Item records related to that specific quote record (as well as the information on the related Product2 records) will be inserted into the generated document.

The ":" can be used for the following:

- `:recordId`
- `:userId`
- `:userProfileId`
- `:userRoleId`
- `:userEmail`
- `:userLocale`
- `:userLanguage`
- `:orgId`

Note: if a record other than the current record should be used to generate information in your document, you will need to adjust the WHERE condition in your SOQL query to specify exactly which record you would like to have information from. This can become rather complex.

For further details on SOQL queries in Salesforce, please review the related documentation from Salesforce:
https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_select.htm

Note: You may also adjust the WHERE condition in your SOQL query to retrieve only certain records in a list. For example, in a table of your document, you may only want to generate Quote Line Items of a specific product family. In this case, your query would look like this:

SELECT Id, Product2.Family,... FROM QuoteLineItem
WHERE QuoteId=:recordId AND Product2.Family='*Product Family x*'

Note: You can use operators such as EQUAL and LIKE here.

Note that it is necessary to adhere exactly to the name of the data provider as they are case sensitive. To achieve an exact search result, please also use % as wildcards. An example can look like this:

SELECT Id, Product2.Family,... FROM QuoteLineItem
WHERE QuoteId=:recordId AND Product2.Name LIKE '%*Product Family x*%'

For further information, see the Troubleshooting Guide.

Adding Fields

It is important to keep the necessary syntax when adding fields to your data provider. The syntax is primarily based on where the field is located in relation to the object chosen for your data provider. Please note that the field's API Name is necessary for the correct syntax.

Field Syntax in Data Providers (Overview)

Type	Field Example (API Name)	Field Relationship*	Syntax in Data Provider
Standard Field	QuoteNumber	on data provider object	QuoteNumber
Custom Field	CustomerNumber__c	on data provder object	CustomerNumber__c
Standard Relationship Field	OpportunityName	on related Opportunity object (standard Lookup field)	Opportunity.Name
Custom Relationship Field	Example__c	on related object „XYZ“, related through a custom lookup field	XYZ__r.Example__c

*Examples are based on a data provider for the „Quote“ object

Standard Fields

Standard Salesforce fields from the object chosen for the data provider have the simplest syntax. To reference them in a Document Data Provider, only the field name (API name) is necessary.

Example:
When generating a quote document on a Quote record, you would use a data provider based on the Quote object. To use standard fields such as the Quote Number or Expiration Date fields, you would write them in your data provider as follows.

```
QuoteNumber
ExpirationDate
```

Note: Blank spaces, tabs or wrong letters could cause the query to fail.

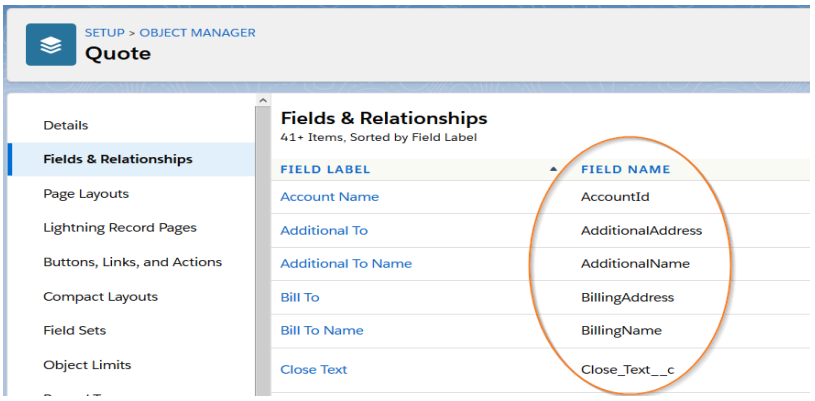
Custom Fields

Custom fields include the “ __c” suffix in the field’s API name.

The API Name of a custom field labeled “CustomerNumber__c” could therefore be `CustomerNumber__c`. This is exactly how the field should be written in your SOQL query, including the underscores and appendix “__c”.

Note that there are **two** underscores preceding the c in custom fields. If you only write one underscore “_c” instead of “__c” the query will fail.

Note: You may simply copy & paste the field from the relevant Document Data Provider as described under the Adding Fields section. You may also check the exact name of the field by navigating to the object in Setup and viewing the Field Name:



Standard Relationship Fields

The fields become more complex, when data on related objects is needed. There is a syntax difference between fields on objects related through a standard Salesforce relationship vs. a custom lookup field.

For example, when generating a document on a Quote record, you may need information from the Account, Contact, Opportunity or Opportunity Owner (User). These objects are related to the Quote object through a standard relationship. To retrieve the data, you need to scale the object relationship.

Example: name, phone and email from the Opportunity Owner related to the Quote

To reference the Owner information, which is stored on the related User object, a standard relationship is involved:

Quote record → Owner (User) record

Therefore, we need this field format in our SOQL query: *RelatedObject.Field*

In the above example, the fields in the SOQL query of your Document Data Provider would look like this:

Owner.Name
Owner.Phone
Owner.Email

In SOQL syntax, these fields would be included like this:

```
SELECT Owner.Name, Owner.Phone, Owner.Email FROM Quote WHERE Id = :recordId
```

Further examples, as seen in the table under Before You Begin: Fields are:

- The sales rep information such as the phone number is not on the quote itself, but found on the Quote Owner's User record.
- The product information is not on the Quote itself but on the related Quote Line Item records or even the Product records related to the Quote Line Item records
- You may also wish to reference the address of the Contact or Account record, instead of using the fields directly on the Quote record.

Some relationships between objects in Salesforce are direct and easy to find:

- Contacts are related to an Account. This means you can easily reference Account information from a Contact's record, or a list of contacts from your Account record.
- Quote Line Items are directly related to a Quote. This means you can easily create a list of quote line items from your Quote record.

Other objects are not directly related to each other in Salesforce and therefore require a workaround to include them in your query:

- Contacts are not directly related to Opportunities. They are related through a junction object called Contact Roles. Therefore, you cannot directly get Contact information from an Opportunity record, but instead need to reference the Contact Roles Object in a data provider.
- Campaigns are not directly related to Contacts. They are related through a junction object called Campaign Members.

Here you may want to create a custom lookup field to reference the Contact record. To use this custom relationship in your data provider, see Custom Relationship Fields.

Please reference the Sales Objects data model to see all relationships between related objects.

Custom Relationship Fields

When working with custom relationships (custom lookup fields), the related object reference in the merge field must include the "`__r`" suffix.

Example: custom lookup on a standard object to a standard object

You may have a custom lookup field on a standard object, such as Quotes, referencing another standard object, such as User. If this relationship was created with a *custom* lookup field (is not an out-of-the box standard Salesforce field), it is a custom relationship. This type of

relationship needs to be written with an "__r" syntax in both the Document Data Provider and merge template.

Let's say you would like to get the name of a user in your custom lookup field. This information is not stored in the field itself, but on the User record related through the custom lookup field. If the custom lookup field is named "Product Expert", you would use the API field name for your merge field (for example "ProductExpert__c"), but replace the "__c" with "__r" to show the custom relationship as follows:

```
ProductExpert__r.Name.
```

Note that there are **two** underscores preceding the r in custom relationship fields. If you only write one underscore "_r" instead of "__r" the query will fail.

Note: if you simply wanted to reference the value in the custom lookup field, in the above case it would be the Salesforce Id of the user, you are only referencing the value on the primary object and do not need to scale any relationships. Therefore, you would only need to write the field in your SOQL query like this: `ProductExpert__c.`

Example: custom lookup on a custom object to a standard object

On a custom object named Invoice, a custom lookup field references the Name field on the standard Account object. The relationship is a custom relationship, therefore the related object has a "__r" suffix and would look like this in your SOQL query: `Account__r.Name`

The appendix "__r" is always necessary when scaling the relationship through a custom lookup field. The "__r" is therefore also necessary for custom lookup fields on standard object to custom objects, as well as from custom objects to custom objects.

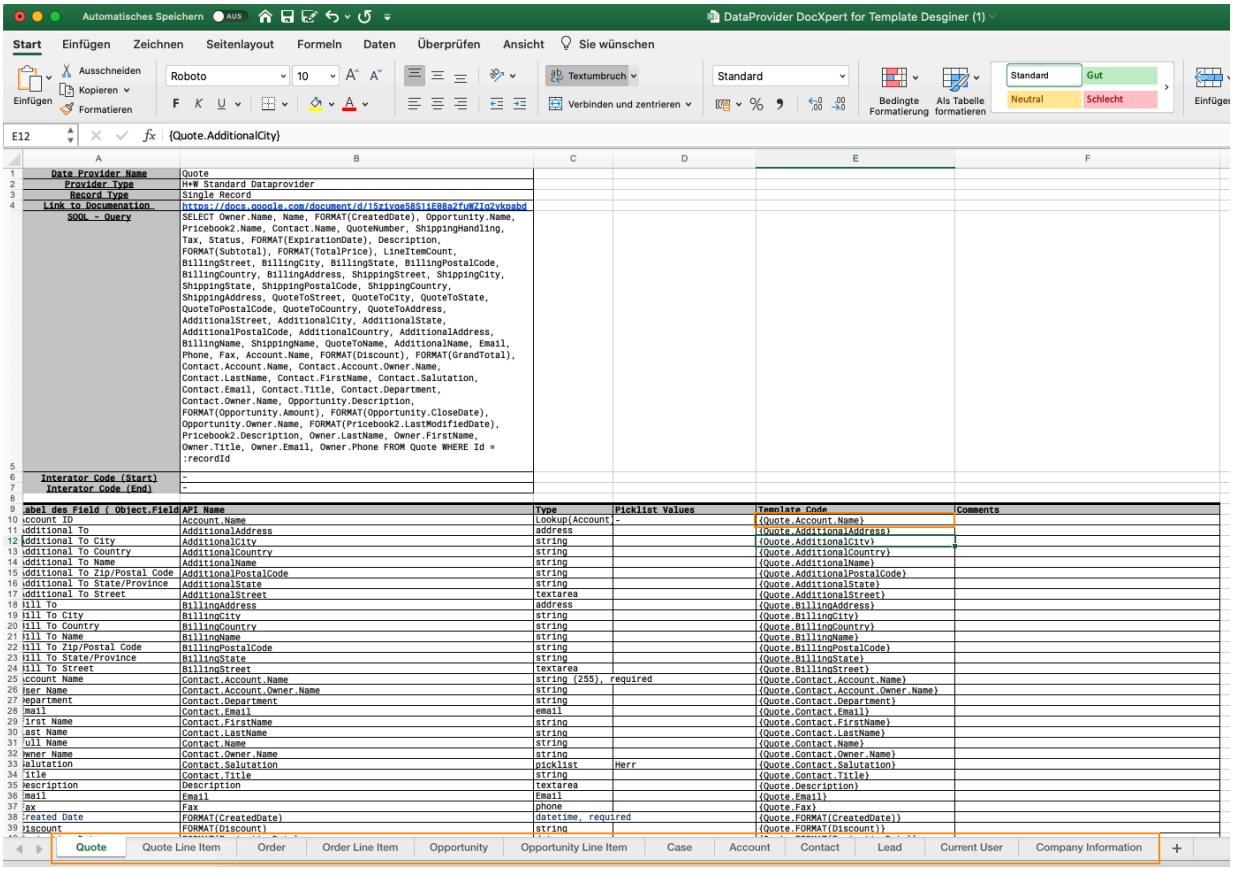
Excel Reference File

We provide you with an Excel reference file to give you an overview and make it easy to insert fields from the default Document Data Providers. The Excel also gives you the complete overview of which standard Document Data Providers come with the DocXpert package.

Data Provider

To create your custom Document Data Provider, copy & paste the whole SOQL query and add your custom fields, or copy & paste only select fields from the Template code column and write your query from scratch. Please also see the Standard Document Data Provider section in this handbook for more information.

When creating custom Data Providers with custom fields, we recommend updating the file and sharing the current version with the person responsible for creating the merge templates. This person will need to reference the fields exactly as they are written in the Document Data Provider.



	A	B	C	D	E	F
1	Date Provider Name	Quote				
2	Provider Type	H+W Standard Dataprovider				
3	Record Type	Single Record				
4	Link to Documenation	https://docs.google.com/document/d/15zivoe58S1iE88a2fuWZIq2vkoabd				
	SOQL - Query	SELECT Owner.Name, Name, FORMAT(CreatedDate), Opportunity.Name, Pricebook2.Name, Contact.Name, QuoteNumber, ShippingHandling, Tax, Status, FORMAT(ExpirationDate), Description, FORMAT(Subtotal), FORMAT(TotalPrice), LineItemCount, BillingStreet, BillingCity, BillingState, BillingPostalCode, BillingCountry, BillingAddress, ShippingStreet, ShippingCity, ShippingState, ShippingPostalCode, ShippingCountry, ShippingAddress, QuoteToStreet, QuoteToCity, QuoteToState, QuoteToPostalCode, QuoteToCountry, QuoteToAddress, AdditionalStreet, AdditionalCity, AdditionalState, AdditionalPostalCode, AdditionalCountry, AdditionalAddress, BillingName, ShippingName, QuoteToName, AdditionalName, Email, Phone, Fax, Account.Name, FORMAT(Discount), FORMAT(GrandTotal), Contact.Account.Name, Contact.Account.Owner.Name, Contact.LastName, Contact.FirstName, Contact.Salutation, Contact.Email, Contact.Title, Contact.Department, Contact.Owner.Name, Opportunity.Description, FORMAT(Opportunity.Amount), FORMAT(Opportunity.CloseDate), Opportunity.Owner.Name, FORMAT(Pricebook2.LastModifiedDate), Pricebook2.Description, Owner.LastName, Owner.FirstName, Owner.Title, Owner.Email, Owner.Phone FROM Quote WHERE Id = :recordId				
5						
6	Interator Code (Start)	-				
7	Interator Code (End)	-				
8						

9	abel des Field (Object.Field	API Name	Type	Picklist Values	Template Code	Comments
10	ccount ID	Account.Name	Lookup(Account	-	{Quote.Account.Name}	
11	dditional To	AdditionalAddress	address		{Quote.AdditionalAddress}	
12	dditional To City	AdditionalCity	string		{Quote.AdditionalCity}	
13	dditional To Country	AdditionalCountry	string		{Quote.AdditionalCountry}	
14	dditional To Name	AdditionalName	string		{Quote.AdditionalName}	
15	dditional To Zip/Postal Code	AdditionalPostalCode	string		{Quote.AdditionalPostalCode}	
16	dditional To State/Province	AdditionalState	string		{Quote.AdditionalState}	
17	dditional To Street	AdditionalStreet	textarea		{Quote.AdditionalStreet}	
18	ill To	BillingAddress	address		{Quote.BillingAddress}	
19	ill To City	BillingCity	string		{Quote.BillingCity}	
20	ill To Country	BillingCountry	string		{Quote.BillingCountry}	
21	ill To Name	BillingName	string		{Quote.BillingName}	
22	ill To Zip/Postal Code	BillingPostalCode	string		{Quote.BillingPostalCode}	
23	ill To State/Province	BillingState	string		{Quote.BillingState}	
24	ill To Street	BillingStreet	textarea		{Quote.BillingStreet}	
25	ccount Name	Contact.Account.Name	string (255),	required	{Quote.Contact.Account.Name}	
26	ser Name	Contact.Account.Owner.Name	string		{Quote.Contact.Account.Owner.Name}	
27	epartment	Contact.Department	string		{Quote.Contact.Department}	
28	mail	Contact.Email	email		{Quote.Contact.Email}	
29	irst Name	Contact.FirstName	string		{Quote.Contact.FirstName}	
30	ast Name	Contact.LastName	string		{Quote.Contact.LastName}	
31	ull Name	Contact.Name	string		{Quote.Contact.Name}	
32	wner Name	Contact.Owner.Name	string		{Quote.Contact.Owner.Name}	
33	alutation	Contact.Salutation	picklist	Herr	{Quote.Contact.Salutation}	
34	itle	Contact.Title	string		{Quote.Contact.Title}	
35	escription	Description	textarea		{Quote.Description}	
36	mail	Email	Email		{Quote.Email}	
37	ax	Fax	phone		{Quote.Fax}	
38	reated Date	FORMAT(CreatedDate)	datetime,	required	{Quote.FORMAT(CreatedDate)}	
39	iscount	FORMAT(Discount)	string		{Quote.FORMAT(Discount)}	

Quote | Quote Line Item | Order | Order Line Item | Opportunity | Opportunity Line Item | Case | Account | Contact | Lead | Current User | Company Information

Tools for SOQL Queries

When creating a custom Document Data Provider, you can quickly build and test your queries using the following tools:

- Workbench, for simple queries such as fields directly on the query object. https://workbench.developerforce.com/login.php
- Developer Console, for queries with object relationships such as Account fields on an Opportunity query. https://help.salesforce.com/articleView?id=code_dev_console.htm&type=5
- VisualStudio Code, for complex queries with auto-complete functionality. https://developer.salesforce.com/tools/vscode/

Record Field example

```
SELECT Id, Name, Account.Name, Contact.Name, BillingCity,
BillingCountry, BillingName, BillingPostalCode, BillingState,
BillingStreet FROM Quote WHERE (Id = :recordId)
```

This query is an example of what can be used for a “Quote” Document Data Provider.

It selects the id, name, name of associated account, name of related contact, and billing information fields of the Quote (`FROM Quote`), that uses the id of the record inserted into the query (`WHERE (Id = :recordId)`) through the custom document generation button.

Example File

You can find an example template file under this link:

http://hundw.com/example.docx

Important Annotation

DocXpert generally gets information from the bottom to the top of the structure.

Example when your structure is as follows:

(Top) Account => Opportunity => Quotes (Bottom)

If you generate the document from the Quotes, then you can get the information from the Opportunity and the Account, as well.

If you generate the document from the Account, you can only get the information from the Account and information in a record list of Opportunities or Quotes.

Therefore, you should start at the bottom most object.
It is, however, possible to make your own data providers to suit your needs.

Image example

```
SELECT getImageContent(Id) FROM ContentDocument WHERE Title =
'MyCompanyLogo'
```

The query finds the id of the image (`SELECT getImageContent(Id)`), which is stored as a Content Document under the “Files” tab (`FROM ContentDocument`), and has the title `'`MyCompanyLogo`'` (`WHERE Titel = 'MyCompanyLogo'`).

Note: Verify that your image is saved under the Files tab in Salesforce before creating the query, and that the name is written exactly the same as the title. Extra spaces or other variations can cause the query to fail. Visit the Before You Begin: Images section of this handbook for more information.

Image Size

All images in your document are automatically given a height of 5 centimeters if not otherwise specified. You can customize this default size, as well as adjust the size of each image individually.

Customizing Default Size for Images

To customize the default size for all images stored as ContentDocuments (uploaded to a record as a File), you need to customize the “Height” and “Width” fields in the Document Generator Settings custom metadata.

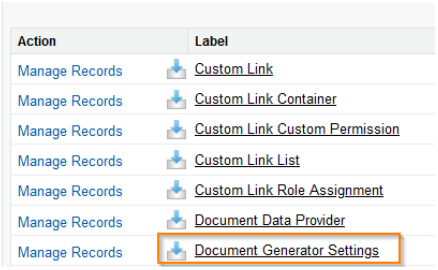
Important points:

- Adjusting the default size will *not* affect the size of images *already in your org*. When you customize the default size, it will *only adjust the size of images uploaded thereafter*. To adjust the images already in your org, you can adjust each image individually.
- If a newly uploaded image should have a different height/width than the default settings, you can adjust the height and width on the image individually after uploading it.
- The default settings automatically keep the aspect ratio of the images so that the image proportions of the newly uploaded images are not changed when generated. This means the picture will be scaled to be as large as possible within your given Default Image Width and Default Image Height.

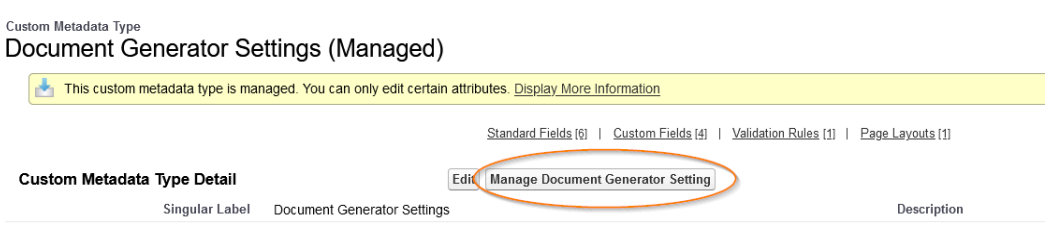
NOTE: To adjust an image individually, please see the next section “Customizing Individual Images” for more information.

Steps to adjust the default settings:

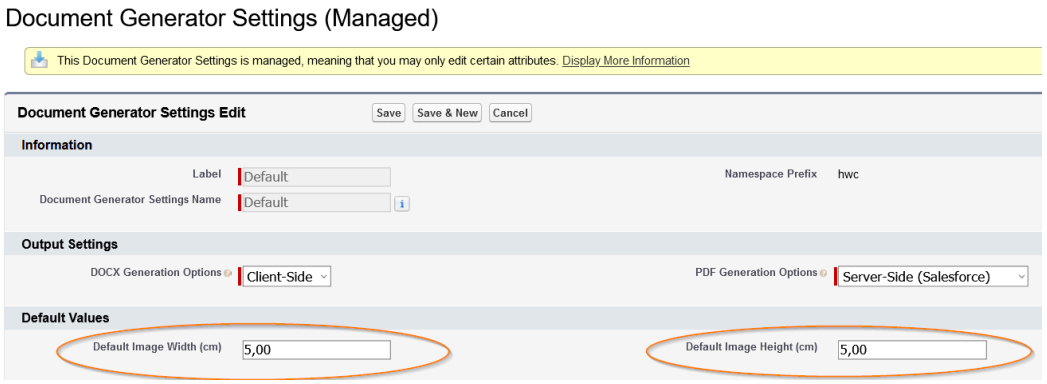
1. Navigate to Setup → Custom Metadata Types → Click on the Link “Document Generator Settings”



2. Click on the “Manage Document Generator Settings” button.



3. Click on the “Default” link in the Label column.
4. Click “Edit” on the Document Generator Settings (Managed) page and enter your custom default height and width.

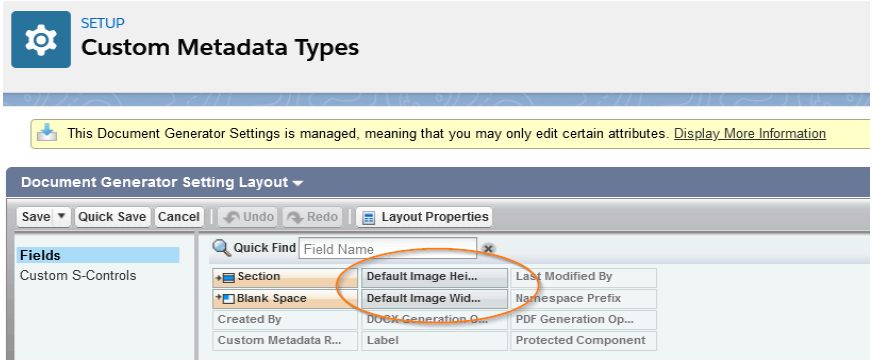


5. Click “Save”.

NOTE: If you are adjusting the default height and width for the first time and do not see the fields on your layout, you can bring them onto your layout using the following steps:

1. Navigate to Setup → Custom Metadata Types → Click on the Link “Document Generator Settings”
2. Scroll down to Page Layouts and click on “Edit” next to the “Document Generator Setting Layout”

3. In the editor, under “fields” find the “Default Image Height (cm)” and “Default Image Width (cm)” fields. Click and drag them onto the layout.

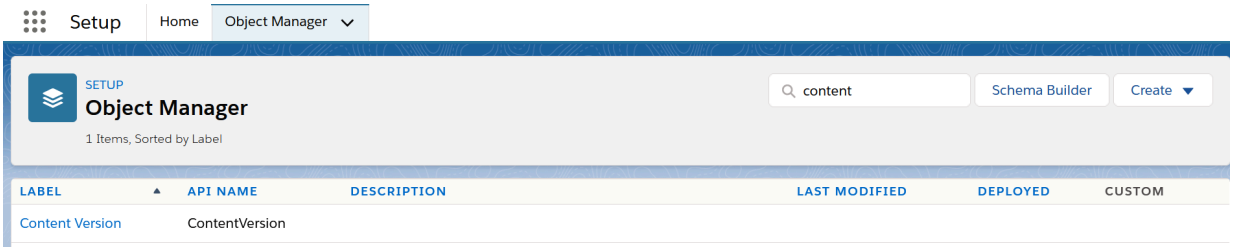


4. Click “Save”.

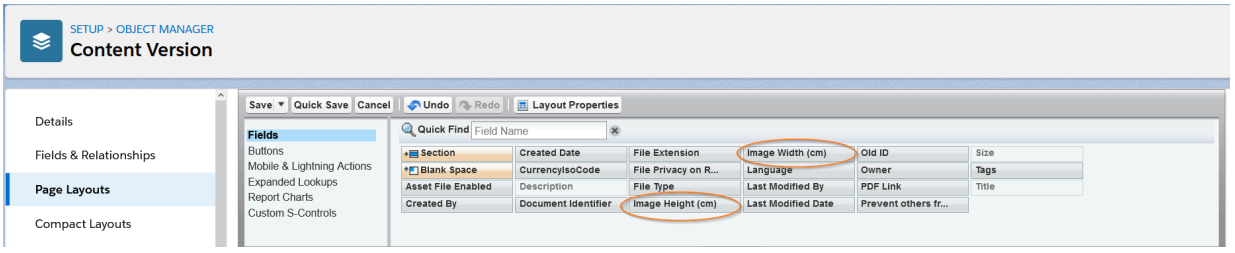
Customizing Individual Images

To customize the size of single images individually, add the “Image Height”, “Image Width” and optionally “Keep Image Aspect Ratio” fields on your page layout of the Content Version object, then edit the field on your image record.

1. Navigate to Setup → Object Manager → Content Version.

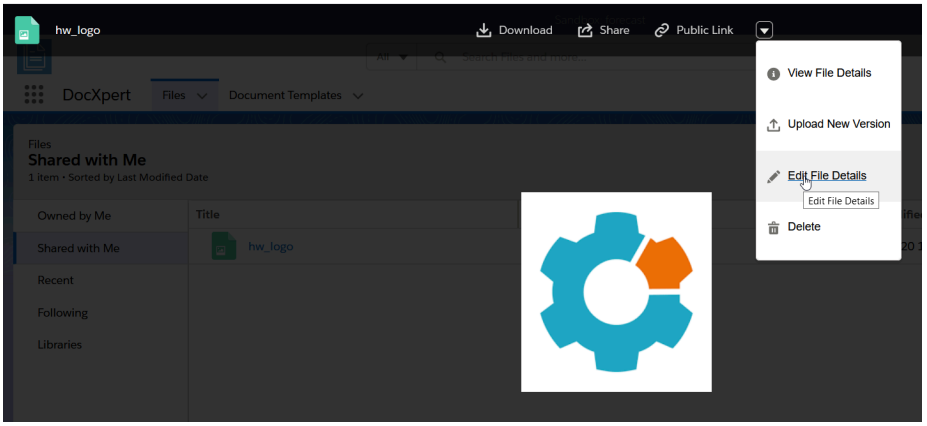


2. Click on page layout in the side menu and click on the relevant page layout. This opens the page layout editor.
3. In the editor, under “fields” find the “Image Height” and “Image Width” fields. You may also want to add the “Keep Image Aspect Ratio”. Click and drag them onto the layout.



4. Click “Save”.
5. Repeat these steps for all relevant page layouts.

6. Now you can enter a custom size for your images. To do this, edit the details of your image record.

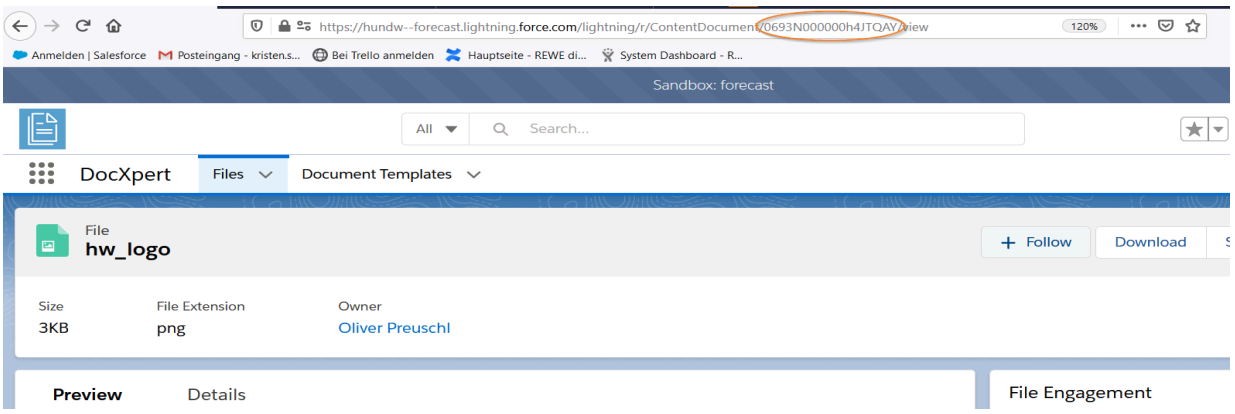


7. You should see the “Image Height (cm)” and “Image Width (cm)” fields, as well as the “Keep Image Aspect Ratio” Field.

   Note: When the “Keep Image Aspect Ratio” Field is activated, the height and width given is the maximum height/width for the generated image.

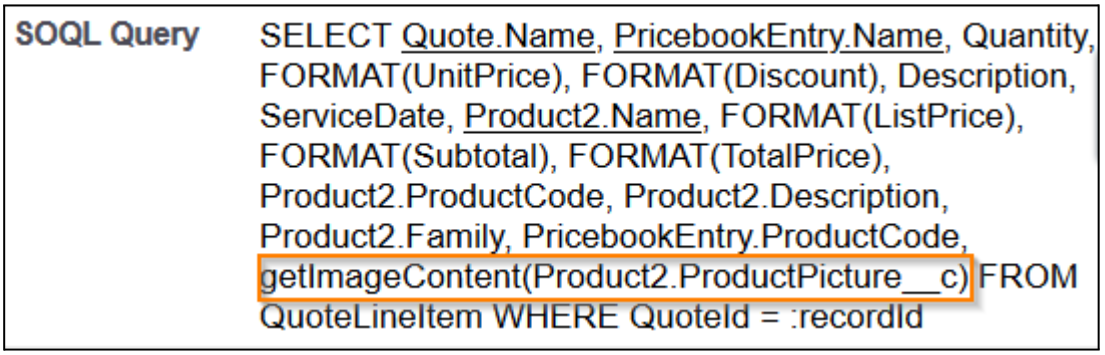
Images on a Record

To store an image on your record, such as a product image on your product record, write the ContentDocumentID in a custom text field. This is a workaround, since a lookup to an image is not currently possible in Salesforce. To find the SalesforeID of the image, navigate to your image in Salesforce, click on “View File Details”, and find the Id in the URL, as shown here:



To configure this workaround:

1. Create a custom text field (a field “ProductPicture__c” on your Product object, for example). Be sure to make this field editable and put on the appropriate page layout for the person responsible for adding the images to the records.
2. Upload the image to your Files tab as described in the Before You Begin: Images section of this handbook. You may also add the picture using your Files related list on your Product records, if the related list is on the layout.
3. Copy the Salesforce ID, navigate to your record, and enter the picture Id in the field.
4. Save the record.
5. Repeat steps 2-4 for each record which should have an image.
6. Now you can reference the image(s) in your Document Data Provider as shown here:



Images from Files Related List

It is possible to dynamically insert ContentDocument images attached to a record as “Files”. For example, at the end of your generated document, you may want to list all .jpeg Files attached to the record.

To do this, you need to:

1. Set up your Data Provider
2. Check the default size that should be used for your images.
3. Be sure to use the correct syntax in your Merge Template. For more information, please reference the ContentDocument Images section of the Template Creator Guide.

Set up your Document Data Provider as follows:

- Name your Data Provider
- Make sure it is Type “Record List”
- Create the SOQL Query in the following syntax:

```
SELECT getImageContent(ContentDocumentID)
```

This syntax uses the getImage function, just as you would with your other Data Providers.

```
FROM ContentDocumentLink
```

The Content Document Link Object is used to find the Content Documents.

```
WHERE (LinkedEntityId=:recordId) AND
ContentDocumentLink.ContentDocument.FileType IN ('JPEG')
```

This filters out which Content Documents you would like to use. It specifies only those on the current record (*=:recordId*) and specifies the Content Documents with a specific format (in this case `'JPEG'`. If you need further file types, simply add them after IN.
Example:

```
            IN ('JPEG','PNG')
```

- At the end, it should look similar to this:

Image Size:

The default image size for ContentDocument images is set to 5cm. If this suffices, there are no further steps to take.

The size of the image can be set on the Content Document itself. Please see Image Size in this document for more information.

User Example

```
SELECT id, Name, Email, Phone FROM User WHERE Id = :userId
```

This queries the user information of the person who generated the document. The above query finds the user id and its name, email and phone number (`SELECT` Id, Name, Email, Phone) of a user (`FROM` User) who is currently carrying out the action (`WHERE` Id = :userId).

Note: `:userId` is the global function that identifies the current user's id. The current user is the one carrying out the action. In our example, it is the person generating the quote by clicking the "generate" button. You may also set this to a static value, using a specific SalesforceID. For example:

```
WHERE Id = '2F0050O0230006XHSj'
```

For the full list of what can be marked current using ":" function, please review the SOQL Query section of this handbook.

FORMAT() Function

This function is used with custom number, date, time, or currency fields in a query to change the date/time format to the localized format. For example, the date in Germany is written in the format “day.month.year”, whereas the same date in the United States is written “month/day/year”. When the FORMAT() function is applied, the generator uses the locale of the person who generates the document.

Example: FORMAT(CreatedDate), FORMAT (Sales_Price)

Tip: More information on the format function can be found here: FORMAT ()

The FORMAT() function *cannot* be used together with the SUM() function in a template. Please make sure the fields used for the SUM() function do not use the FORMAT() function in your data provider.

Note: This function always uses the user’s format. To specify the format based on a specific locale, a “convertToLocale” function can be used. Please note that in order for the”convertToLocale” functions to work, the field cannot use the FORMAT() function in the Document Data Provider. For more information, see the “Date/Currency Conversion” section of the Template Creator Guide.

convertCurrency() Function:

When working in a multi-currency org, you may need to convert the currency itself. For example from USD to EUR. The Format() function only sets the “.” and “,” in the correct position (USD = $ 1,000.00 / EUR = 1.000,00€).

Tip: More information on the **convertCurrency()** function can be found here:
convertCurrency ()

Queries function very similarly to the Query Editor in the Developer Console.
To learn more about writing SOQL Queries in Salesforce, visit this trailhead:
Write SOQL Queries

Note: This function always uses the user’s format. To specify the format based on a specific locale, a “convertToLocale” function can be used.

Enable Multiple Currencies:

In Setup, enter Company Information in the Quick Find box, then select **Company Information** and click **Edit**. Ensure that your selected currency locale is the currency you would like to use as a default for current and future records. Enable **Activate Multiple Currencies**, then save your changes.

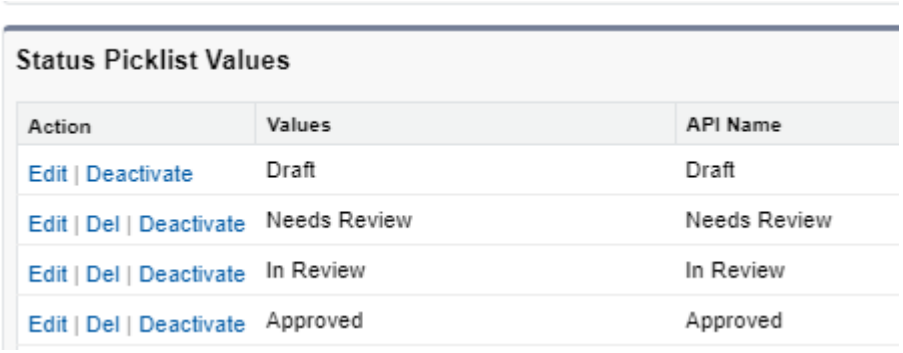
After enabling Multiple Currencies, users can select a currency in their user profile.

Picklist Value Labels

When using picklist values in DocXpert templates, the value's "API name" is generated as default, not the value's label ("Value"). In many cases, these two values are the same, but it depends on your configuration. You can check this if you notice unexpected generated values in your template. Simply check your picklist field:

Setup → Object Manager → Object (ex: Quote) → Field (ex: Status) → Status Picklist Values

In the following screenshot, the Values for the "Status" field are the same as the API Name:



Status Picklist Values

Action	Values	API Name
Edit | Deactivate	Draft	Draft
Edit | Del | Deactivate	Needs Review	Needs Review
Edit | Del | Deactivate	In Review	In Review
Edit | Del | Deactivate	Approved	Approved

Should the API Name be different and not what you would like to generate, you can use the TOLABEL function to generate the Values shown instead. This requires customizations in your data provider as well as in the template.

In the Data Provider, use the TOLABEL function: TOLABEL (Field Name) FieldNameLabel

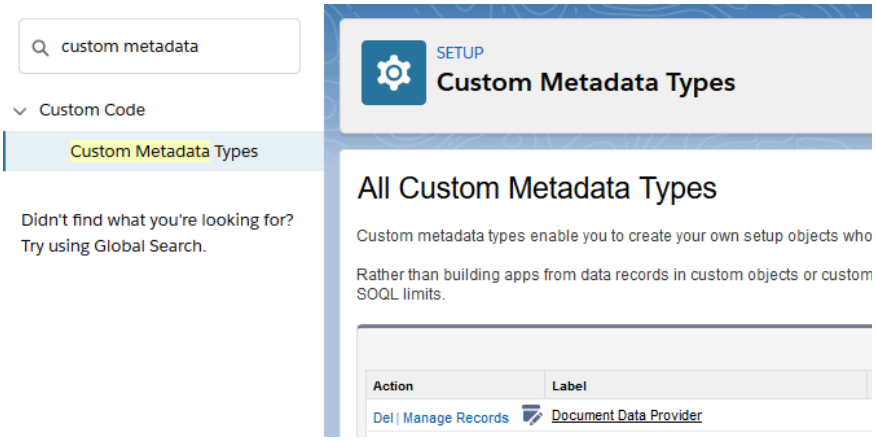
Example: TOLABEL(Status__c) StatusLabel

In the template, use the FieldNameLabel you specified in the data provider: {DataProvider.FieldNameLabel}

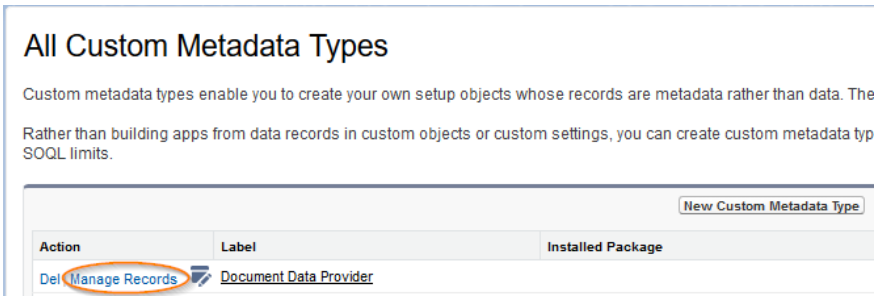
Example: {Quote.StatusLabel}

How to Create Document Data Providers

1. Navigate to Setup → click on “Custom Metadata Types”

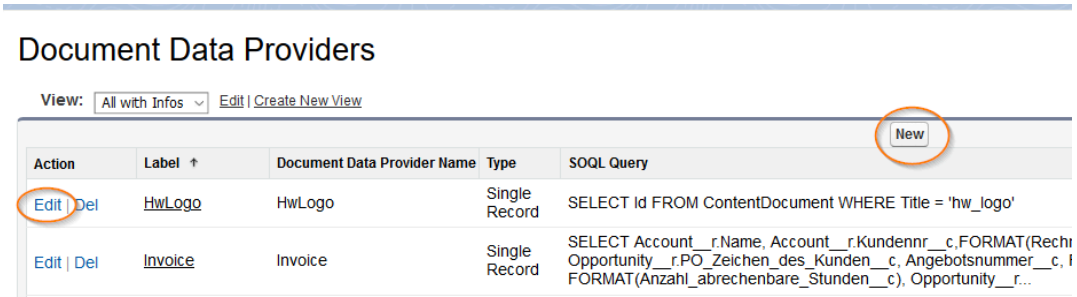


2. Find the Document Data Provider metadata type in the list.
In the Action column, click on “Manage Records”

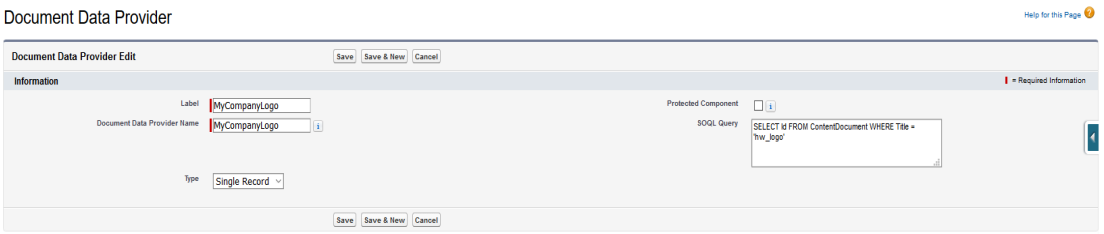


Note: if you accidentally click on the “Document Data Provider” link instead, you will come to a different screen. Simply press the “Manage Document Data Providers” button at the top of the page to continue to the next step.

3. Click on the “New” button to create a new Document Data Provider, or click on the “Edit” Link in the Action column to edit an existing Document Data Provider.



4. Enter the Label, Document Data Provider Name, Type, Protected Component (if desired) and enter your query to create a new Document Data Provider, or edit the area you would like to change on an existing Document Data Provider.



Note: Please review the information under Document Data Provider for what should be entered in the fields shown. Information specific to the SOQL Query in Document Data Providers can be found in the SOQL Query section of this handbook.

5. Press “Save” to complete the Document Data Provider.

Tip: When creating several Document Data Providers, click the “Save & New” button to directly create a second Document Data Provider.

Tip: A Document Data Provider can be edited at anytime.

This handbook provides you with standard Document Data Providers as a starting point. In most cases, you will need to modify these to fit your org. You can do this in a few ways:

- Simply copy and paste the SOQL queries from these standard providers and customize as needed. Please refer to the Standard Document Data Providers section for the queries.
- Refer to the queries and fields in the Excel Reference File that came with your package. All standard queries and included fields are listed here. For more information, please refer to the Excel Reference File section of this handbook.
- You can also create your SOQL query from scratch based on the fields and objects in your org.

Standard Document Data Providers

Your DocXpert installed package comes with the following standard document data providers:

- Quote
- QuoteLineItem
- Order
- OrderLineItem
- Opportunity
- OpportunityLineItem
- Case
- Account
- Contact
- Lead
- CurrentUser
- CompanyInformation

These are based on standard Salesforce objects using the standard Salesforce fields. Although they are not visible in your org, these standard providers are installed in the back-end of your org and can be used for document generation. Please be sure to check the standard queries to see if they include all the fields in the query logic necessary for your document generation.

If you have additional customized fields, custom objects or alternative logic you would like to use in your documents, first create your own data providers following this guide. The steps can be found in the How to Create Document Data Providers in this handbook.

Please note the following before customizing your data providers:

- Currency fields:
  All currency fields are converted to the correct user format through the FORMAT() function. However, if you are using a multi-currency org, please add the convertCurrency() function as well to convert the number.
  Example: `convertCurrency(FORMAT(Amount))`

- Owner fields:
  Only specific Owner fields can be used in a query. If you need additional fields from the User object of the record owner, create a separate document data provider which queries the User object based on the Owner in the current record. The query would be nested, such as this: `SELECT ….. FROM User WHERE id= (SELECT OwnerID FROM record WHERE id :recordid)`

- Some standard objects in Salesforce must be set up before use. For example, you can query Assets from a Case document data provider, but Assets are not on the Case Layout in a new standard org. Therefore, if you would like to expand the queries with

such objects, first be sure that the object is activated through the Setup Menu (if necessary), the lookup field which contains the object relationship is on the record layout (such as the Asset field on the Case page layout), and the field is viewable by the users (Field Level Security may be “hidden” per default and must be changed to at least “read” access on the relevant profiles).

NOTE: for more information on object relationships in the Salesforce, please reference the Sales Objects guide.

Here are the standard document data providers and their queries:

Quote

Contains standard Quote fields, as well as some fields from the related Contact and Owner (User) objects. Type = Single Record

```
SELECT IsSyncing, Owner.Name, Name, CreatedDate, FORMAT(CreatedDate)
FormattedCreatedDate, Opportunity.Name, Pricebook2.Name,
Contact.Name, QuoteNumber, ShippingHandling, Tax, Status,
ExpirationDate, FORMAT(ExpirationDate), FormattedExpirationDate,
Description, Subtotal, FORMAT(Subtotal) FormattedSubtotal,
TotalPrice, FORMAT(TotalPrice) FormattedTotalPrice, LineItemCount,
BillingStreet, BillingCity, BillingState, BillingPostalCode,
BillingCountry, BillingAddress, ShippingStreet, ShippingCity,
ShippingState, ShippingPostalCode, ShippingCountry, ShippingAddress,
QuoteToStreet, QuoteToCity, QuoteToState, QuoteToPostalCode,
QuoteToCountry, QuoteToAddress, AdditionalStreet, AdditionalCity,
AdditionalState, AdditionalPostalCode, AdditionalCountry,
AdditionalAddress, BillingName, ShippingName, QuoteToName,
AdditionalName, Email, Phone, Fax, Account.Name, Discount,
FORMAT(Discount) FormattedDiscount, GrandTotal, FORMAT(GrandTotal)
FormattedGrandTotal, Contact.Account.Name,
Contact.Account.Owner.Name, Contact.LastName, Contact.FirstName,
Contact.Salutation, Contact.Email, Contact.Title,
Contact.Department, Contact.Owner.Name, Opportunity.Description,
Opportunity.Amount, FORMAT(Opportunity.Amount)
FormattedOpportunityAmount, Opportunity.CloseDate,
FORMAT(Opportunity.CloseDate) FormattedOpportunityCloseDate,
Opportunity.Owner.Name, Pricebook2.LastModifiedDate,
FORMAT(Pricebook2.LastModifiedDate)
FormattedPricebook2LastModifiedDate, Pricebook2.Description,
Owner.LastName, Owner.FirstName, Owner.Title, Owner.Email,
Owner.Phone FROM Quote WHERE (Id = :recordId)
```

QuoteLineItem

Contains standard Quote Line Item fields, as well as some fields from the related Product2 and PricebookEntry objects. Type = Record List

```
SELECT Quote.Name, PricebookEntry.Name, Quantity, UnitPrice,
FORMAT(UnitPrice) FormattedUnitPrice, Discount, FORMAT(Discount)
FormattedDiscount, Description, ServiceDate, Product2.Name,
ListPrice, FORMAT(ListPrice) FormattedListPrice, SubTotal,
FORMAT(Subtotal) FormattedSubTotal, TotalPrice, FORMAT(TotalPrice)
FormattedTotalPrice, Product2.ProductCode, Product2.Description,
Product2.Family, PricebookEntry.ProductCode FROM QuoteLineItem WHERE
(QuoteId = :recordId)
```

Order

Contains standard Order fields, as well as some fields from the related Account, Contract, and Owner (User) objects. Type = Single Record

Note: You can also query further fields from the “original order” record, using the format: `OriginalOrder.`*`OrderFieldName`*. When doing this, be sure to use the exact field name! An example in the query below is `OriginalOrder.OrderNumber`.

```
SELECT Owner.Name, Account.Name, Pricebook2.Name,
OriginalOrder.OrderNumber, EffectiveDate, FORMAT(EffectiveDate)
FormattedEffectiveDate, EndDate, FORMAT(EndDate) FormattedEndDate,
Status, Description, CustomerAuthorizedBy.Name,
CompanyAuthorizedBy.Name, Type, BillingStreet, BillingCity,
BillingState, BillingPostalCode, BillingCountry, BillingAddress,
ShippingStreet, ShippingCity, ShippingState, ShippingPostalCode,
ShippingCountry, ShippingAddress, ActivatedDate,
FORMAT(ActivatedDate) FormattedActivatedDate, ActivatedBy.Name,
OrderNumber, TotalAmount, FORMAT(TotalAmount) FormattedTotalAmount,
CreatedDate, FORMAT(CreatedDate) FormattedCreatedDate, Account.Type,
Account.BillingStreet, Account.BillingCity, Account.BillingState,
Account.BillingPostalCode, Account.BillingCountry,
Account.BillingAddress, Account.ShippingStreet,
Account.ShippingCity, Account.ShippingState,
Account.ShippingPostalCode, Account.ShippingCountry,
Account.ShippingAddress, Account.Phone, Account.Fax,
Contract.StartDate, FORMAT(Contract.StartDate)
FormattedContractStartDate, Contract.EndDate,
FORMAT(Contract.EndDate) FormattedContractEndDate,
Contract.BillingStreet, Contract.BillingCity, Contract.BillingState,
```

```
Contract.BillingPostalCode, Contract.BillingCountry,
Contract.BillingAddress, Contract.ShippingStreet,
Contract.ShippingCity, Contract.ShippingState,
Contract.ShippingPostalCode, Contract.ShippingCountry,
Contract.ShippingAddress, Contract.ContractTerm,
Contract.Owner.Name, Contract.Status, Contract.CompanySigned.Name,
Contract.CompanySignedDate, FORMAT(Contract.CompanySignedDate)
FormattedContractCompanySignedDate, Contract.CustomerSigned.Name,
Contract.CustomerSignedTitle, Contract.CustomerSignedDate,
FORMAT(Contract.CustomerSignedDate)
FormattedContractCustomerSignedDate, Contract.SpecialTerms,
Contract.ActivatedBy.Name, Contract.ActivatedDate,
FORMAT(Contract.ActivatedDate) FormattedContractActivatedDate,
Contract.Description, Contract.ContractNumber,
Contract.LastApprovedDate, FORMAT(Contract.LastApprovedDate)
FormattedContractLastApprovedDate, Owner.LastName, Owner.FirstName,
Owner.Title, Owner.Email, Owner.Phone FROM Order WHERE
(Id=:recordId)
```

OrderLineItem

Contains standard Order Item fields, as well as some fields from the related Product2 and PricebookEntry objects. Type = Record List

Note: You can also query further fields from the “original order item” records, using the format: `OriginalOrderItem.Order.`*`FieldName`*. When doing this, be sure to use the exact field name! An example in the query below is `OriginalOrderItem.Order.OrderNumber`.

```
SELECT Product2.Name, Order.OrderNumber, PricebookEntry.Name,
OriginalOrderItem.Order.OrderNumber, AvailableQuantity, Quantity,
UnitPrice, FORMAT(UnitPrice) FormattedUnitPrice, ListPrice,
FORMAT(ListPrice) FormattedListPrice, TotalPrice, FORMAT(TotalPrice)
FormattedTotalPrice, ServiceDate, FORMAT(ServiceDate)
FormattedServiceDate, EndDate, FORMAT(EndDate) FormattedEndDate,
Description, CreatedDate, FORMAT(CreatedDate) FormattedCreatedDate,
OrderItemNumber, Product2.ProductCode, Product2.Description,
Product2.Family, PricebookEntry.ProductCode FROM OrderItem WHERE
(OrderId = :recordId)
```

Opportunity

Contains standard Opportunity fields, as well as some fields from the related Account and Owner (User) objects. Type = Single Record

```
SELECT Account.Name, Name, Description, StageName, Amount,
FORMAT(Amount) FormattedAmount, CloseDate, FORMAT(CloseDate)
FormattedCloseDate, Type, NextStep, LeadSource, Pricebook2.Name,
Owner.Name, LastActivityDate, FORMAT(LastActivityDate)
FormattedLastActivityDate, FiscalQuarter, FiscalYear, Fiscal,
Account.Type, Account.BillingStreet, Account.BillingCity,
Account.BillingState, Account.BillingPostalCode,
Account.BillingCountry, Account.BillingAddress,
Account.ShippingStreet, Account.ShippingCity, Account.ShippingState,
Account.ShippingPostalCode, Account.ShippingCountry,
Account.ShippingAddress, Account.Phone, Account.Fax, Owner.LastName,
Owner.FirstName, Owner.Title, Owner.Email, Owner.Phone FROM
Opportunity WHERE (Id = :recordId)
```

OpportunityLineItem

Contains standard Opportunity Line Item fields, as well as some fields from the related Product2 and PricebookEntry objects. Type = Record List

```
SELECT Opportunity.Name, ProductCode, Name, Quantity, TotalPrice,
FORMAT(TotalPrice) FormattedTotalPrice, UnitPrice, FORMAT(UnitPrice)
FormattedUnitPrice, ListPrice, FORMAT(ListPrice) FormattedListPrice,
ServiceDate, FORMAT(ServiceDate) FormattedServiceDate, Description
FROM OpportunityLineItem WHERE (OpportunityId = :recordId)
```

Case

Contains standard Case fields, as well as some fields from the related Account, Contract, and Owner (User) objects. Type = Single Record

```
SELECT CaseNumber, Contact.Name, Account.Name, SuppliedName,
SuppliedEmail, SuppliedPhone, SuppliedCompany, Type, Status, Reason,
Origin, Subject, Priority, Description, ClosedDate, OwnerId,
CreatedDate, CreatedById, LastModifiedDate, FORMAT(LastModifiedDate)
FormattedLastModifiedDate, LastModifiedBy.Name,
FORMAT(LastModifiedBy.Name) FormattedLastModifiedByName,
ContactPhone, ContactMobile, ContactEmail, ContactFax, Comments,
Account.Type, Account.BillingStreet, Account.BillingCity,
Account.BillingState, Account.BillingPostalCode,
Account.BillingCountry, Account.BillingAddress,
Account.ShippingStreet, Account.ShippingCity, Account.ShippingState,
Account.ShippingPostalCode, Account.ShippingCountry,
```

```
Account.ShippingAddress, Account.Phone, Account.Fax,
Contact.Account.Name, Contact.LastName, Contact.FirstName,
Contact.Salutation, Contact.OtherStreet, Contact.OtherCity,
Contact.OtherState, Contact.OtherPostalCode, Contact.OtherCountry,
Contact.OtherAddress, Contact.MailingStreet, Contact.MailingCity,
Contact.MailingState, Contact.MailingPostalCode,
Contact.MailingCountry, Contact.MailingAddress, Contact.Phone,
Contact.Fax, Contact.MobilePhone, Contact.HomePhone,
Contact.OtherPhone, Contact.AssistantPhone, Contact.Email,
Contact.Title, Contact.Department, Contact.AssistantName,
Contact.Owner.Name, Owner.LastName, Owner.FirstName, Owner.Title,
Owner.Email, Owner.Phone FROM Case WHERE (Id = :recordId)
```

Account

Contains standard Account fields, as well as some fields from the related Owner (User) object. Type = Single Record

```
SELECT Name, Type, BillingStreet, BillingCity, BillingState,
BillingPostalCode, BillingCountry, BillingAddress, ShippingStreet,
ShippingCity, ShippingState, ShippingPostalCode, ShippingCountry,
ShippingAddress, Phone, Fax, Website, Industry, AnnualRevenue,
NumberOfEmployees, Description, Owner.Name, Owner.LastName,
Owner.FirstName, Owner.Title, Owner.Email, Owner.Phone FROM Account
WHERE (Id = :recordId)
```

Contact

Contains standard Contact fields, as well as some fields from the related Account and Owner (User) objects. Type = Single Record

```
SELECT Account.Name, LastName, FirstName, Salutation, Name,
OtherStreet, OtherCity, OtherState, OtherPostalCode, OtherCountry,
OtherAddress, MailingStreet, MailingCity, MailingState,
MailingPostalCode, MailingCountry, MailingAddress, Phone, Fax,
MobilePhone, HomePhone, OtherPhone, AssistantPhone, Email, Title,
Department, AssistantName, Owner.Name, LastModifiedDate,
FORMAT(LastModifiedDate) FormattedLastModifiedDate,
LastModifiedBy.Name,LastActivityDate, FORMAT(LastActivityDate)
FormattedLastActivityDate, EmailBouncedReason, EmailBouncedDate,
FORMAT(EmailBouncedDate) FormattedEmailBouncedDate, Account.Type,
Account.BillingStreet, Account.BillingCity, Account.BillingState,
Account.BillingPostalCode, Account.BillingCountry,
Account.BillingAddress, Account.ShippingStreet,
```

```
Account.ShippingCity, Account.ShippingState,
Account.ShippingPostalCode, Account.ShippingCountry,
Account.ShippingAddress, Account.Phone, Account.Fax, Owner.LastName,
Owner.FirstName, Owner.Title, Owner.Email, Owner.Phone FROM Contact
WHERE (Id = :recordId)
```

Lead

Contains standard Lead fields, as well as some fields from the related Owner (User) object. Type = Single Record

```
SELECT LastName, FirstName, Salutation, Name, Title, Company,
Street, City, State, PostalCode, Country, Address, Phone, Email,
Description, LeadSource, Status, Industry, Rating, AnnualRevenue,
NumberOfEmployees, Owner.Name, EmailBouncedReason, EmailBouncedDate,
Owner.LastName, Owner.FirstName, Owner.Title, Owner.Email,
Owner.Phone FROM Lead WHERE (Id = :recordId)
```

CurrentUser

Contains standard User fields of the user who clicks on the “generate document” button. Type = single record

```
SELECT Username, LastName, FirstName, Name, CompanyName, Division,
Department, Title, Street, City, State, PostalCode, Country,
Address, Email, Signature, StayInTouchSubject, StayInTouchSignature,
StayInTouchNote, Phone, Fax, MobilePhone, Contact.Name,
Account.Name, Extension FROM User WHERE (Id=:userId)
```

CompanyInformation

Contains standard fields of the Organization object, which stores your company profile information in your Salesforce Org. Type = Single Record

```
SELECT Name, Division, Street, City, State, PostalCode, Country,
Address, Phone, Fax, PrimaryContact, FiscalYearStartMonth FROM
Organization
```

Document Template Customizations

Document Templates are stored as records on the custom Document Template object. Therefore, you can create page layouts and assign them to users, add fields to the layout, create validation rules, and adjust sharing settings as usual in Salesforce.

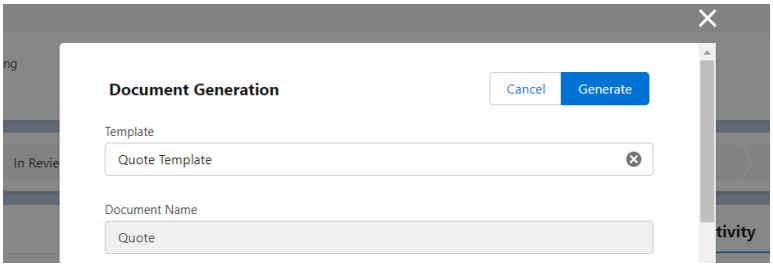
There are also optional fields that may be useful for your users, which first need to be put onto the page layout when setting up DocXpert for the first time:

Deactivate Versioning
When this checkbox is activated, a new document will be generated instead of a new version.

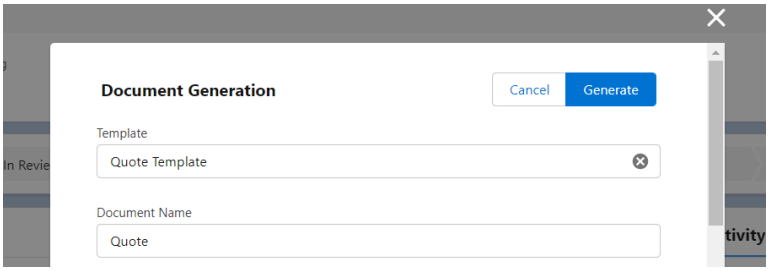
NOTE: On the Quote object, this setting only deactivates the versioning on generated Word Documents, not for PDFs. PDFs on Quotes will continue to be generated as a new version. This is due to the Salesforce data structure and cannot currently be changed.

Allow Document Name Change
Activating this checkbox allows the user generating the document to overwrite the name when generating the document. The name will be pre-filled with the “Document Base Name”, but can be changed by the user.

In this screenshot, the checkbox is deactivated/not in use, so the name cannot be changed. This is the default setting:

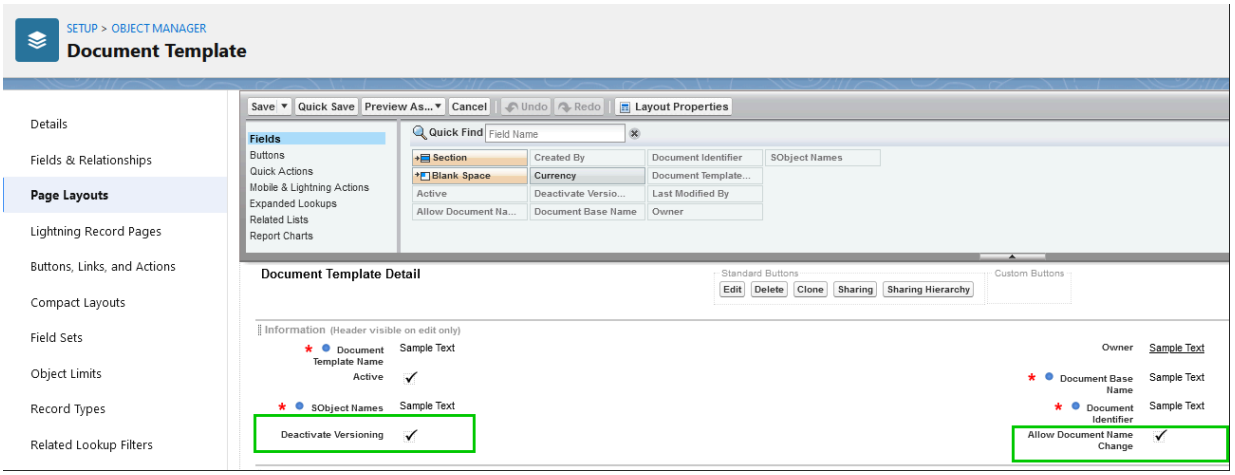


In this screenshot, the checkbox is on the layout and activated. The name can be changed:



If not on the page layout, add the optional fields by following these steps:

1. Navigate to Setup, and click on the Object Manager.
2. In the Object Manager, find and click on the Document Template object.
3. Click on Page Layouts and choose the page layout being used.
4. Drag the optional field(s) you'd like to use to the layout and click "Save".
5. These fields will now be shown on all Document Templates using that page layout.



Note: If users cannot see the new fields on the page layout, please check the Field Level Security for these fields.

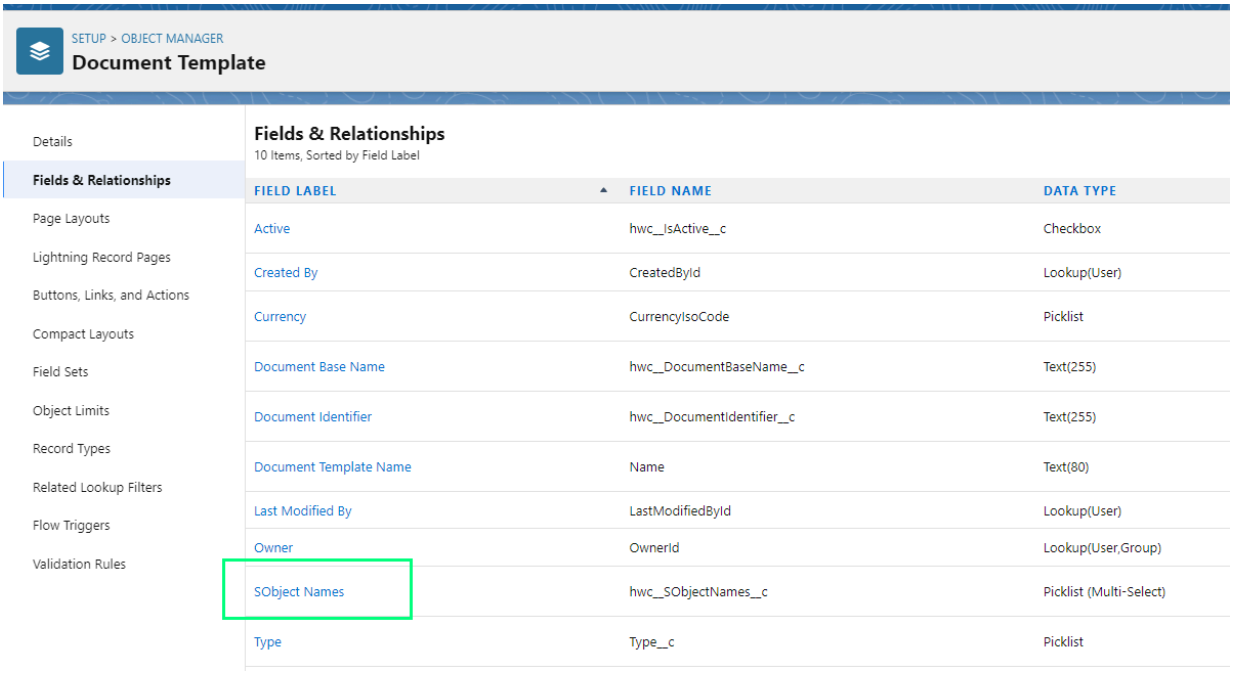
Set Up User Access

Document Templates are stored as records on the Document Template object. Therefore, determine which templates should be used by which users and adjust the sharing settings accordingly as described below.

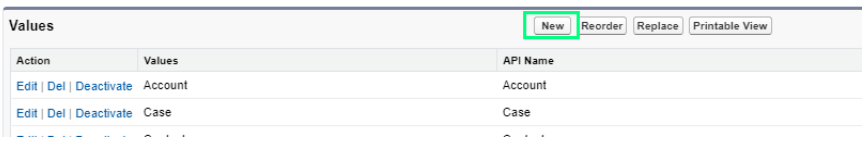
When users click on the “generate document” button, they only see the templates they have access to. Two main areas control this access:

1. SObject field on the Document Template record:
   When a Document Template record is created (not the template itself, but the record where the template is stored), one or more objects can be chosen in the SObject field. This controls which object(s) the template can generally be used for. This is usually done by the person responsible for the merge template.

   NOTE: If you want your Documents available for your custom objects, you will need to add them to the SObject Picklist on the “Document Template” Object in Setup:



   In the SObject Names field, in the Values section, click “New” to add the name of your Object:



2. Document Template record access:
   Access is controlled using the Document Template object. To set up sharing, please follow the Salesforce sharing model, including CRED object settings, the organization-wide default settings, sharing rules, and - if relevant - manual sharing. For more information on the Salesforce sharing model, visit the Control Who Sees What documentation.

Note: To generate documents, the object settings on user's profiles must be at least Read. Also, the default Organization-Wide Defaults for Document Templates is public read/write.

Examples

1. Sales Reps may have access to the quote template, while Marketing users may only have access to templates for accounts or contacts.

2. In an international org, you may have the same template in various languages. Set up sharing to give users access to only the languages they need in their region. For example, the North American Sales Rep profiles can only access English, French and Spanish templates, whereas Asian Sales Rep profiles can access English and Chinese templates, and all non-sales profiles have no access to templates at all.

Note: You can add custom fields to your Document Template object to use for criteria-based sharing rules. For example, add a custom language field to filter templates for specific groups of users based on template language.

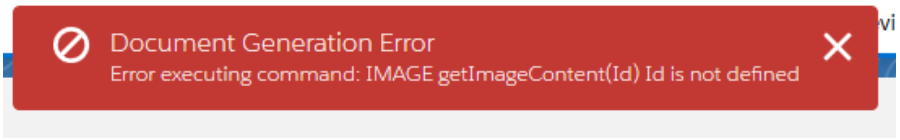
Testing & Deployment

Testing

Step 1: Test as Administrator

After configuration, be sure to test your document generation before deploying it to users. Test it first as a system administrator to be sure it generally works. Find your custom button on a record of your choice, use it to generate your document, and check to see if everything looks as planned. If not, troubleshoot:

- If you do not see your template, verify that the sharing settings are correct and your template is active. (The active checkbox must be checked on the Document Template record).
- If the document cannot be generated, and you receive an error message, troubleshoot depending on the error. For example, if your error message looks like this,



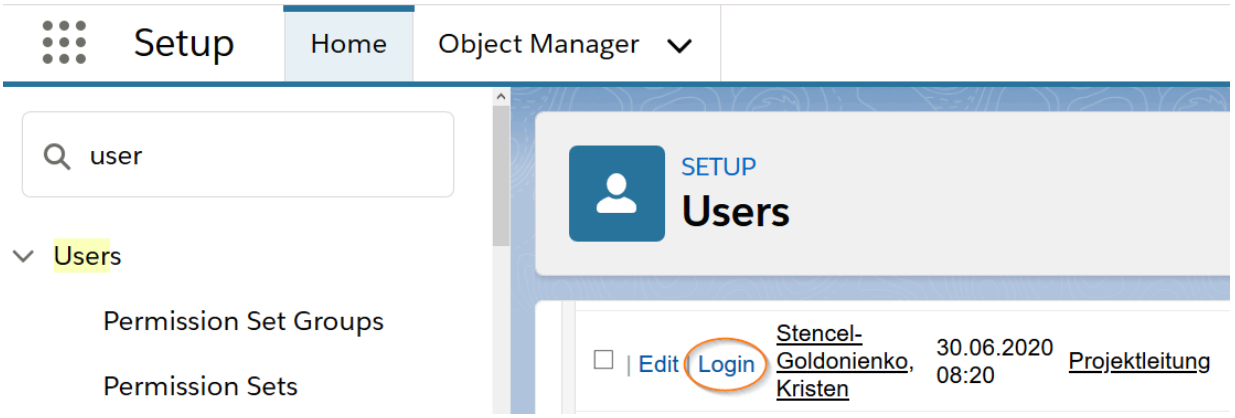
  you know there is an image error. In this specific case, the Document Data Provider name was not inserted correctly in the Image function on the merge template.
- If your document shows a blank space where you expected a merge field text, check that the field is included in your Document Data Provider and that your merge field is referenced correctly on your merge template.
- If the static text in your document doesn't look as you expected, check your merge template text for errors.

Once you have created your document successfully from a system administrator standpoint, you are ready to test as a user.

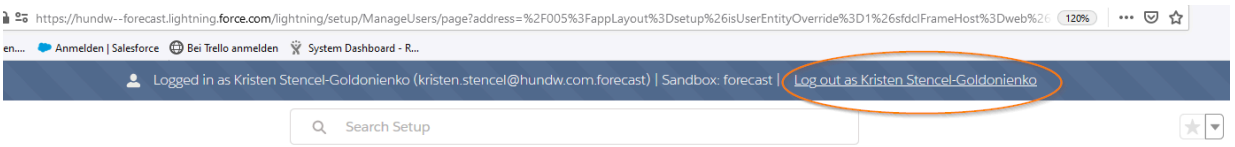
Step 2: Test as User

Since system administrator profiles typically have more access and rights than normal users, we recommend to then test as your users in different profiles. For example, if you are deploying DocXpert to your Sales Reps and Marketing Managers, test using users with these profiles. This can only be done by a system administrator if the "login as user" function is turned on in your sandbox.

Go to Setup → Users → and click the “Login” link next to the user’s name you wish to test with:



This logs you into the sandbox as that person, so you can generate a document using their role and profile. After testing, log yourself out of that person’s view by clicking on the “log out of username” link, as shown here:



If you need to troubleshoot your configurations, you will need to log out of that person’s view, troubleshoot, then log in again using the login link for that person to test again.

Step 3: User Acceptance Tests

After testing the functionality to ensure everything works from a technical perspective, you can run user acceptance tests (UAT) with a group of users. Be sure to include users who represent the various types of users who will be ultimately be generating documents, for example Sales Reps, Sales Managers, Marketing Managers, etc.

After showing your UAT user group how to use the new functionality, these users should test the document generation and report any technical bugs they may find, as well as give feedback on the usefulness of the document generation from a practical perspective. This could include feedback such as:

- Can they locate the custom button?
- Can they find the template they want to generate easily?
- Does the template include the information they would expect or find useful, or is something important missing?

Step 4: Make Adjustments

After the UATs, adjust your configuration according to the feedback you received. This may include, but is not limited to:

- Creating new fields to include in your Document Data Provider and merge templates
- Adjusting sharing settings
- Adjusting your merge template
- Renaming or changing the placement of your custom button

Step 5: Re-tests

After adjusting your configuration based on UAT feedback, be sure to re-test your configuration as an Admin, as the users, and/or in a second round of User Acceptance Tests. If the tests are successful, you are ready to deploy!

Preparing Your Users

Before deploying the Document Generator for your users, it is important to communicate the coming change to them and make sure they feel comfortable using the functionality. There are many ways to do this.

Important is to let them know:

- When the functionality will be live for them
- How to use the functionality

The best way of ensuring user acceptance from the start is to train the users by giving them a short e-learning module or hosting a short remote training sessions, depending on the impact the generator has on your users. The “training” can also be worked into other topics or a regularly occurring standard meeting, such as a weekly jour fixe.

At the very least, the users should have access to a user’s guide, e-learning module or other step-by-step document in case of any questions. The steps in this documentation under the User Guide can be used as a template. Be sure to include who the users can turn to for support or constructive feedback in your user’s guide.

Smaller companies may deploy the generator for all users at once, while larger companies may consider rolling out the generator to one group at a time, for example certain departments, countries, or groups of power users across the company. If this is the case, be sure to plan enough time for the users to start using the functionality, give feedback, and for you to make adjustments before rolling out to the next group.

Also consider the following to get your users excited about the new generator:

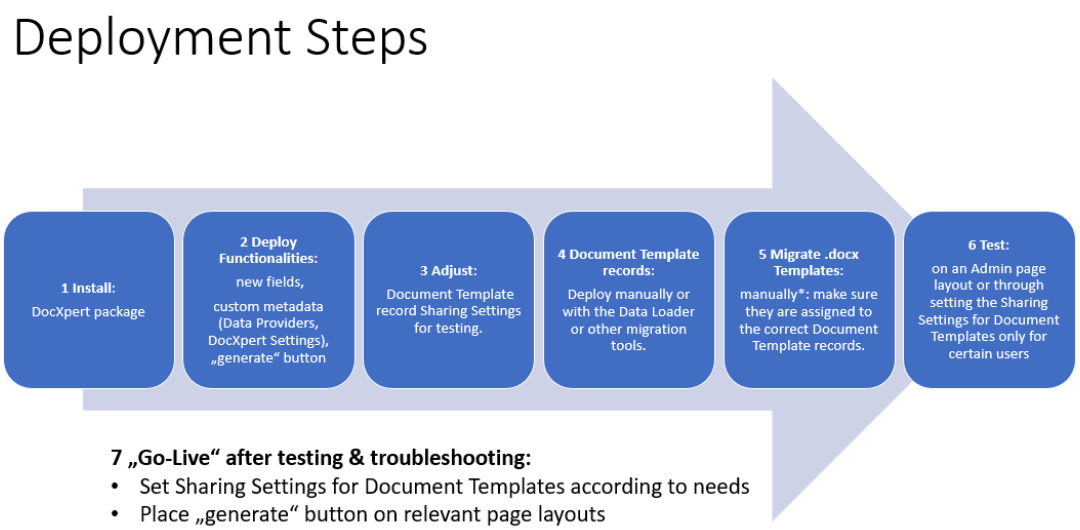
- Mention it in meetings/emails/Chatter leading up to the deployment date.
- Explain the benefits for the users, such as the time or clicks it saves them.
- Before rolling out to all users, give the Document Generator to selected power users for them to try out and spread the word.

Technical Deployment

Our recommendation is to configure the Document Data Generator in your organization's developer / test Sandbox, then deploy it to your live system.

Preparation: Sandbox Testing

Please be sure to install, configure and test DocXpert in a Sandbox before deploying it to your live environment! For more detailed information, please refer to the Testing section of this handbook.



Step 1: Install

Please install the DocXpert package "for All Users" in your live environment.
Alternatively, you can install "for Administrator only" for testing. For Go-Live, you will then need to assign the DocXpert license and permission sets to all DocXpert users, so they can generate documents. It is possible to do this on a one-off basis for testing purposes. Please refer to the section Licenses & Permission Sets in this handbook for more information.

Step 2: Deploy Functionalities

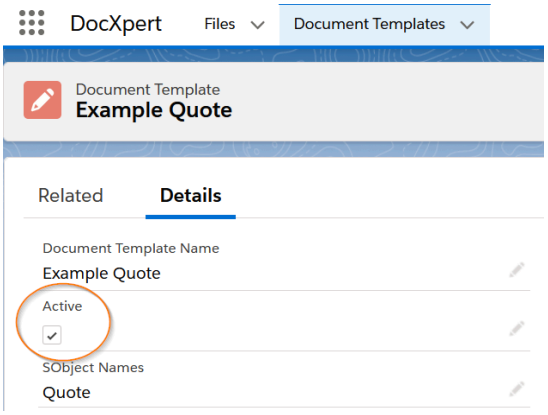
Deploy your functionalities. Please make sure to move all new fields or images to your live org first, since these are referenced in your data providers and templates. Then you can also deploy the DocXpert metadata including settings, data providers, as well as the "generate" button(s) on your object(s).

Step 3: Adjust Document Template Sharing (optional)

If you are planning to test in your live org first before deploying to all users, you can adjust your Document Template sharing settings / sharing rules accordingly to restrict users from generating documents. Alternatively, simply give the DocXpert license and permission sets to only the users who will be testing.

Step 4: Deploy Document Template Records

First deploy your Document Template records. You can do this manually or with the Data Loader or other migration tools. Verify that your templates are active, by checking the active checkbox on the Document Template record, otherwise it will not be visible for use! The active checkbox should look like this:



Step 5: Migrate .docx Templates

Once your Document Template records are in the live org, you can move your corresponding .docx templates to live by uploading them to the correct Document Template records. This is usually a manual process.

Note: DocXpert currently comes with the Import/Export (Beta) function to move DocXpert Metadata (settings & data providers) as well as Document Templates (records & templates) to the live system all at once. This would cover those functionalities from Steps 2, 4 & 5. Please note, this tool is only beta! For more information, see Import/Export Wizard Beta below.

Step 6: Test (in Live System)

Before deploying to all users and making the “generate” button visible for everyone, we recommend first testing all templates as an administrator on an admin-only page layout to make sure nothing has been forgotten in the live system.

Go-Live: Deploy to Users

After general testing, place your custom button(s) on the intended page layouts and make sure the sharing settings are given appropriate access to your users. Be sure to do this as the

final step. Once the button is on the layout, sharing is set, and users are assigned licenses & permission sets, users can use DocXpert immediately. We recommend doing this in off-hours (such as after 6pm or on the weekend) and communicating that the new functionality will be available at the start of the following work day.

See the section in this handbook on Set Up User Access for more information.

Import/Export Wizard BETA

For an easier migration of document templates and data providers between environments (such as deployment from Sandbox to Live), DocXpert now offers the Import/Export Wizard (BETA).

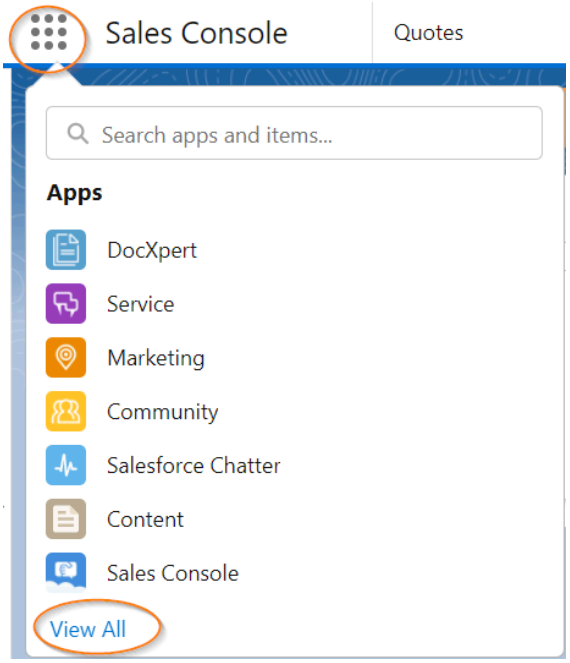
Note: As it is only Beta, please use the wizard with caution and check the migrated data providers and document templates after using it.

To use the Import/Export Wizard, follow these steps.

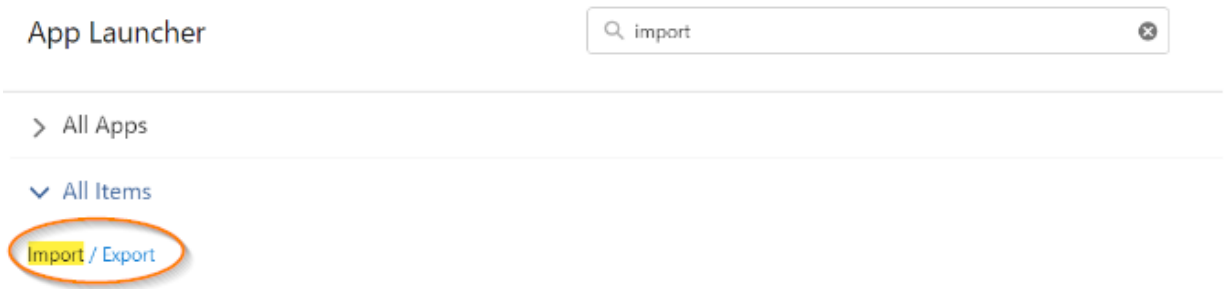
Export

In the org you need to export form (Sandbox org):

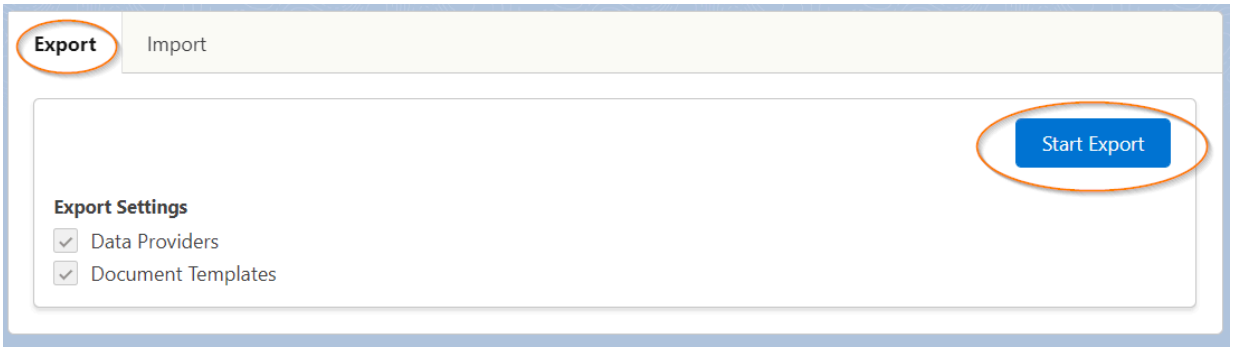
1. Click on the app menu in your org (upper left corner), then click on “View All” at the bottom of the menu.



2. In the App Launcher window that pops up, scroll down until you find the “Import / Export” object, or enter “Import / Export” in the search box. Click on the Link.



3. Make sure you are in the Export tab. (If not, click on Export). In the beta version, both Data Providers and Document Templates are selected. All Data Providers and all Document Templates will be exported. Click on “Start Export”.



4. A file in JSON format will be downloaded. Save the file on your computer where you can find it again. This will be used for the import.

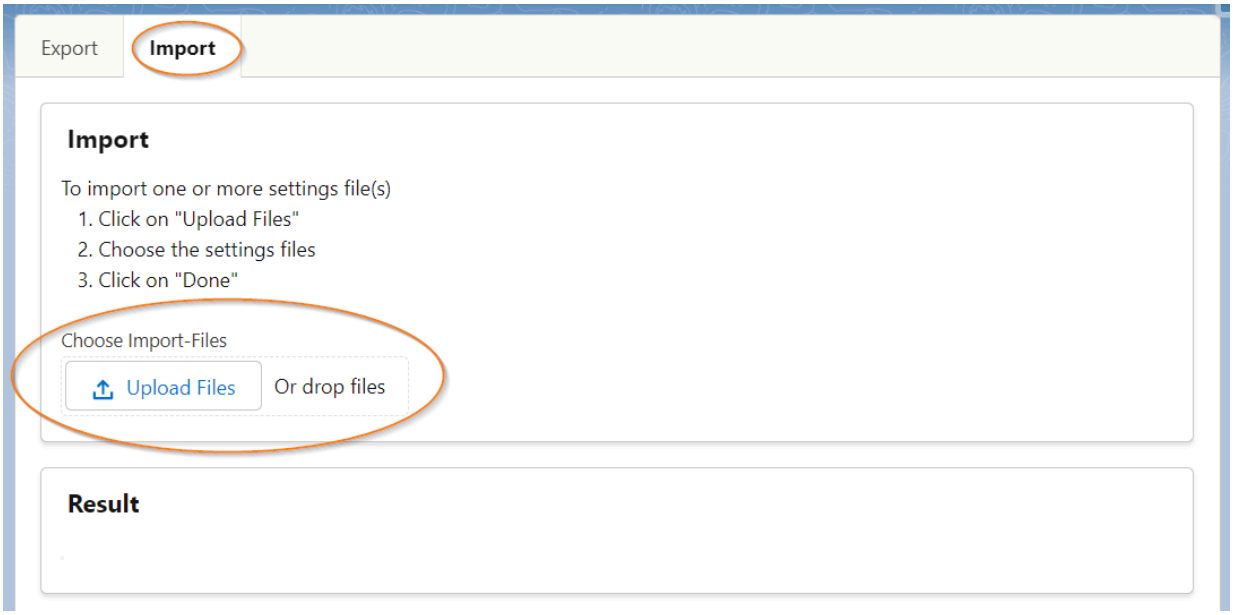
Import

In the org you need to import to:

1. Click on the app menu in your org (upper left corner), then click on “View All” at the bottom of the menu (just as you did for the Export.)

2. Make sure you are in the Import tab. Upload the JSON File you exported and saved on your computer, by either using the Upload File button or drag-and-dropping the file.

   NOTE: Uploading the file *automatically* starts the import!

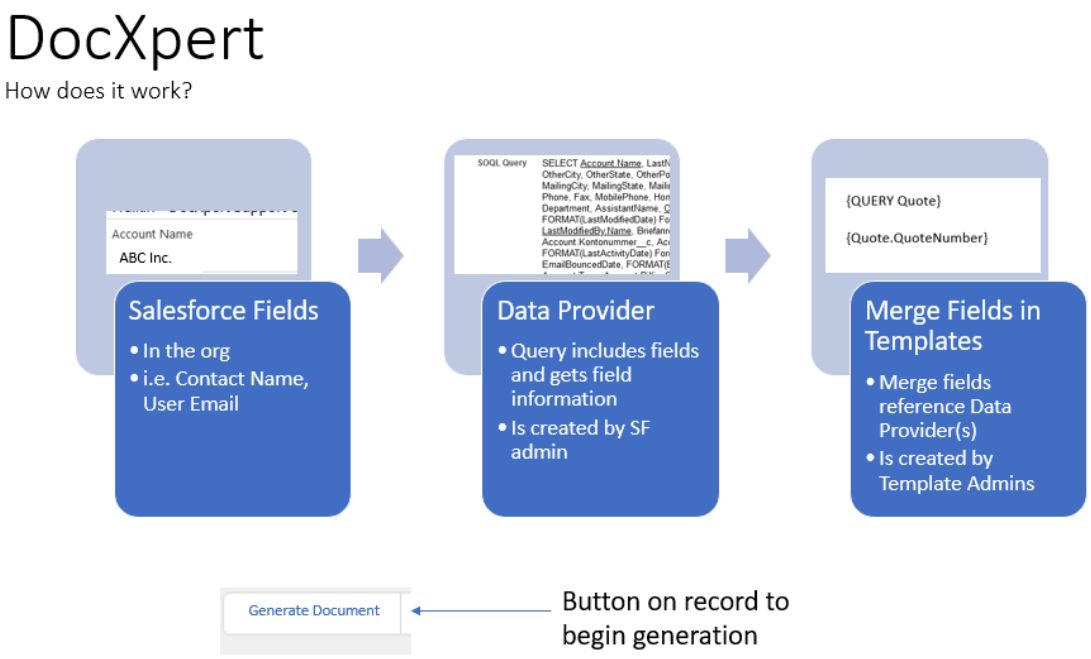


3. As soon as the import beings, the status and results can be seen in Results window.

Template Creator Guide

Before You Begin

DocXpert has various parts that work together in the background to generate a document. Some parts can only be set up by your Salesforce Administrator, but it is important for you as a Template Creator to know the basic parts of DocXpert. You will be creating the Merge Fields in Templates. Here is an short overview:



Merge Templates

Merge templates define the format and content of the document. They include the fixed text, such as standard text blocks, as well as variable “merge fields” for the auto-generated content, such as the customer’s name or a list of products.

Be sure to use .docx format for your merge template. Any other format cannot currently be used is DocXpert. If you accidentally upload a template in a different format, it will not appear under the list of templates to choose from when clicking the “generate document” button.

To keep your generated documents structured, here are some tips to use when creating your merge templates with Microsoft Word:

- Use STRG + Enter to include page breaks in your content.

- You can also follow the instructions in this article to keep words together: Three simple ways to keep Word text together
- Watch out for extra blank spaces! When using copy & paste in a Word document, Word inserts an extra blank space around the copied text without notice. This may cause errors in your merge field syntax or in the structure of your document.
- Be careful when using picklist fields. The picklist values may contain extra spaces or tabs, even when you cannot see this in your Salesforce org. This can especially cause problems when using IF statements with picklist values. For more information, see Inserting conditional content with IF statements.
- **Best practice:** Color code your merge template to show which text will actually be displayed in the generated document and what is simply a query statement or merge field function. For example, make the Query statement at the top of the merge template blue, all IF() and For() statements in the document orange, and all the static text and merge fields that will generate text black. This especially helps if your template becomes more complex and makes troubleshooting easier. (See screenshot below.)

Merge fields need to be written in a specific format and must match exactly to work. Insert merge fields for fields or use functions such as:

- **IMAGE getImageContent(*DataProvider*.ID)** to insert images,
- **IF()** to insert content only if a certain condition is true, or
- **FOR()** to create a loop - meaning a list - of records in your document without having to specify each record.

Here is an example of what a Quote merge template could look like, including the color coding best practice explained above:

{QUERY Quote,QuoteLineItems,HwLogo}

{IMAGE getImageContent(HwLogo.Id)}

{Quote.BillingStreet}
{Quote.BillingPostalCode} {Quote.BillingCity}
{Quote.BillingState}
{Quote.BillingCountry}

Ref: Quote #{Quote.QuoteNumber}

Dear {Quote.Contact.Name},

Thank you for your interest in our products!

Below you will find our quotation for you to review. It includes all products as discussed.
Please review it at your earliest convenience and let us know when we can place your order.

Best regards,
{Quote.Owner.Name}
{Quote.Owner.Phone}
{Quote.Owner.Email}

Product	Description	Product Code	List Price	Quantity	Total Price
{FOR lineItem IN QuoteLineItems}					
{$lineItem.Product2.Name} {IMAGE getImageContent($lineItem.Product2.ProductPicture__c)}	{$lineItem.Product2.Description}	{$lineItem.Product2.ProductCode}	{$lineItem.ListPrice}	{$lineItem.Quantity}	{$lineItem.TotalPrice}
{END-FOR lineItem}					

{IF Quote.BillingCountry === 'USA'}

Additional Information:
{Quote.Additional_Information__c}

{END-IF}

Note: If the merge field cannot be recognized as an API name, an error occurs when trying to generate the document. Merge fields are not case-sensitive when generated, but we recommend writing your merge fields as exact as possible. Spelling errors or blank spaces may lead to errors when trying to generate the document.

Query

To begin a merge template, always enter the query at the top. This tells the generator which Document Data Providers to reference, so it can locate the correct fields or images.
The format is as follows:

{QUERY DataProviderName1,DataProviderName2,DataPoviderName3}

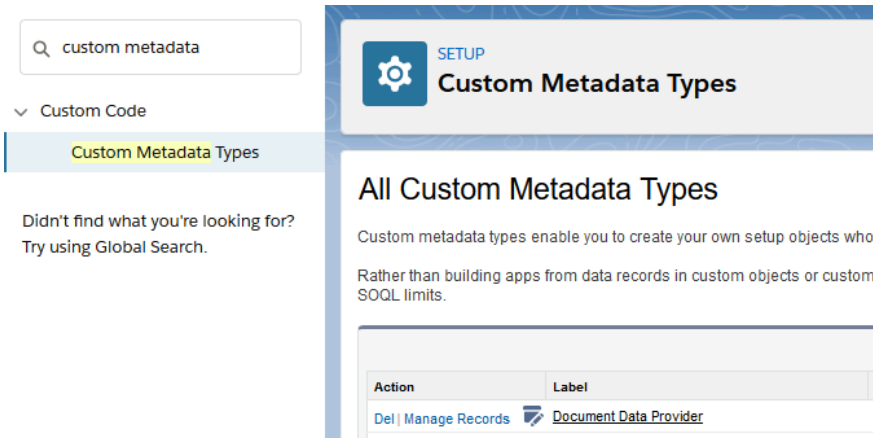
The Query can have as many data providers as necessary. Be sure to separate each data provider name with a “,”.

Quote document example:
{QUERY Quote,QuoteLineItems,HwLogo}

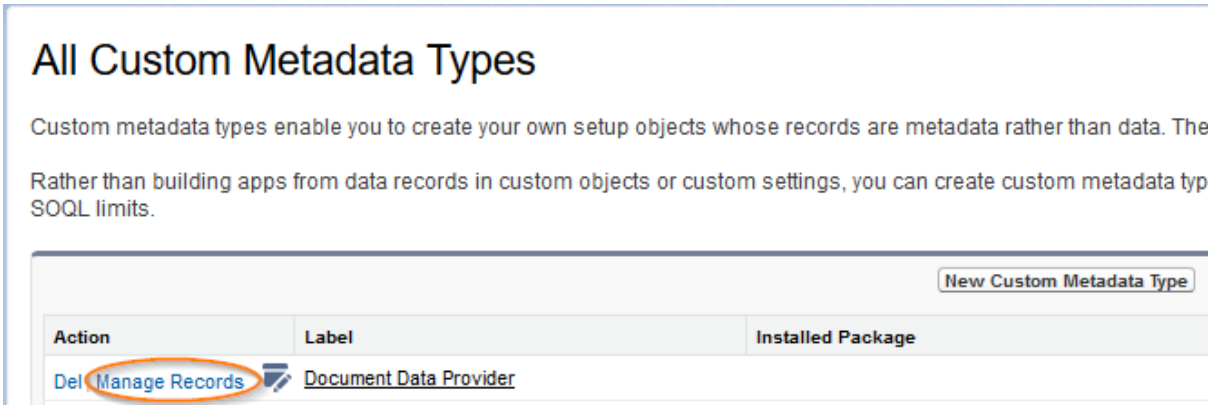
NOTE: The names of the document data providers are case-sensitive! If they are not written exactly, the document generation will fail.

You can find the Document Data Providers for your template. If you do not have system administrator permissions for your Salesforce org, please ask your administrator to look this up for you:

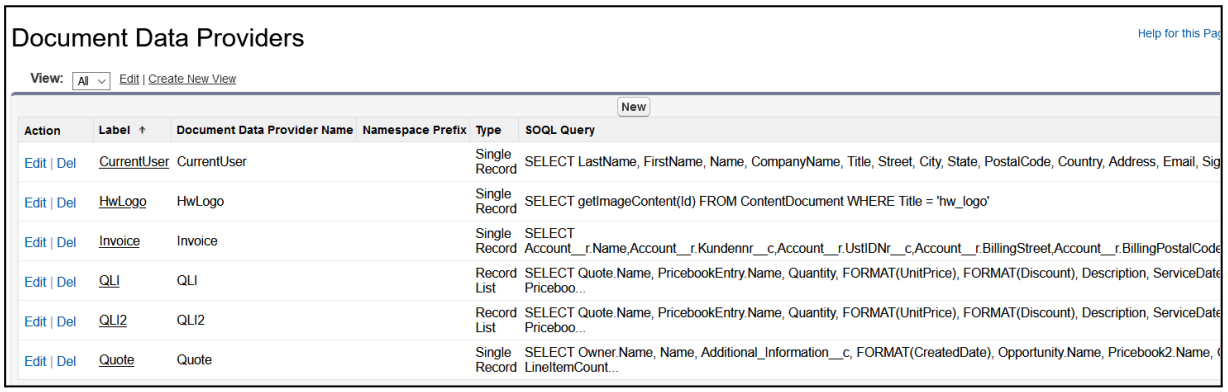
1. Navigate to Setup → click on “Custom Metadata Types”



2. Find the Document Data Provider metadata type in the list.
In the Action column, click on “Manage Records”



Note: if you accidentally click on the “Document Data Provider” link instead, you will come to a different screen. Simply press the “Manage Document Data Providers” button at the top of the page to see a list of your Document Data Providers:



Inserting a field

Merge Fields in your template can have different syntaxes, depending on the type of field and object relationships. Here is an overview, which will be explained in this section:

Field Syntax in Templates (Overview)

Type	Feld Example (API Name)	Feld Relationship*	Syntax in the Template
Standard Field	QuoteNumber	on data provider object	{Quote.Name}
Custom Field	CustomerNumber__c	on data provder object	{Quote.CustomerNumber__c}
Standard Relationship Field	OpportunityName	on related Opportunity object (standard Lookup field)	{Quote.Opportunity.Name}
Custom Relationship Field	Example__c	on related object „XYZ“, related through a custom lookup field	{Quote.XYZ__r.Example__c}

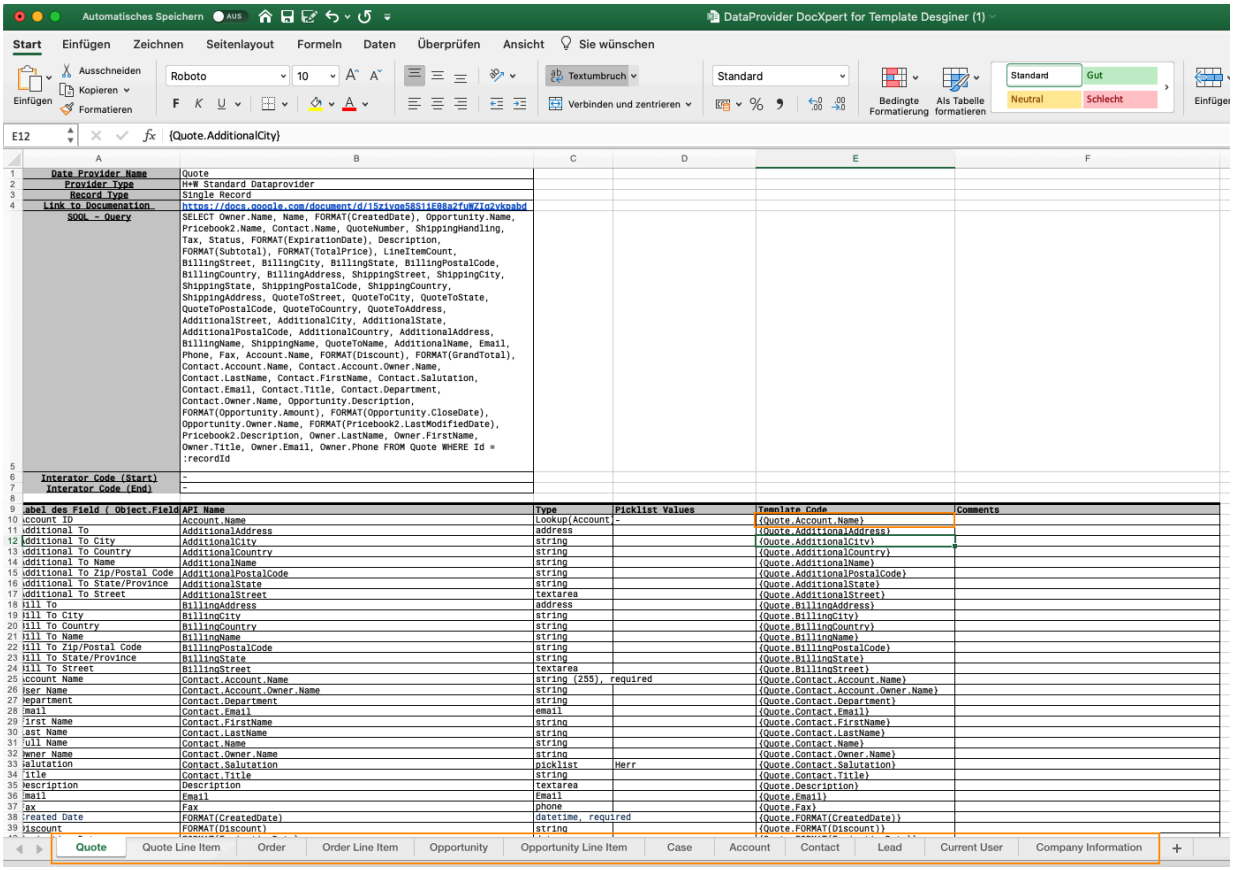
* Examples are based on the use of a data provider for the „Quote“ object

DocXpert Template Designer

To make it easier for you to insert merge fields in the correct syntax and document custom data providers and fields, we provide the DocXpert Template Designer. This is an Excel reference file which gives you an overview of the standard Document Data Providers that come with the DocXpert package, including the fields in each standard data provider and their merge template syntax.

To use the DocXpert Template Designer, simply open the file and copy and paste the merge fields from the “Template Code” column into your merge template.

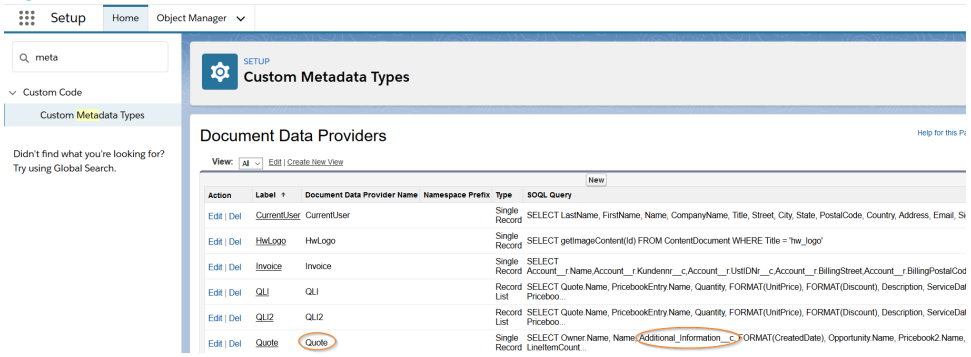
For each standard Document Data Provider, the code for the start and end FOR() function to create loops in your merge templates is also provided. For more information on loops, please reference the Loop section of this document.



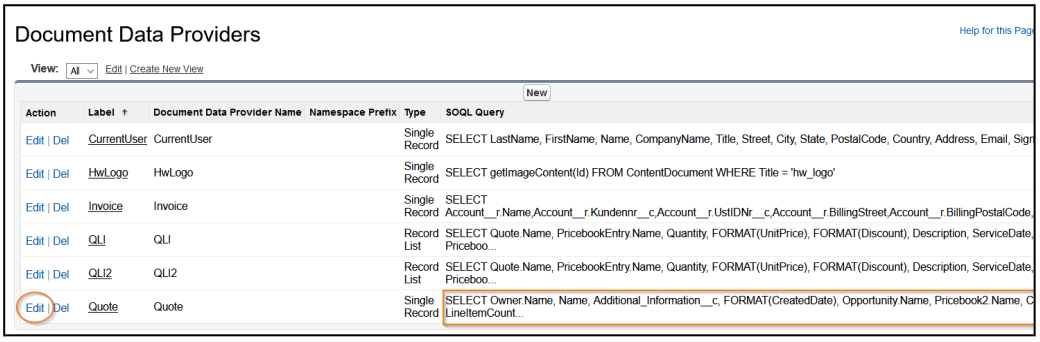
When using *custom* Document Data Providers, *custom* fields, or *custom* objects, you can expand the DocXpert Template Designer to include this custom functionality and serve as documentation. *The exact names of the data providers and field API names are necessary.*

If you do not have system permissions in your org, please ask your Salesforce administrator to add your custom data providers and custom field information to the DocXpert Template Designer for you. You can then determine the correct merge field syntax, document it for future use in the “Template Code” column of the Designer, and use it for your templates.

> Note: You will need to make sure the field name is exactly the same in your data provider and merge templates. Salesforce administrators can ensure this by navigating to the data provider, finding the field in the SOQL Query of the appropriate provider, and copy & pasting it into the DocXpert Template Designer:



If the necessary field is not shown in the query column, click on the “Edit” button to view the full SOQL query.



Be sure to click the “Cancel” button to exit the edit screen afterwards! This ensures you do not accidentally change the SOQL query.

Once you have correct field names from your Document Data Provider, follow the correct syntax to enter them into merge fields in your template. The general merge field syntax is as follows:

{*DataProviderName.Field*}

The example from the above screenshot would be written like this in your template:

{Quote.Additional_Information__c}

Standard Fields

Standard Salesforce fields from the object chosen for the data provider have the simplest syntax. To reference them in a template merge field, only the field name (API name) is necessary.

Example:
When generating a quote document on a Quote record, you may need the Quote Number field or Expiration Date field. Using a data provider based on the Quote object, the merge fields in your template would be written as follows:

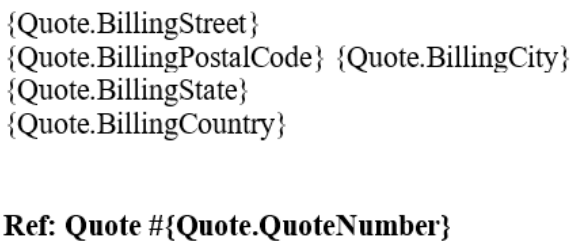
```
{Quote.QuoteNumber}
{Quote.ExpirationDate}
```

Note: As stated above, be sure the name matches the *exact* name written in your Document Data Provider. Blank spaces, tabs or wrong letters could cause document generation to fail.

Always make sure of the following to be sure the merge field will be recognized correctly by the document generator:

- Be sure the Document Data Provider Name matches the data provider exactly.
- Be sure the field in your merge field matches the *exact* name written in your Document Data Provider.
- Be sure to use the { } parentheses exactly as shown here. If one is forgotten or the wrong parentheses are used, you will receive an error message when generating the document.
- Be sure to use a “.” between the Object and the field, otherwise the merge will fail.
- The fields are not case-sensitive. Our recommendation is to write it as exact as possible in your merge fields, but in most cases, this will not cause an error when generating the document.

In the Quote merge template example, standard merge fields look like this:



Custom Fields

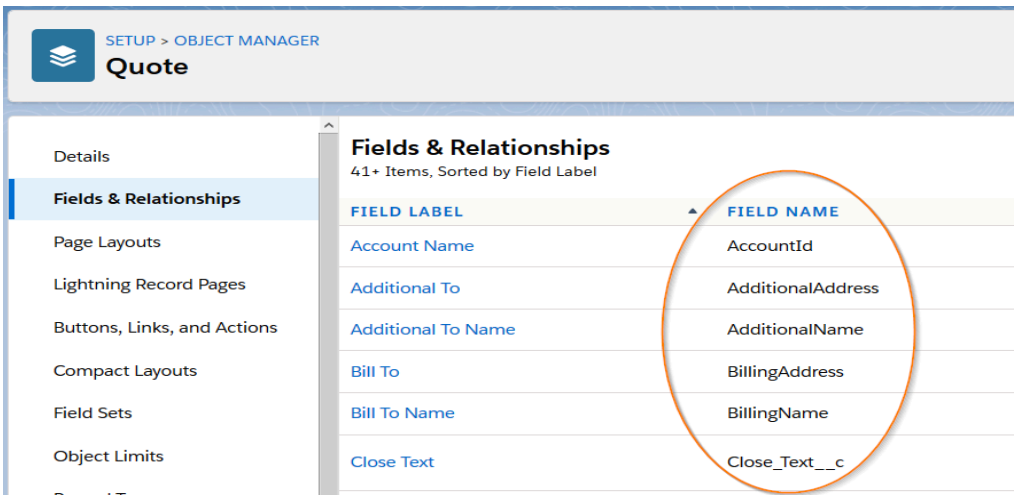
Custom fields include the “ __c” suffix in the field name.

The API Name of a custom field labeled “Customer Number” could therefore be `CustomerNumber__c`. This is exactly how the field should be written in your merge field, including the underscores and appendix “__c”: In this case, the merge field in your template would look as follows:

{Quote.CustomerNumber__c}

Note that there are **two** underscores preceding the c in custom relationship fields. If you only write one underscore “_c” instead of “__c” the query will fail.

NOTE: You may simply copy & paste the field from the relevant Document Data Provider as described under the Inserting a Field section. You may also check the exact name of the field by navigating to the object in Setup and viewing the Field Name:



Standard Relationship Fields

The fields become more complex, when data on related objects is needed. There is a syntax difference between fields on objects related through a standard Salesforce relationship vs. a custom lookup field. If you are unsure which relationship is used, and you are not using the DocXpert Template Designer, ask your Salesforce administrator.

For example, when generating a document on a Quote record, using a data provider based on the Quote object, you may need information from the Account, Contact, Opportunity or

Opportunity Owner (User). For this, you need to scale the object relationships to retrieve the correct data.

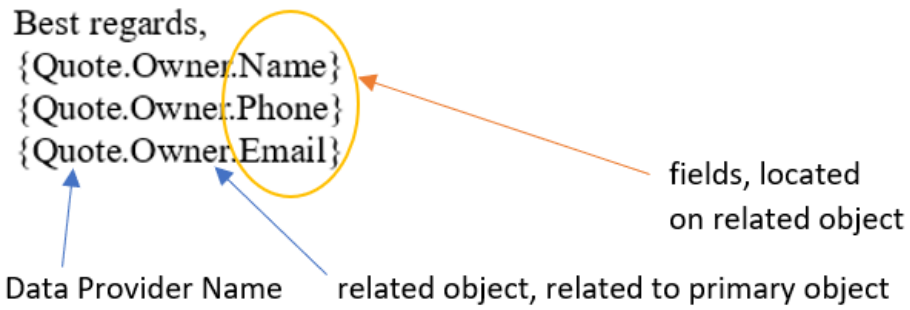
Example: name, phone and email from the Opportunity Owner related to the Quote

To reference the Owner information, which is stored on the related User object, a second level is involved:

Quote record → Owner (User) record

Therefore we need this field format: *RelatedObject.Field*

Then we add the Data Provider in front of this complex field name to create the completed merge field. In the example Quote merge template, the merge fields therefore look like this:



Custom Relationship Fields

When working with custom relationships (custom lookup fields), the related object reference in the merge field must include the "`__r`" suffix.

Example: custom lookup on a standard object to a standard object

You may have a custom lookup field on a standard object, such as Quotes, referencing another standard object, such as User. If this relationship was created with a *custom* lookup field (is not an out-of-the box standard Salesforce field), it is a custom relationship. This type of relationship needs to be written with an "`__r`" syntax in the merge template.

Let's say you would like to get the name of a user in your custom lookup field. This information is not stored in the field itself, but on the User record related through the custom lookup field. If the custom lookup field is named "Product Expert", you would use the API field name for your merge field (for example "`ProductExpert__c`"), but replace the "__c" with "__r" to show the custom relationship as follows:

```
ProductExpert__r.Name
```

If your Document Data Provider Name “Quote”, your merge field would look as follows: `{Quote.ProductExpert__r.Name}`

Note that there are **two** underscores preceding the r in custom relationship fields. If you only write one underscore “_r” instead of “__r” the query will fail.

Note: if you simply wanted to reference the value in the custom lookup field, in the above case it would be the Salesforce Id of the user, you are only referencing the value on the primary object and do not need to scale any relationships. Therefore, you would only need to write the merge field like this: `{Quote.ProductExpert__c}`

Example: custom lookup from custom object to standard object

On a custom object named Invoice, a custom lookup field references the Name field on the standard Account object. The relationship is a custom relationship, therefore the related object has a “`__r`” suffix and would look like this: Account__r.Name

If your Document Data Provider Name is “Invoice”, your merge field would look as follows: `{Invoice.Account__r.Name}`

Generally speaking, “__r” is always necessary when scaling the relationship through a custom lookup field. The “__r” is therefore also necessary for custom lookup fields on standard objects to custom objects, as well as from custom objects to custom objects.

Note: To make it easier, you may simply copy & paste the field given to you by your system administrator in the DocXpert Template Designer, or (if you have system administrator permissions), directly from the relevant Document Data Provider. For more information, please read the section DocXpert Template Designer.

Field Considerations

Date/Currency conversion

Conversion based on user settings:

To convert the format of a date or currency field dynamically based on the user's locale, use the Format() function in the Data Provider. For currency fields, this only changes the format, not the numeric value.

To convert the numeric value of a currency field based on the user's locale, use the convertCurrency() function in the Data-Provider.

Note: If these fields should always be converted to the user's settings when generating their document, it is best practice for the administrator to use these functions in the Document Data Provider. Every currency and date field in the default Data-Providers have 2 versions available.

- Filename = Without user-based formatting
- FormattedFieldname = with user-based formatting

Example: `{Quote.FormattedTotalPrice} or {Quote.TotalPrice}`

Conversion to a specified locale in the template itself:

If your user sends documents to customers in various countries with different formats, the user may want to generate dates or currency fields using the customer's format. This can be done with the following conversion functions on the template:

Note: Please make sure the fields used in the following "convert" functions are not formatted in your Data Provider. Using the Format() function will cause an error.

- For date field values:
  `convertDateToLocale(`***field, locale, timeZone***`)`

- For date/time field values:
  `convertDateTimeToLocale(`***field, locale, timeZone***`)`

- For time field values:
  `convertTimeToLocale(`***field, locale, timeZone***`)`

- For currency values:
  `convertCurrencyToLocale(`***field, locale, currencyCode***`)`

For example, a German user in Germany may view all Salesforce data in German with German formatting, but may have customers in the U.S. Therefore, using the Format() or convertCurrency() functions are not appropriate, because it would use the user's settings to set the values. In this situation, you can use the above "convertCurrencyToLocale" functions to convert these values to the customer's format using the customer locale you enter.

When entering the **locale**, please note the following:

- Use the locale code, for example: en_US
- All locale codes are found here: Support Date and Time Formats
- Make sure you enter the correct quotations around your locale. They must be the straight, single quotations, not curly quotes. The quotations should look like this: 'en_US'

The **timeZone** can be relevant when a date/time value is, for example, 6:00 a.m. on July 31st at your user's locale, but the customer is nine hours behind in a different time zone. This would mean that the value converts to 9:00 p.m. on July 30th for the customer. When entering the locale, please note the following syntax:

- Use the "tz database name" for your time zone, for example: America/New_York
- All names can be found here: List of tz Database Time Zones
- Make sure you enter the correct quotations around your time zone. The must be the straight, single quotations not curly quotes. The quotations should look like this: 'America/New_York'
- If you do not enter a time zone, the function will convert the value's format but use the user's time zone for the value.

  Example date/time fields:
  `convertDateTimeToLocale(`*Order.CreatedDate*`, 'en_US', 'America/New_York')`

When using the **currency** function, the locale should be entered the same as described above using the local code and straight quotations. The currency Code is optional. By entering a currency Code, it does *not* convert the value, but *only changes the currency symbol and format*, for example from Euro (€) to U.S. Dollar ($). Please use multi-currency and set the record to the right currency. Please note the following:

- If you do not enter a currency code, the value's format will be changed based on the locale, but the currency symbol will not be changed
- To enter a currency Code, be sure to use a currency code listed here: Supported Currencies

- Make sure you enter the correct quotations around your locale. They must be the straight, single quotations not curly quotes. The quotations should look like this: 'USD'

  Example currency field:
  `convertCurrencyToLocale(`*$LineItem.ListPrice*`,` 'en_US'`,`'USD'`)`

Tip: To make currencies dynamic, you can use the “Currency ISO Code” field on your records and enter the currency code. Then reference that field in your template instead of entering the code itself. For example:

{convertCurrencyToLocale($lineItem.ListPrice, 'en_US', $lineItem.CurrencyIsoCode)}

Note: These functions for specifying a locale will not work, if the Document Data Provider includes the Format(), convertCurrency(), or any other functions on the field you are trying to convert. Therefore, please check the field in the SOQL query of the Document Data Provider or check with the administrator responsible for the Document Data Providers. The data provider may need to be adjusted, or a new data provider specifically for templates with locale specifications may be needed.

Date/Time format function

To statically set the format of a date/time field, use the formatDateTime() function in your template. The syntax is as follows:

`formatDateTime(`***field***`, `***'format'***`)`

You can create your own format by arranging any of these placeholders to create the format of your choice:

- yy = 2-digit-year
- yyyy = full year
- M = digit month
- MM = 2-digit month
- MMM = short month name
- MMMM = full month name
- EEEE = full weekday name
- EEE = short weekday name
- d = digit day
- dd = 2-digit day
- h = hours (12 hour time, am/pm)
- hh = (24 hour time, 2-digit am/pm)
- H = hours (24 hour time)
- HH = 2-digit hours (24 hour time)
- m = minutes
- mm = 2-digit minutes
- a = AM/PM
- s = seconds
- ss = 2-digit seconds
- S = miliseconds

Example: `{formatDateTime(field__c, 'yyyy_MM_dd')}`

If you would like to use the current date, you can use the now() function.

Example: `{formatDateTime(now(),('yyyy_MM_dd'))}`

Summarizing values in a table

You can add values together directly in a document's table by using the SUM() function in your template. The syntax for the SUM() function is as follows:

`sum(`***Data Provider***, '***field***'`)`

Note: This function is NOT compatible with the FORMAT() function in your data providers. Therefore, before using this function in your template, please make sure the FORMAT() function is not used for the fields you wish to add together.

Because the SUM() function only generates a non-formatted number (i.e. 2500 instead of $2,500.00) You will need to format those fields in the template using a further function.

Example 1: The generated sum should always be shown in a static format.

For this scenario, use the convertCurrencyToLocale() function. The syntax is:

{convertCurrencyToLocale(sum(***DataProvider***, '***Field***'), '***Locale***', '***CurrencyISOCode***')}

For example, if your company only works with German-formatted currency, generated in Euros, the function would look like this:

{convertCurrencyToLocale(sum(QuoteLineItems, 'TotalPrice'), 'de_DE', 'EUR')}

Note: The fields being added together also need to be formatted accordingly. Please note the syntax for using the covertCurrencyToLocale function for fields in a loop. Instead of using the Data Provider, the loop identifier must be used. In this example the identifier is $lineItem. and would look as follows:

{convertCurrencyToLocale ($lineItem.TotalPrice, 'de_DE', 'EUR')}

In the example table below, the sum() function is used to calculate the total List Price and Total Price, using the convertCurrencyToLocale() function and static values for the formatting:

Product	Description	Product Code	List Price	Quantity	Total Price
{FOR lineItem IN QuoteLineItems}					
{$lineItem.Product2.Name}	{$lineItem.Product2.Description}	{$lineItem.Product2.ProductCode}	{convertCurrencyToLocale($lineItem.ListPrice, 'de_DE', 'EUR')}	{$lineItem.Quantity}	{convertCurrencyToLocale($lineItem.TotalPrice, 'de_DE', 'EUR')}
{END-FOR lineItem}					
TOTAL:			{convertCurrencyToLocale(sum(QuoteLineItems, 'ListPrice'), 'de_DE', 'EUR')}		{convertCurrencyToLocale(sum(QuoteLineItems, 'TotalPrice'), 'de_DE', 'EUR')}

Example 2: The generated sum should be based on the user's format and default currency.

For this scenario, use the convertCurrencyToLocale() together with the getUserLocale() and getDefaultCurrency function. You will have four functions, including the sum() function. The syntax is as follows:

{convertCurrencyToLocale(sum(***DataProvider***, '***Field***'), getUserLocale(), getDefaultCurrency())}

For example, if you are calculating the total price, the functions could look like this in your template:

{convertCurrencyToLocale(sum(QuoteLineItems, 'TotalPrice'), getUserLocale(), getDefaultCurrency()}

Note: The fields being added together also need to be formatted accordingly. Please note the syntax for using the covertCurrencyToLocale function for fields in a loop. Instead of using the Data Provider, the loop identifier must be used. In this example the identifier is $lineItem and would look as follows:

{convertCurrencyToLocale ($lineItem.TotalPrice, 'de_DE', 'EUR')}

In the example table below, the sum() function is used to calculate the total List Price and Total Price, using the convertCurrencyToLocale() function, with the user's locale and default currency for formatting:

Product	Description	Product Code	List Price	Quantity	Total Price
{FOR lineItem IN QuoteLineItems}					
{$lineItem.Product2.Name}	{$lineItem.Product2.Description}	{$lineItem.Product2.ProductCode}	{convertCurrencyToLocale($lineItem.ListPrice, getUserLocale(), getDefaultCurrency())}	{$lineItem.Quantity}	{convertCurrencyToLocale($lineItem.TotalPrice, getUserLocale(),getDefaultCurrency())}
{END-FOR lineItem}					
TOTAL:			{convertCurrencyToLocale(sum(QuoteLineItems, 'ListPrice'), getUserLocale(),getDefaultCurrency())}		{convertCurrencyToLocale(sum(QuoteLineItems, 'TotalPrice'), getUserLocale(),getDefaultCurrency())}

Referencing blank values

- In simple merge fields, if the field referenced has a blank value, the merge field is simply ignored when generating the document. A blank space will be entered into the document where the field's value would have been, but no error will occur.
- In complex merge fields which scale object relationships, behavior depends on which field is blank:
    - Example1- does not create an error and inserts a blank space:
      `Quote.BlankField1__r.BlankField2__r.BlankField3__c`
    - Example 2- creates an error, because the first field has a value:
      `Quote.TextField1__c.BlankField2__r.BlankField3__c`

Text fields

We recommend writing any static text directly in your template.
If you decide to insert text fields, make sure they are *not* rich text fields.

Note: Rich Text fields are ***not*** supported by DocXpert! Using rich text fields would insert the HTML markup along with the text itself and would look similar to this:

Additional Information:
When ordering, please be sure to include your Customer ID and Quote number. Feel free to call your representative, otherwise you can gladly order in our online shop.

General Template Functions

We have also integrated some functions which are not field specific.

Please note that **all functions are case sensitive**!

Template function	Example usage
Generate the current day	{today()} **complex example:** {convertDateToLocale(today(),'en_US','America/New_York')}
Repeat a string	{repeatString(".", 5)}
Length of a string	{length(Quote.BillingCity)}
UpperCase a string	{toUpperCase(quote.BillingCity)}
LowerCase a string	{toLowerCase(quote.BillingCity)}

Note: If you are missing functionality you can always use the salesforce formula field in combination with a custom data-provider to handle more complex use-cases.
If you need a specific function for your template contact our support.

IMAGE() Inserting an image

To insert an image using DocXpert, use the IMAGE function. Please be aware that all images must be saved under the Files tab. For more information visit the Before you Begin: Images section in this user's guide.

Note: Alternatively, you can put a static image *directly* in the header or footer of your Word template without needing a DocXpert data provider or query. Such images *directly in your Word template* do not need to be uploaded in Salesforce as Files or queried in your template.

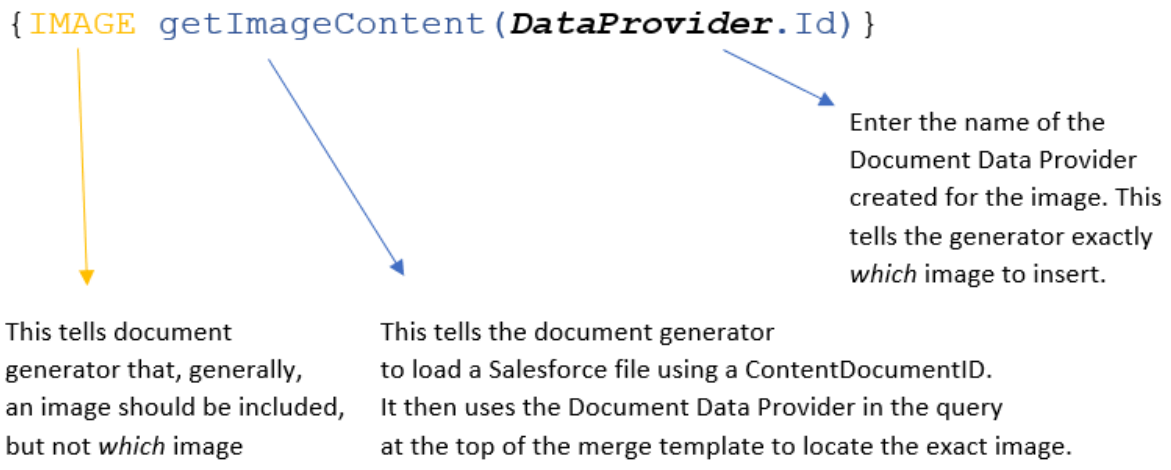
Static image (ex: Logo)

Before including the image function in your merge document, be sure of the following:

- The image is uploaded and saved to your "Files" tab.
- A Document Data Provider for that image has been created. See the "Document Data Provider" section in the Administration Guide for how to create a Document Data Provider. This needs to be done by a system administrator.
- The name of the Document Data Provider is included in your query at the top of your merge document. Otherwise the image function will fail.

Note: Please note that all functions are case sensitive!

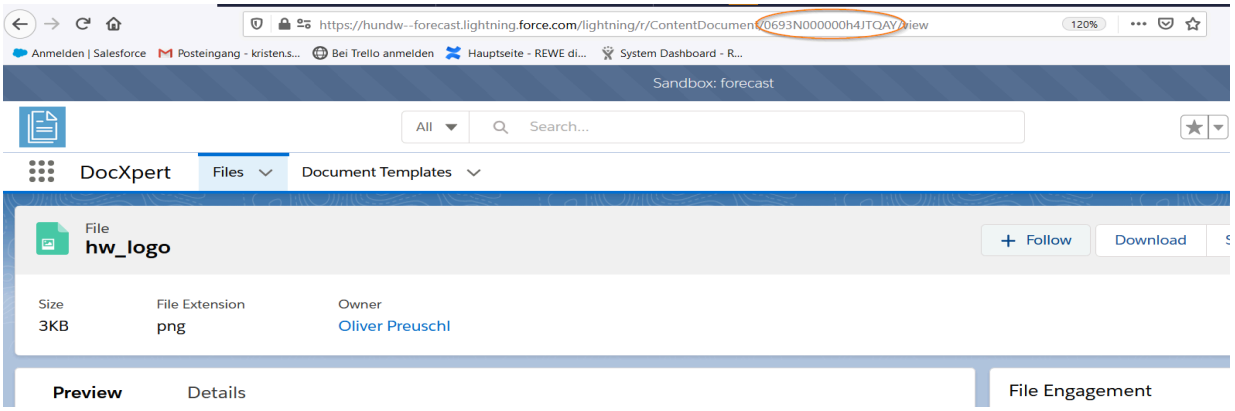
The syntax for the image function is as follows:
{IMAGE getImageContent(*DataProvider*.Id)}



Note: The ContentDocument ID is required for all images. No other ID is accepted by the template engine. Therefore, please be sure to use the ContentDocument ID, not the ContentVersion ID.

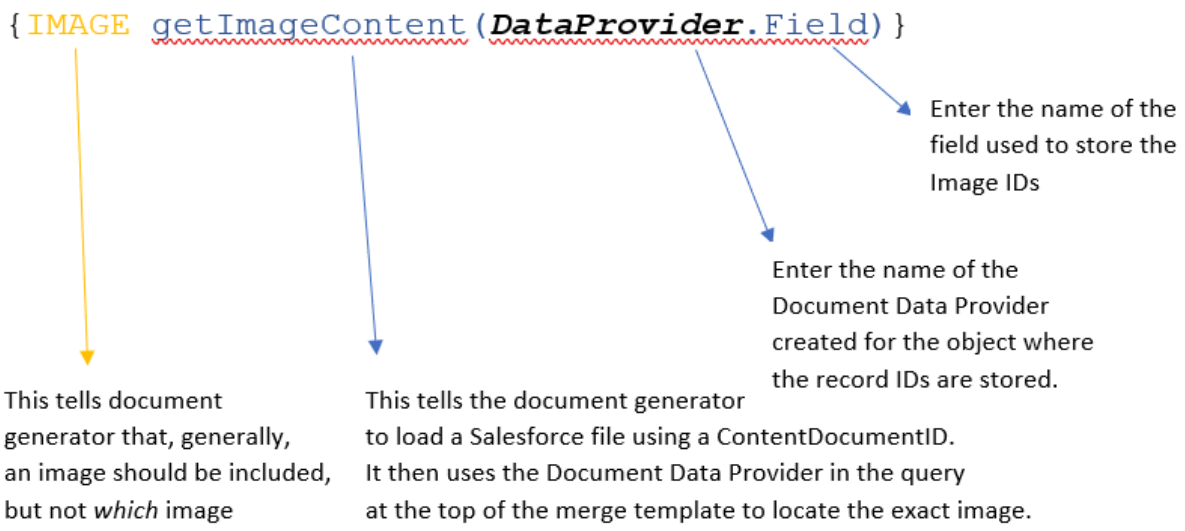
Images on a record

To store an image on your record, such as a product image on your product record, write the ContentDocumentID in a custom text field. This is a workaround, since a lookup to an image is not currently possible in Salesforce. To find the SalesforeID of the image, navigate to your image in Salesforce, click on “View File Details”, and find the ID in the URL, as shown here:



To configure this workaround:

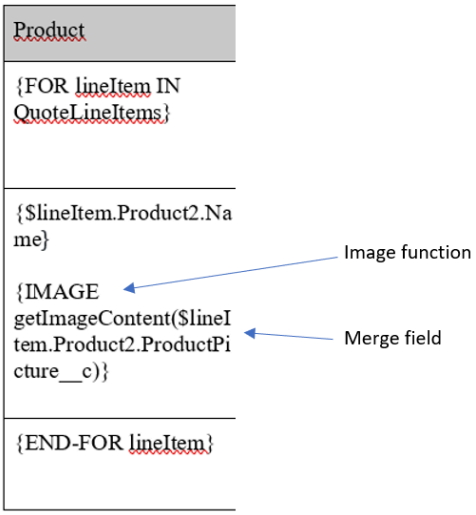
1. Create a custom field (a field “ProductPicture__c” on your Product object, for example). Be sure to make this field editable and put on the appropriate page layout for the person responsible for adding the images to the records.
2. Upload the image to your Files as described in the “Before You Begin: Images” section of the Administration Guide. You may also use the Files related list on your Product records, if the related list is on your object layout.
3. After uploading the image, copy the Salesforce ID from the URL, navigate to your Product record, and enter it in the field.
4. Save the record.
5. Repeat steps 2-4 for each record which should have an image.
6. Now you can reference the image in a merge field using this syntax:



For this example, it is:
{IMAGE getImageContent($lineItem.Product2.ProductPicture__c)}

Note: Although the Data Provider name is normally used in theIMAGE() function, "$lineItem" is used here instead. This is only because it is included in a loop and must reference the loop name "lineItem" in this case. If not included in a loop, the Data Provider image would be entered, as explained above.

In this advanced example, which shows the Product picture for a Quote Line Items loop, the whole statement (including the loop) would look like this in the template:



To learn more about loops in your merge template, read the following section on the FOR() function.

ContentDocument Images

It is possible to dynamically insert ContentDocument images attached to a record as “Files”. For example, at the end of your generated document, you may want to list all .jpeg Files attached to the record.

To do this, you need to:

- Make sure a Data Provider and default size for the images has been set up by your administrator. For more information, please see the ContentDocument Images section of the Administration Guide.
- Use the Loop function to list all ContentDocuments one after the other in your document. For more information on the Loop function, please see the FOR() Creating a loop section of this document.
- Use the image function to generate the image. Please note the syntax below!
- Be sure to use the appropriate Document Data Provider name. If you are unsure, ask your administrator.

The syntax should be as follows:

```
{FOR ContentDocumentLink IN Document Data Provider Name}
{IMAGE getImageContent($ContentDocumentLink.ContentDocumentId)}
{END-FOR ContentDocumentLink}
```

Note: Please use the Syntax exactly as written above! You only need to insert the appropriate Document Data Provider Name from your org where specified. This will generate all images one after the other in your document.

FOR() Creating a loop

Loops can be used to create a dynamic list of records without having to specify each specific record in your merge template. For example, you could use a loop to include the Quote Line Items on your generated quote document. Loops are written using the FOR() function.

Always use the following syntax to be sure the FOR() statement functions properly:

- Begin your FOR statement with this syntax:
  {FOR ***Record*** IN ***Document Data Provider Name***}

  Note: the ***Record*** name can be anything you choose. Best practice is to name it so that it logically explains what is being referenced. The name chosen needs to remain consistent throughout the entire loop, otherwise the loop will fail. See the Quote example below.

- Then enter the content you which to auto-generate using the appropriate function(s).
- Conclude your FOR statement with this syntax: {END-FOR ***Record*** }
- Be sure to use the { } parentheses exactly as shown in the example below. If any are forgotten or the wrong type of parentheses are used, such as () or [ ], the function will fail.
- Always include the $ symbol, as shown in the example below. It must be inserted in front of each merge field included for a record in the loop, such as in this example for the List Price field in a Quote Line Item loop: {$lineItem.ListPrice)
- If quotations are necessary, always use the ' ' quotations. These are singular and straight, not slanted `´, curved ‘’, or double “ ”. Using the wrong quotation marks will cause the function to fail. Word often creates problems if typing these quotations in by hand. Best practice is to copy & paste the quotations out of the Document Data Provider (where they are correct) or an SOQL query tool.

Note: Please note that **all functions are case sensitive**!

As a reminder, this is what should be generated in the Quote document example:

Product	Description	Product Code	List Price	Quantity	Total Price
Product Ex1 *Product Image, if there is an associated image*	*Description 1 Text*	*XX00001*	*ListPrice €*	*x*	*TotalPrice €*
Product Ex2 *Product Image, if there is an associated image*	*Description 2 Text*	*XX00002*	*ListPrice €*	*x*	*TotalPrice €*
Product Ex3 *Product Image, if there is an associated image*	*Description 3 Text*	*XX00003*	*ListPrice €*	*x*	*TotalPrice €*

This is how you would write the loop and fields on the merge template to achieve this:

Begins FOR statement for Line Items records on Quote object

Includes all content that should be auto-generated

Closes FOR statement for Line Item records

Product	Desciption	Product Code	List Price	Quantity	Total Price
{FOR lineItem IN QuoteLineItems}					
{$lineItem.Product2.Name} {IMAGE getImageContent($lineItem.Product2.ProductPicture__c)}	{$lineItem.Product2.Description}	{$lineItem.Product2.ProductCode}	{$lineItem.ListPrice}	{$lineItem.Quantity}	{$lineItem.TotalPrice}
{END-FOR lineItem}					

This tells the generator the following: Find the “lineItem” records referenced in the DocumentDataProvider “QuoteLineItems” and insert the data in the Product2.Name, Product2.Description, Product2.ProductCode, ListPrice, Quantity, and TotalPrice fields for each record found, listing the data for each record in a separate row in the table.

Note: The records in a loop will be listed in a random order unless otherwise specified. You can order the products in the SOQL data provider query using `ORDER BY`. More information can be found in this Salesforce Developer article.

The “lineItem” represents the individual records. This name can be anything you choose, but must remain consistent throughout the loop. As shown in the example, it must be used:

- In the beginning of the FOR statement: {FOR lineItem... }

- At the beginning of each merge field in the loop, with the $ before it: {$lineItem.Product2.Name} or {$lineItem.TotalPrice}
- At the end of the FOR Statement: {END-FOR lineItem}

Note: The Product Picture field is not a standard field. To follow this exact template, create a new custom text field on your Product object, upload a corresponding picture, and copy the SFID of the uploaded image to the Product Picture field.

Index Function

If you would like a numbering of the elements of an array, in a document, you can use the index function.
For example, in a quote, you want to list the individual quote line items and number them.

The syntax for this is as shown in the example below. The Function would look like this:

```
{getIndex($lineItem)+1}
```

Note: Make sure you have a For() function in the table to define the elements you want to list and number.

Example:

Index#	Product	Image	Product Code	List Price	Quantity	Total Price
	{FOR lineItem IN Opportuni-tyLineItems}					
{getIn-dex($lineI-tem)+1}	{$lineI-tem.Name}		{$lineItem.Pro-duct2.Product-Code}	{$lineI-tem.List-Price}	{$lineI-tem.Quantity}	{$lineItem.To-talPrice}
	{END-FOR lineItem}					

The example given would look like this in a finished document:

Index#	Product	Image	Product Code	List Price	Quantity	Total Price
1	DocXpert GenWatt Die-sel 10kW			5000	1	5000
2	DocXpert GenWatt Die-sel 1000kW			100000	2	200000
3	DocXpert SLA: Bronze			10000	20	200000
4	DocXpert SLA: Gold			30000	30	900000

Length Function

This Function returns the length or size of a collection (like an array or list). So if you have a collection and you would like to show the aggregation of elements of this collection in one of your documents you can use this function.

{getLength(**DataProvider**)}

As an example here is the use of this function with Quote Line Items of the quote object:

{getLength(QuoteLineItems)}

In this example are 3 Quote Line Items on this specific quote. An the function is used to aggregate the sum and show the result:

Example:

Amount of Quote Line Items: {getLength(QuoteLineItems)}

Index#	Product	Image	Product Code	List Price	Quantity	Total Price
	{FOR lineItem IN Quote-LineItems}					
{getIn-dex($lineI-tem)+1}	{$lineI-tem.Pro-duct2.Name} IsFirst: {is-First($lineI-tem)} IsLast: {is-Last($lineI-tem)}		{$lineItem.Pro-duct2.Product-Code}	{$lineI-tem.List-Price}	{$lineI-tem.Quantity}	{$lineItem.To-talPrice}
	{END-FOR lineItem}					

Amount of Quote Line Items: 3

Index#	Product	Image	Product Code	List Price	Quantity	Total Price
1	GenWatt Die-sel 10kW		GC1020	5000	3	15000
2	SLA: Bronze		SL9020	10000	1	10000
3	Installation: Industrial - High		IN7080	85000	3	255000

IF() Inserting conditional content

The IF() function is used to enter conditional information. Examples uses for the IF() function include:

- Ex 1: Using an IF statement to enter text specific to only one country or region
- Ex 2: Using the IF statement to check if a merge field value is blank, and - when blank - insert a different field instead.
- Ex 3: Using the IF statement to check if a checkbox is activated

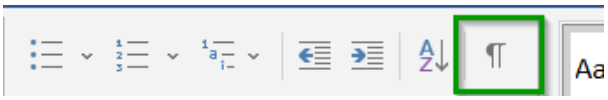
Note: You do not need to use an IF statement to check if a field is blank to avoid an error. If a field is blank, the field is left out and document is still generated.

Always use the following syntax to be sure the statement functions properly:

- Begin your IF statement with this syntax: {IF ***Condition to be checked*** }
- Then enter the content you which to auto-generate using the appropriate function(s).
- Conclude your IF statement with this syntax: {END-IF}
- Be sure to use the { } parentheses exactly as shown here. If any are forgotten or the wrong parentheses are used, the function will fail.
- If quotations are necessary, always write them using the ' ' quotations. These are singular and straight, not slanted ` ´ , curved ‘’, or double “ ”. Using the wrong quotation marks will cause the function to fail. Word often creates problems if typing these quotations in by hand. Best practice is to copy & paste the quotations out of the Document Data Provider (where they are correct) or an SOQL query tool.
- **To avoid unwanted spaces/lines when using complex IF statements:**
  the examples below are written user-friendly on several lines to explain how the IF statement works. To avoid empty spaces or empty lines in your generated document, do not use spaces or carriage returns when combining IF statements.
    - Example of syntax to eliminate extra spaces:
      {IF Quote.Billing.Country ===‘USA’}Additional Information: {Quote.AdditionalInformation}{END IF}{IF Quote.Billing.Country ===’UK’}Additional Information: {Quote.AdditionalInformationUK}{END IF}

    - In the above example, there are no spaces or returns between the two IF statements. They are written as one block. Using a space or return in that position would cause blanks in the generated document, even if an IF statement criteria is not met.

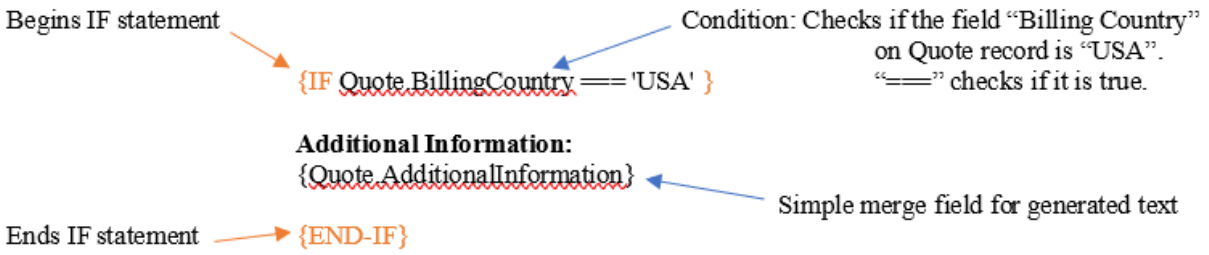
- Tip: Use the Word function in the Start menu to show all spaces/returns in the document. The symbols themselves will not be generated in the document:



Note: Please note that **all functions are case sensitive**!

Example 1: simple IF() statement
Using an IF statement to enter text specific to only one country or region

This example is shown on the example Quote template in this handbook:
“If the customer is located in a specific country, enter the additional text for that country”



Note: The Additional Information field is not a standard field. To follow this exact template, create a new custom text field on your Quote object.

In the above example, we are checking if something “is equal to” using the === operator. Here is the list of all possible operators to use in IF statements:

Operator	Meaning
<=	less than or equal
>=	greater than or equal
<	less than
>	greater than
==	equal
!=	not equal
===	strict equal
!==	strict not equal

Note: These are based on JavaScript operators. For more information, please reference this developer guide.

Example 2: IF () statement with isBlank () function
Using the IF statement to insert certain field when a field has a value, but enter alternative fields if a merge field is blank.

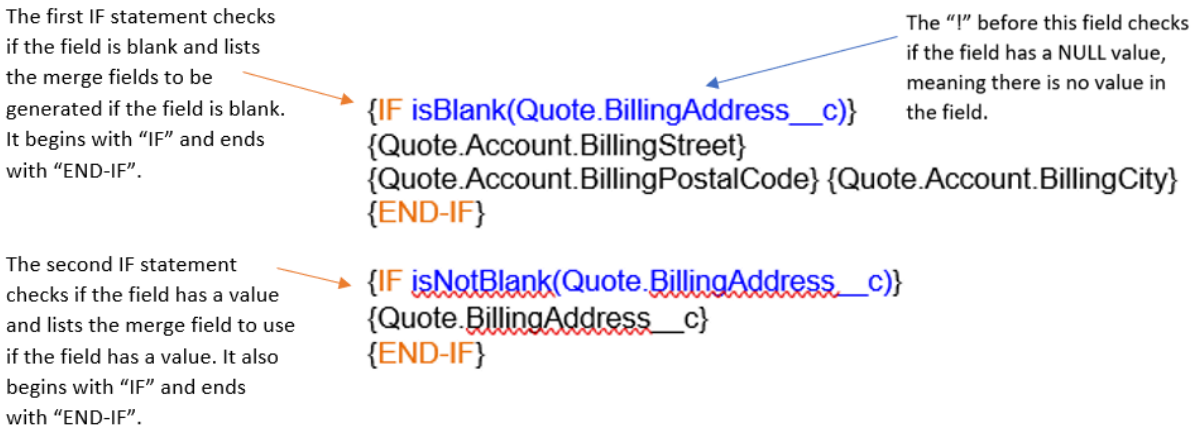
There are two functions you can use for this:

isBlank() checks if a field is blank.The syntax for your merge field is as follows:
`isBlank(`***Field to Check***`)`

isNotBlank() checks if a field has a value. The syntax for your merge field is as follows:
`isNotBlank(`***Field to Check***`)`

For example, use this syntax with two IF statements to enter a specific field value is a field is blank, and a different field if it is not blank:

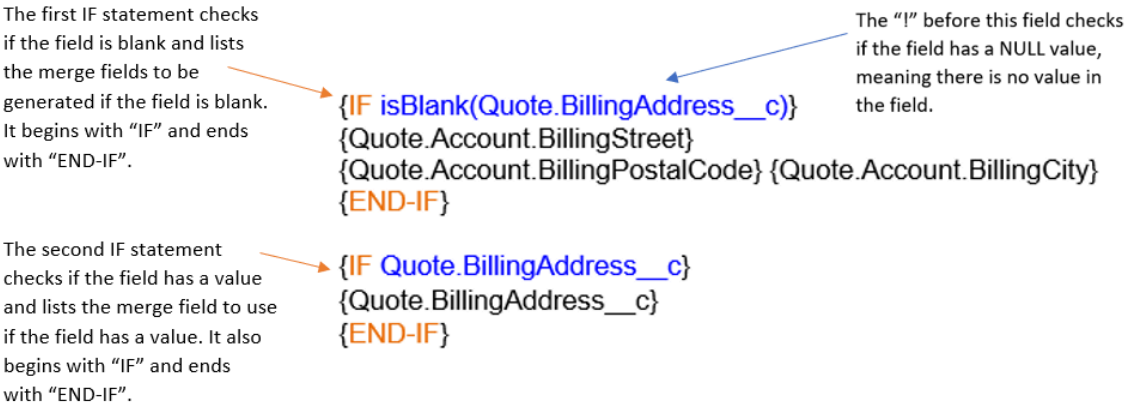
```
{IF isBlank(FieldtoCheck)}
{AlternativeFieldtoInsertIfBlank}
{END-IF}
{IF isNotBlank(FieldtoCheck)}
{FieldtoInsertIfValueExists}
{END-IF}
```



Note: The function "isNotBlank()" is not necessary to check if a field has a value. When left out in an IF statement, the statement automatically checks if the field "is not blank". Therefore, an alternative way of writing the above IF statement is:

```
{IF isBlank(FieldtoCheck)}
{AlternativeFieldtoInsertIfBlank}
{END-IF}
```

```
{IF(FieldtoCheck)}
{FieldtoInsertIfValueExists}
{END-IF}
```



Example 3: IF () statement with checkbox
Using the IF statement to check if a checkbox is activated

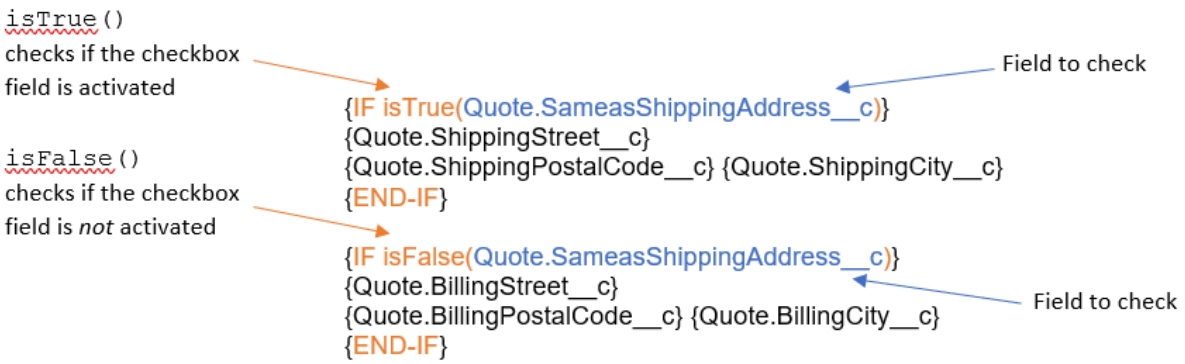
You can use an IF statement to check if a checkbox is active and enter a certain field when it is, and a certain field when it is not. This is similar to example 2 above.

To check if a checkbox is activated (true), use this syntax:
`IF isTrue(`***Field to Check***`)`

To check if a checkbox is not activated (false), use this syntax:
`IF isFalse(`***Field to Check***`)`

For example, you could have a custom checkbox on your Quote regarding the customer's Billing Address called "Same as Shipping Address". If the checkbox is activated, the billing address and shipping address are the same. If not activated, they are different. Based on whether the checkbox is activated or not, the correct field could be entered. To do this, use the following syntax with two IF statements:

```
{IF isTrue(FieldtoCheck)}
{Field to Insert if True}
{END-IF}
{IF isFalse(FieldtoCheck)}
{Field to Insert if False}
{END-IF}
```



Using Picklist values with the IF () function

When using the IF function to evaluate a value in a picklist, be aware that there may be extra spaces or tabs in your value. These are not able to be seen in the UI or in Setup of Salesforce, but can cause problems when trying to reference an exact value.

Best practice: To make sure the field looks exactly as expected, consider incorporating a debug line at the end of your template while testing. Once the template works as it should, this line can then be deleted before deployment.

For example, if you are checking if the value “very interested” is chosen in your “Level of Interest” picklist field, write a debug line which gives you the value given in a record, and test this on a record with this value chosen. The text generated will show you if there is an extra space or tab in the value. In this case, you may find that the value - when written exactly - is “`very interested `”, with an extra space at the end.

Conditional tables using IF()

You can use the IF() function to conditionally generate tables in your documents only when records are found by the data provider.

For example, you may have various kinds of Quote Line Items in your organization, categorized by the field “Product Family”. These could include a Product Family called “fees”. In your document, you may wish to have these “fee” Quote Line Items in a separate table, and only show them if there are any such records on the Quote. (Some Quotes may not have “fees”.) For this, you need to make sure your data provider is set up accordingly, and then use the IF() function to evaluate if any records have been found.

The syntax in your document would be as follows:

{IF ***DataProvider***.length>0}
insert Word table here
{END-IF}

For further information, see the Troubleshooting Guide.

toFixed() Fixing decimal places

When generating decimal places with DocXpert, you may have more decimal places in the field in Salesforce as you would like to have in your generated document.

For example, you have a number field in Salesforce with two decimal places (100.00) for calculation purposes, but do not wish to have any decimal places in your document (100).

You can define the number of decimal places using the toFixed() function in directly in your DocXpert template. The syntax is as follows:

toFixed(FieldName, #)

For example, for 0 decimal places on a custom field called Total Value:

toFixed(TotalValue__c, 0)

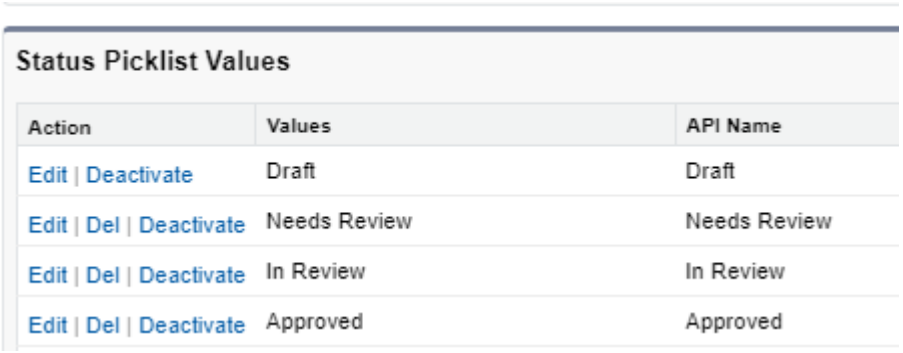
NOTE: This function automatically rounds the number.
5 and higher gets rounded up. Below 5 gets rounded down.

Picklist Value Labels

When using picklist values in DocXpert templates, the value's "API name" is generated as default, not the value's label ("Value"). In many cases, these two values are the same, but it depends on your configuration. You can check this if you notice unexpected generated values in your template. Simply have your Salesforce administrator check your picklist field:

Setup → Object Manager → Object (ex: Quote) → Field (ex: Status) → Status Picklist Values

In the following screenshot, the Values for the "Status" field are the same as the API Name:



Status Picklist Values

Action	Values	API Name
Edit | Deactivate	Draft	Draft
Edit | Del | Deactivate	Needs Review	Needs Review
Edit | Del | Deactivate	In Review	In Review
Edit | Del | Deactivate	Approved	Approved

Should the API Name be different and not what you would like to generate, you can use the TOLABEL function to generate the Values shown instead. This requires customizations in your data provider as well as in the template.

In the Data Provider use the TOLABEL function: TOLABEL (Field Name) FieldNameLabel
This needs to be done by a Salesforce Administrator.

Example: TOLABEL(Status__c) StatusLabel

In the template, use the FieldNameLabel you specified in the data provider:
{DataProvider.FieldNameLabel}

Example: {Quote.StatusLabel}

Document Template Object

Merge templates are stored as records on the Document Template object. Therefore, to upload and use your new template, create a Document Template record by following the steps in the How to Create and Upload Your Document Template section of this handbook.

The steps will lead you to fill out the following fields on the Document Template record:

Document Template Name
This can be any name you choose. This is the name of the document in Salesforce.

Active
Mark active for use. If the template is no longer needed, you can deactivate it at any time and reactivate it if it becomes relevant again in the future.

Note: Be sure to activate the template, otherwise it will not be visible for use on the custom button!

SOBject Names
This is the object the document will be generated on. In our example, it is the Quote object.

Owner:
This field will be automatically filled with the user creating the Document Template.

Document Base Name
This is the name of the generated document that will be seen when sent to a customer. To dynamically customize the name, you can include merge fields such as “Quote-{Name}-{QuoteNumber}”. When using merge fields, make sure you use the field name, in the correct syntax, without the “*dataprovider*. “ prefix.

Document Identifiers
This is the name under which the document template versions are saved in Salesforce.

Optional Fields for Document Templates

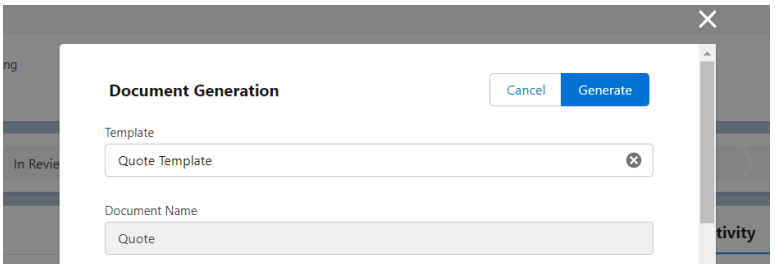
There are additionally optional fields, which can be placed on the Document Template page layout for use:

Deactivate Versioning
When this checkbox is activated, a new document / PDF will be generated instead of a new version.

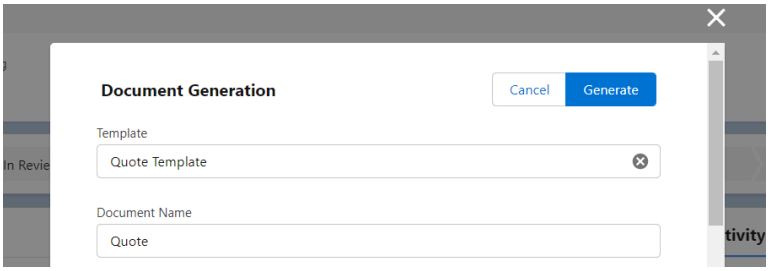
NOTE: On the Quote object, this setting only deactivates the versioning on generated Word Documents, not for PDFs. PDFs on Quotes will continue to be generated as a new version. This is due to the Salesforce data structure and cannot currently be changed.

Allow Document Name Change
Activating this checkbox allows the user generating the document to overwrite the name when generating the document. The name will be pre-filled with the “Document Base Name”, but can be changed by the user.

In this screenshot, the checkbox is deactivated/not in use, so the name cannot be changed. This is the default setting:

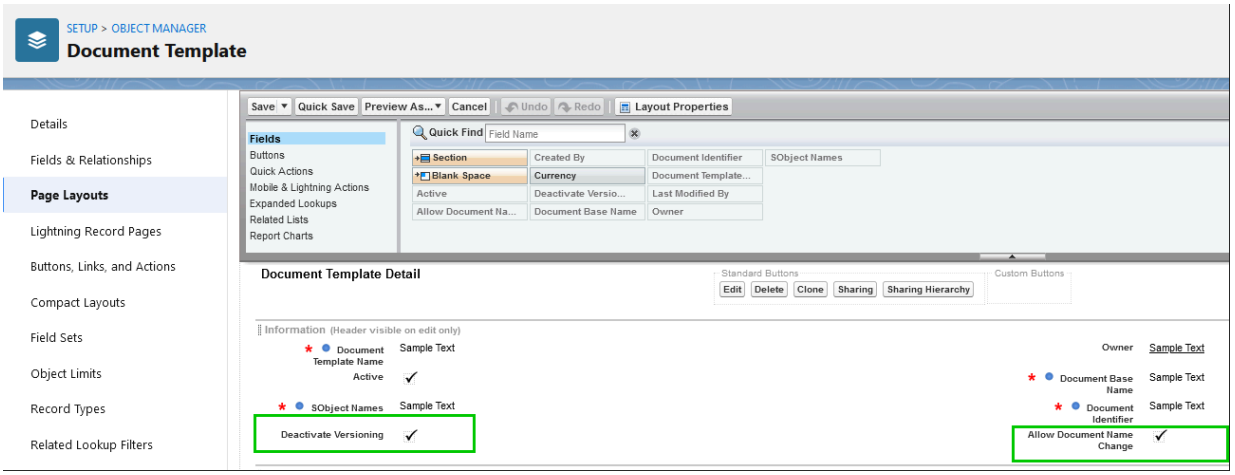


In this screenshot, the checkbox is on the layout and activated. The name can be changed:



If not on the layout, you can add the optional fields by following these steps. If you do not have the system permission to edit the object as shown below, please ask your administrator to do this for you:

1. Navigate to Setup, and click on the Object Manager.
2. In the Object Manager find and click on the Document Template object.
3. Click on Page Layouts and choose the page layout being used.
4. Drag the optional field(s) you'd like to use to the layout and click "Save".
5. These fields will now be shown on all Document Templates using that page layout.

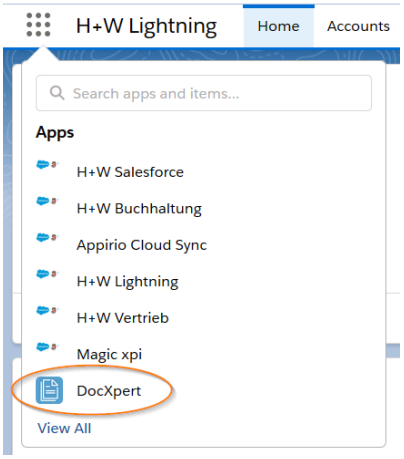


Note: If users cannot see the new fields on the page layout, please check the Field Level Security for these fields.

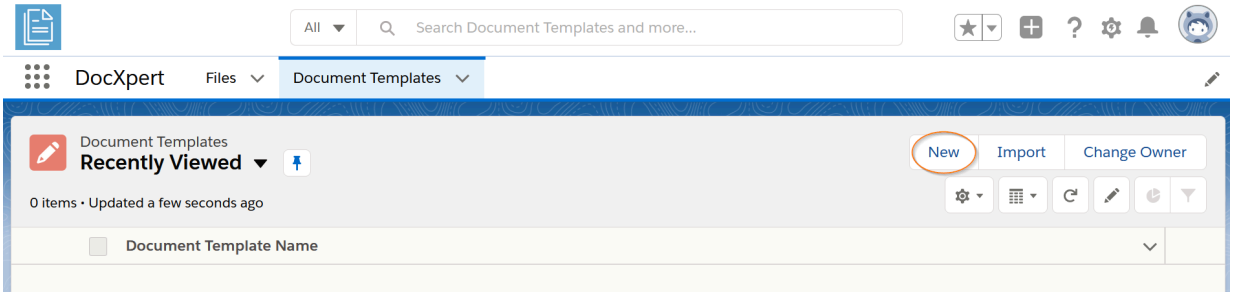
How to Create and Upload Your Document Template

1. Create your template and save it in a place you can find easily on your computer.

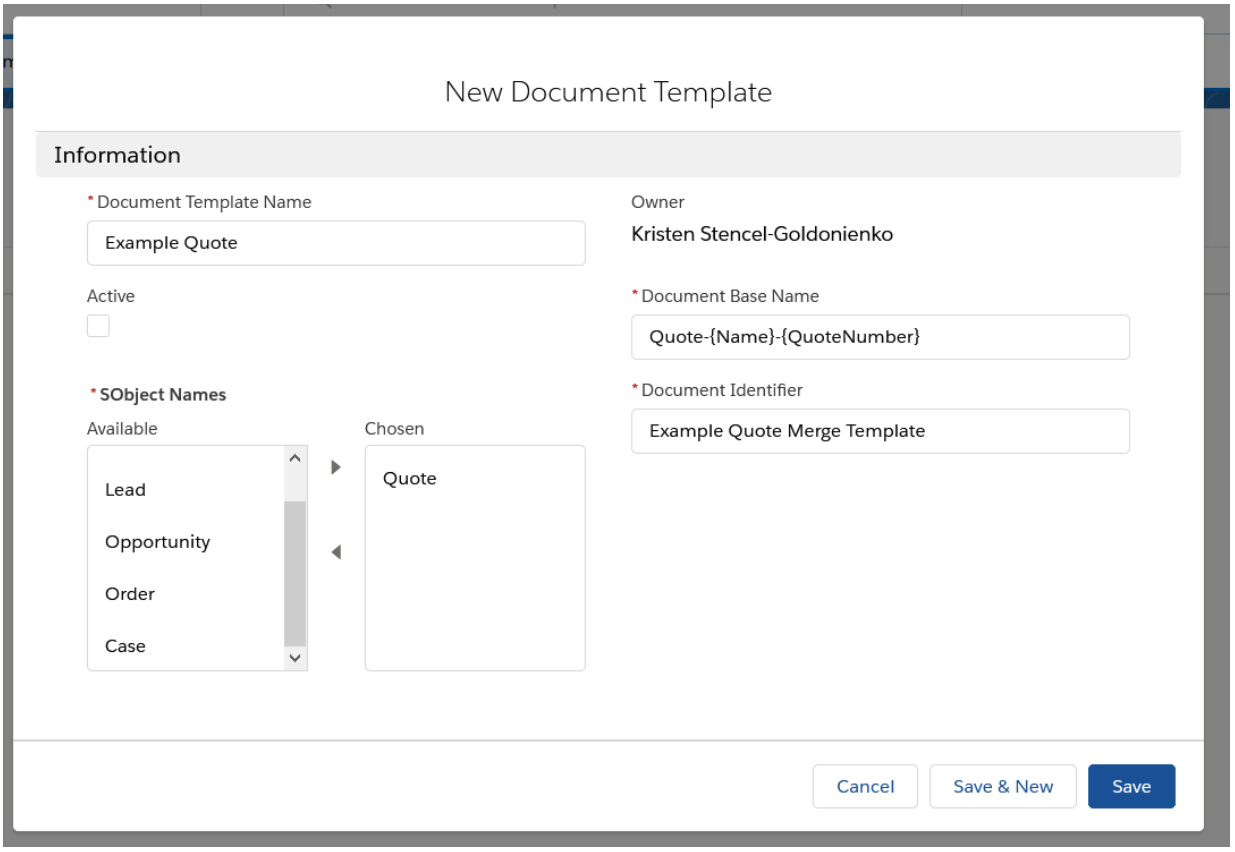
2. Navigate to the Document Generator App in the App Menu:



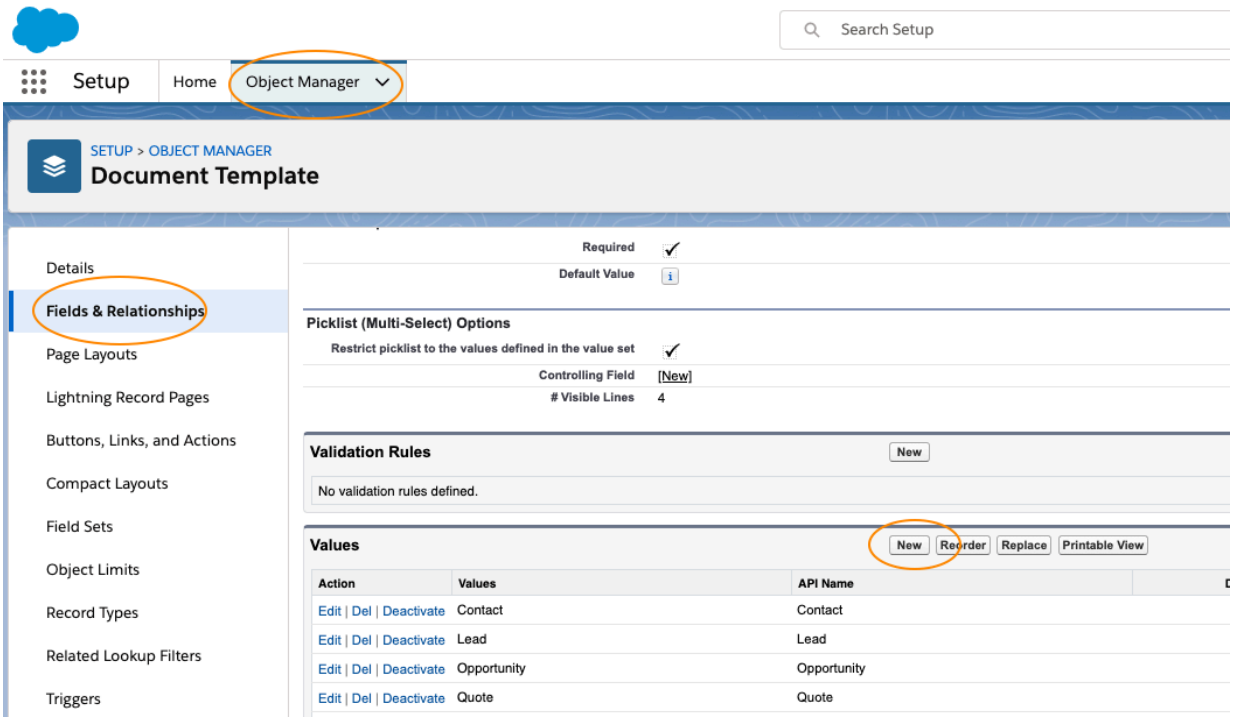
3. Click on the Document Templates tab, then “New”:



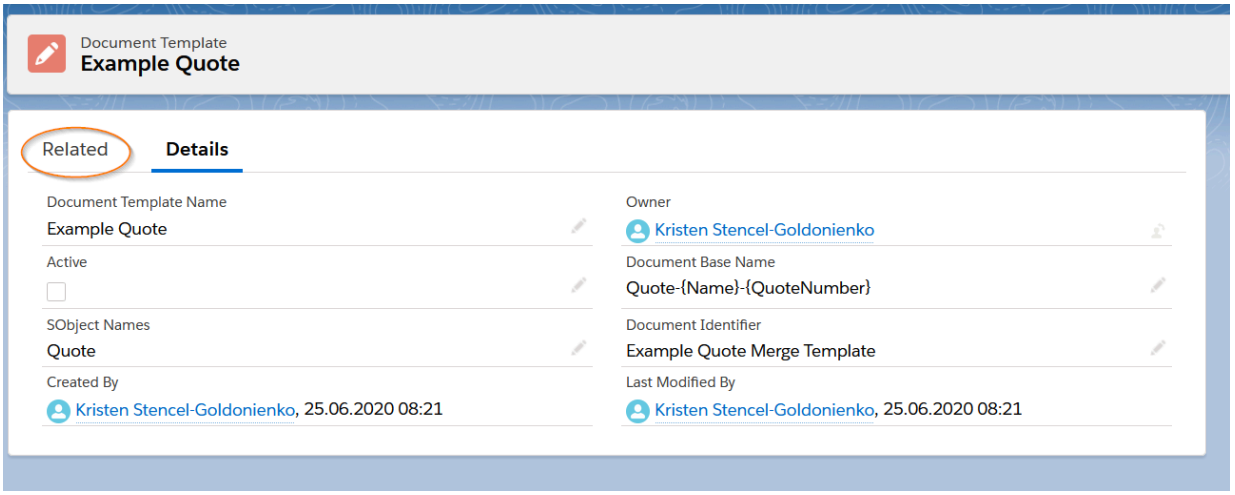
4. Fill out the information. Be sure to choose the object(s) for which your template should be available! Then click Save.



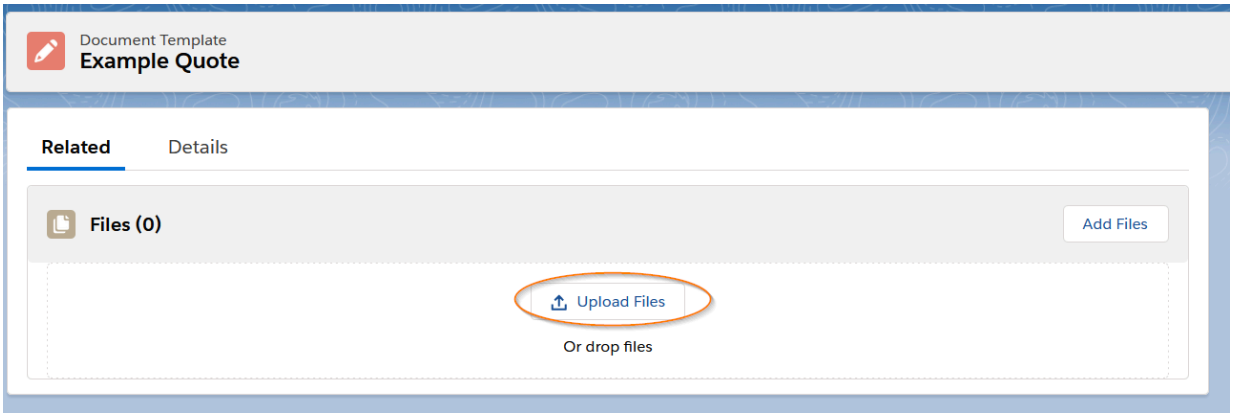
Note: If you cannot find the appropriate object in the list SObject Names list shown, edit the SObject Names multi-select picklist field on the Document Template object via Setup. This will be necessary for custom objects. If you do not have the system permission to edit the object as shown below, please ask your administrator to do this for you:



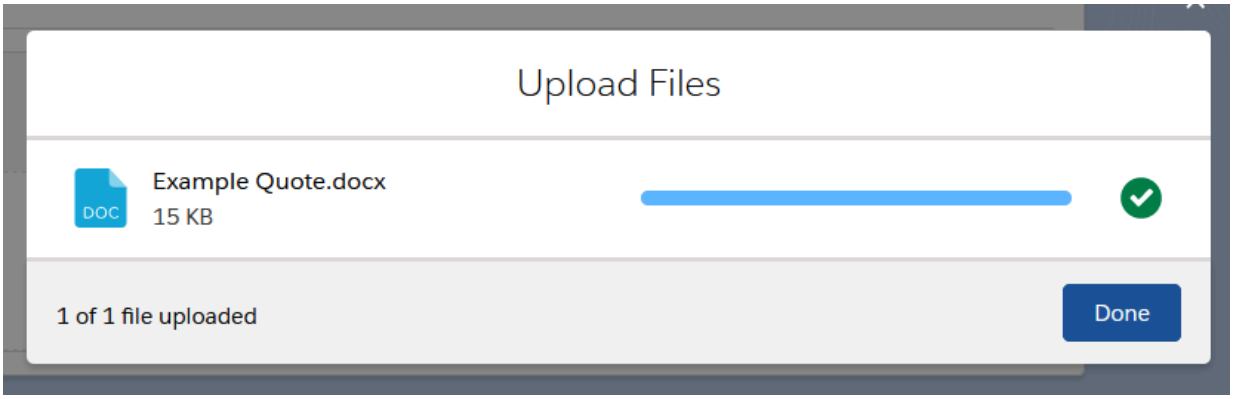
5. After saving, you will see the record view of your newly created Document Template record. Click the “Related” tab on your record to upload your merge template.



6. To upload your merge template, drag and drop your template onto the screen, or click the Upload Files button.



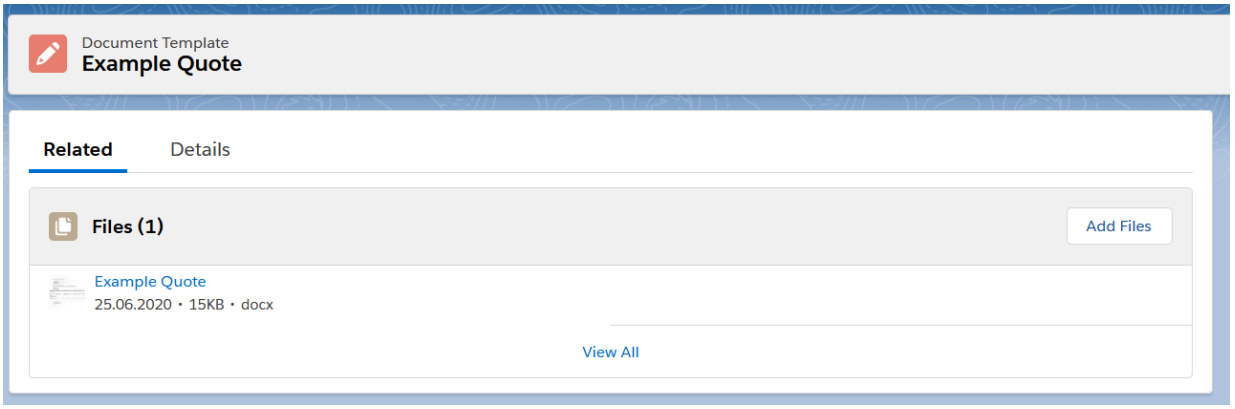
7. Click done, when your template has been uploaded.



Note: The document generator always uses the most currently uploaded version of your template. Be sure your template is in .docx format, otherwise it will not be recognized by the generator and cannot be used!

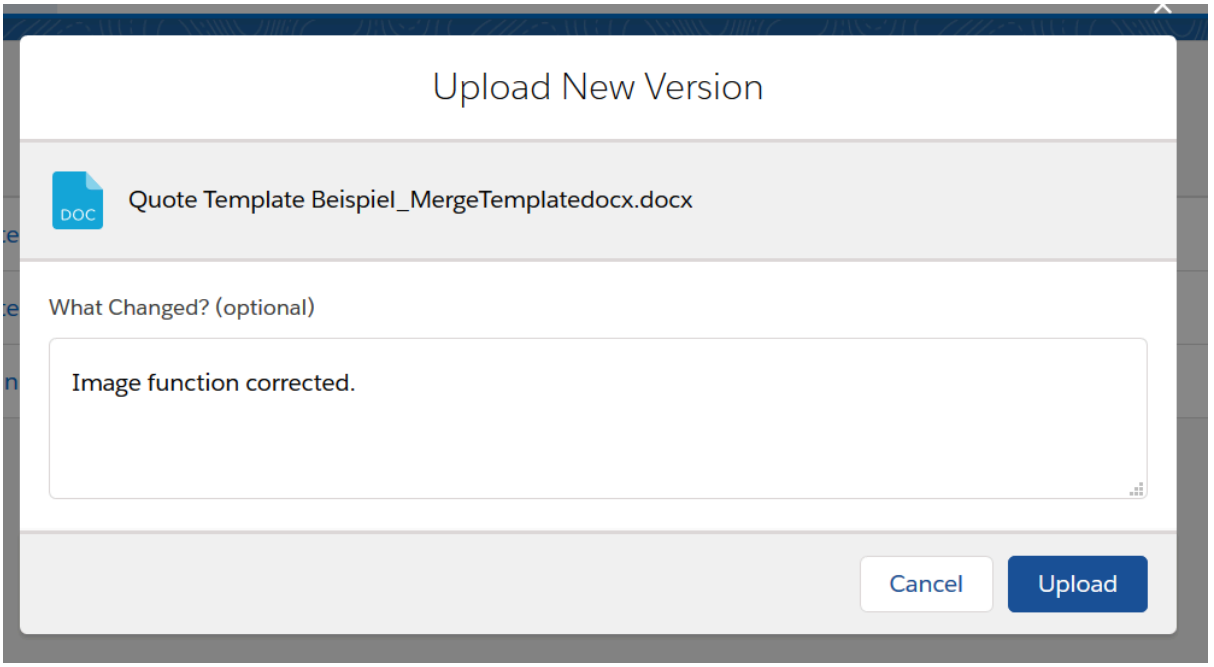
Your template is now saved in your Files library and is attached to your Document Template record. If you need to edit the merge template, simply edit the original template in Word and upload it as a new version in .docx format. Do this using the following steps:

1. Navigate to your Document Template that contains the template you would like to update. Click on the Related tab, then on the name link of the file.



2. This will open the file. Verify that it is the correct file. Then click “Upload New Version” from the drop-down menu under the arrow at the top of the screen and choose the new version of your template.

3. Enter the template changes made (best practice) and click “Upload”:



4. Your edited template will be uploaded and automatically given a version number.

Note: it may take a moment for Salesforce to generate your preview. If you don't want to wait for the preview, just click the x in the upper right corner to exit the preview modus. Your new version will still be uploaded.

To view or edit the Salesforce details or description of your template, navigate to your template document as in in step 1 above (as if you were to upload a new version), and choose "View File Details" from the drop-down menu.

From here, you can:

- View the file size, extension and owner
- Preview your file under the "Preview" tab
- Edit details such as Title and Description under the "Details" tab
- Download the file
- Upload a new version
- View the version history and access old versions
- Delete the file

User Guide

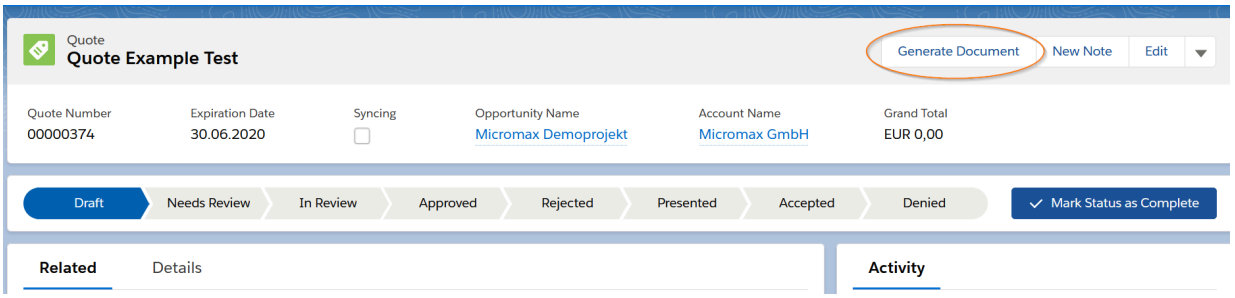
How to Generate Documents

Here are the steps to generate a document:

1. Navigate to the record you would like to generate a document for. In the example case, this is a Quote record.

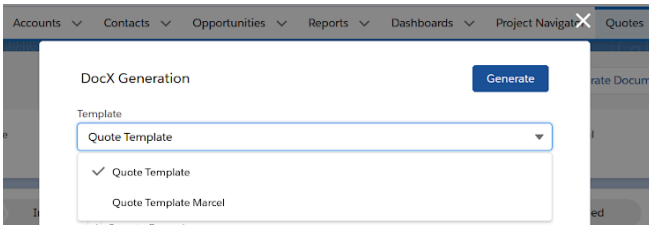
2. In the upper-right of the screen, locate the document generator button. In the example, it is called “Generate Document”.

   Tip: Ask your administrator, if you are not sure of the button’s name.
   Tip: You may need to open the drop-down menu using the arrow to find the button.

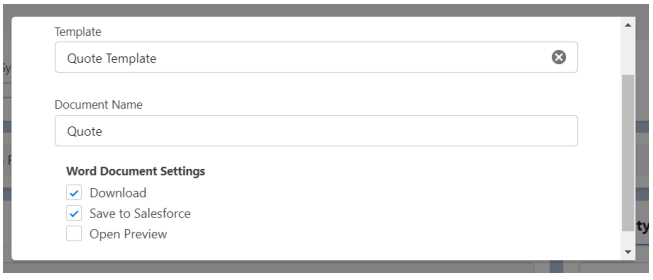


3. Select the correct template in the Template picklist as shown.

   Tip: The picklist automatically shows all templates you are able to use for the record you are on. If you do not see a template you expect to use, contact your administrator.



4a. Choose from the “Word Document” options:



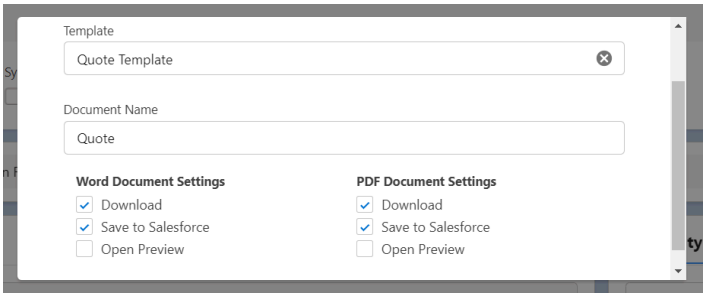
- **Download**: This option prompts you to open the document directly in a word processing program or save the generated document to your local computer.

- **Save to Salesforce**: (default setting) This automatically saves the Word document on the Files & Attachments related list on the record, in this example on the Quote record.

- **Open Preview**: This option can only be chosen when “Save to Salesforce” is also activated. It shows you a preview of your file directly in Salesforce. *Please note that the preview can take a long time and does not always show the generated document in the correct formatting. To make sure your document is formatted correctly, download the .docx / PDF after generation and open it on your computer to check the images and formatting.*

Note: The option “Open Preview” may take a while to generate your preview. You can cancel the preview generation at any time and open the document from the Notes & Attachments related list instead. If the generated document did not save at all, try to generate the document again by activating only the Save to Salesforce option. If it still does not work, contact your administrator.

4b. If the Generate PDF feature has been activated in your org, you will also see the PDF checkboxes.



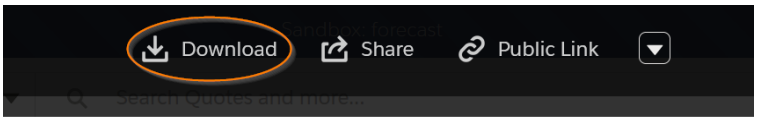
NOTE: if using DocXpert in the Salesforce Mobile App, the download function cannot be used. Save to Salesforce and Open Preview are still usable. In the preview, you can forward and share the generated document via email. To download the documents you can also open the saved files in a mobile text processing App and save it to your device.

5. To personalize the Word document (or convert it to a PDF if the Generate PDF feature has not been activated in your organization), download the generated

document by either activating the “Download” checkbox in Step 4, or by downloading it after generation using the “Download” link on the file.

To use the “Download” link:

5a. Navigate to your Notes & Attachments related list and click on the name of the generated document.
5b. On the top of the page, click “Download”



6. After downloading the generated document, you can edit it and convert it to a PDF on your local computer, then upload it as you would any other file to the Notes & Attachments related list on your record using the “Upload Files” button. Then it will be available to attach when sending an email through Salesforce.