dox42 SalesForce integration

Documentation

Léon Emmer

Summary

This document explains the integration of dox42 into SalesForce.

Content

Document details

Version: dox42 SalesForce integration V 1.0

Author: Léon Emmer

Date: 29 August 2019

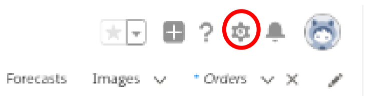
Creating a connected SalesForce App

If you want to read SalesForce data directly from the dox42 Add-Ins, e.g. for testing, you need to provide your credentials. You may pass them as input parameters in order to avoid storing them in the Data Map, or encrypt the credentials using a dox42 Crypto Data Source.

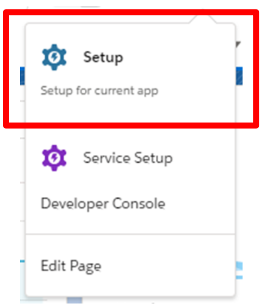
During the dox42 call directly from SalesForce a Security Token is passed to dox42, so these Credentials are NOT necessary. In this case the credentials are stored as named credentials encrypted in SalesForce.

Creating a connected SalesForce App with a service user

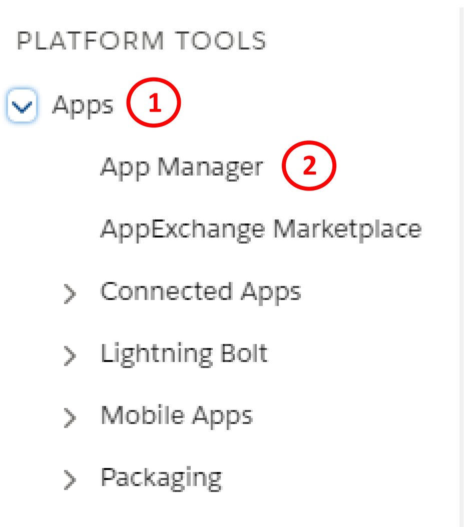
Click on the “settings” gear



Click on “Setup”

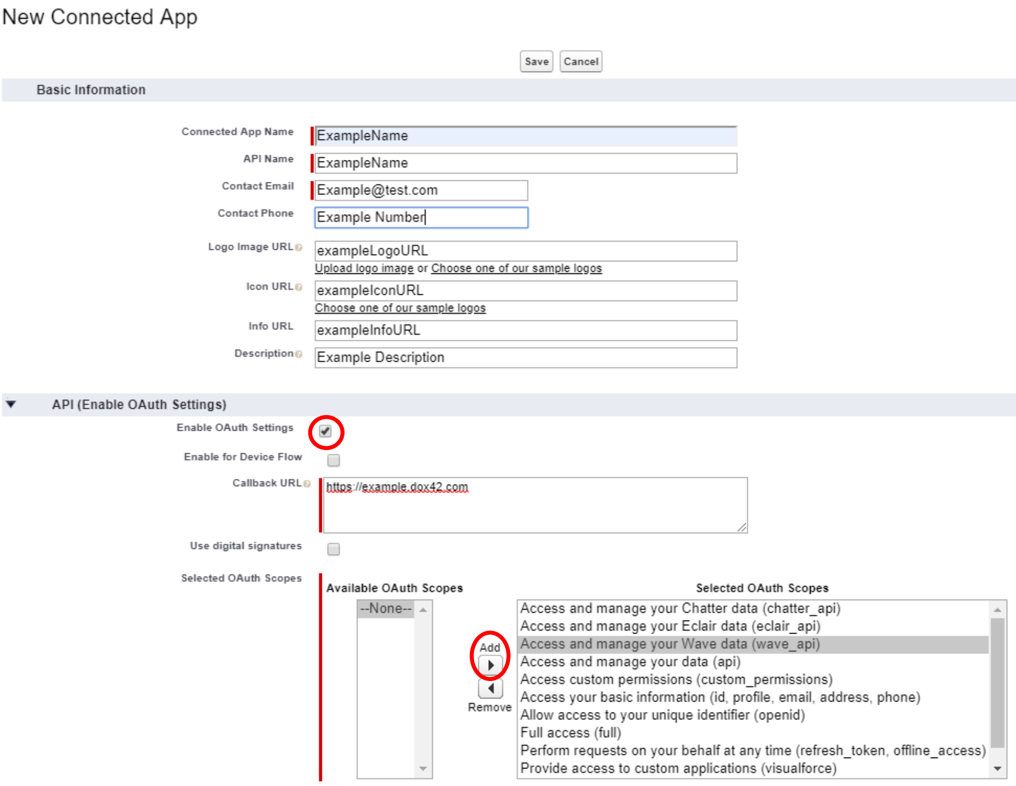


Click on “App Manager”



Click on “New Connected App”

Configure your app as following

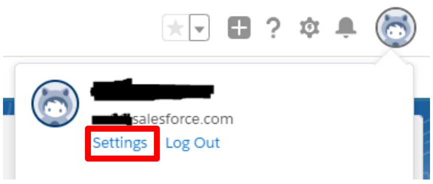


Click on "Save": 

Copy "Consumer Key" and "Consumer Secret"

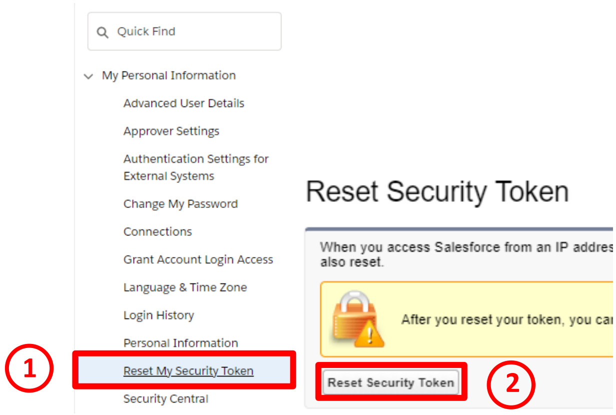
Receive Security Token

Click on your profile and on "Settings"



Click on "Reset My Security Token"

You (service user) will get an email with your "Security Token" which you need to copy as well

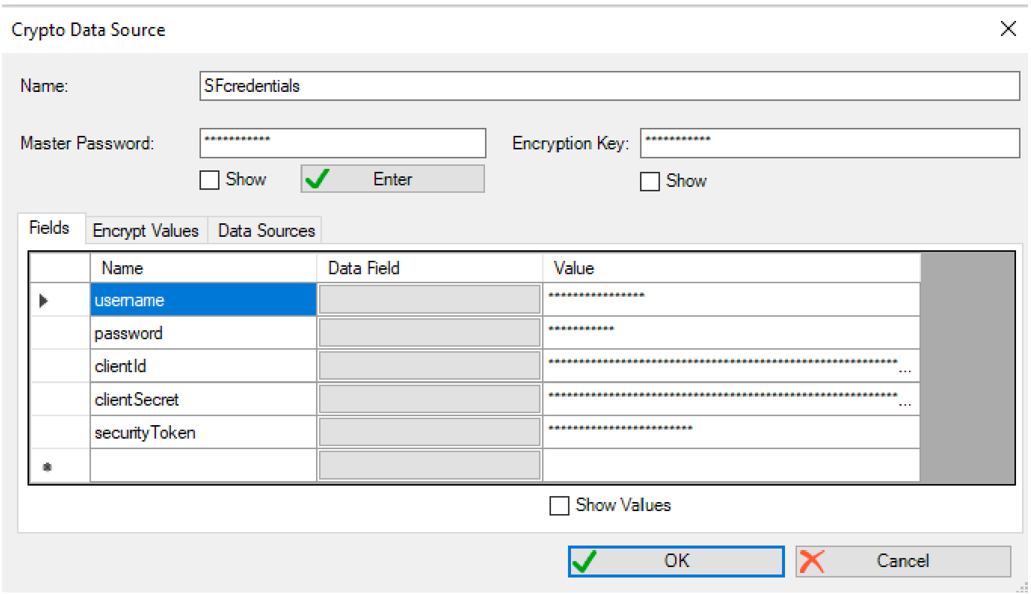


Create a crypto data source in the dox42 office Add-In with your SF-credentials

Create a new crypto data source with the following field names:

clientId = Consumer Key

clientSecret= Consumer Secret



Create a XML Data source in the dox42 office Add-In which is used to generate token for the SalesForce authentication

Name: “TokenGetterSource”

Use the POST method

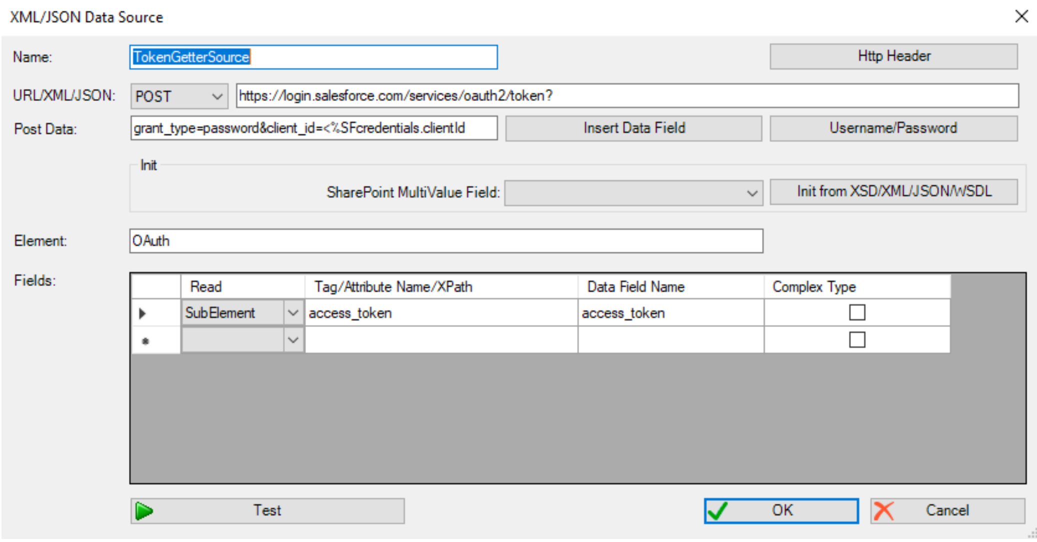
URL: https://login.salesforce.com/services/oauth2/token?”

POST-Data:
grant_type=password&client_id=<%SFcredentials.clientId%>&client_secret=<%SFcredentials.clientSecret%>&username=<%SFcredentials.username%>&password=<%SFcredentials.password%><%SFcredentials.securityToken%>&format=xml

Element: OAuth

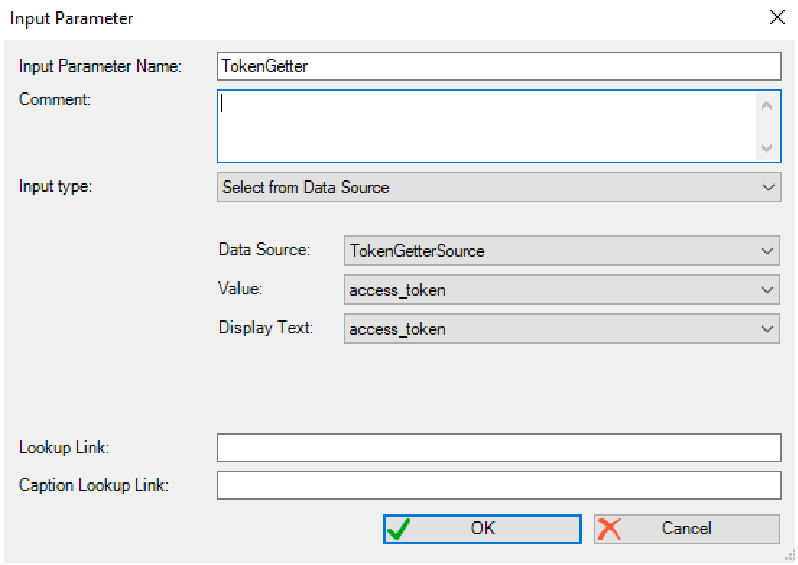
Fields:

	Read	Tag/Attribute Name/XPath	Data Field Name	Complex Type
▶	SubElement	access_token	access_token	☐
*				☐



How to get your data from SalesForce

Create an Input Parameter for the Security Token in the dox42 Add-In



The SalesForce syntax is based on database queries that are implemented in the URL.

To access your data you need to know from which table you want to get the data as well as the field names.

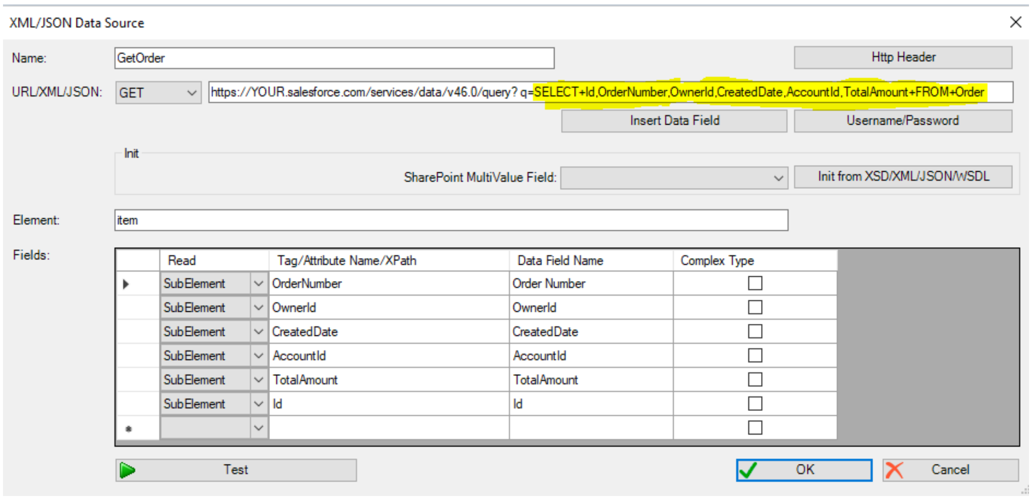
You can look them up here: https://developer.salesforce.com/docs/atlas.en-us.sfFieldRef.meta/sfFieldRef/salesforce_field_reference.htm

Query statement: Functions like a SQL statement. The only difference is that between the operators spaces are replaced with “+”

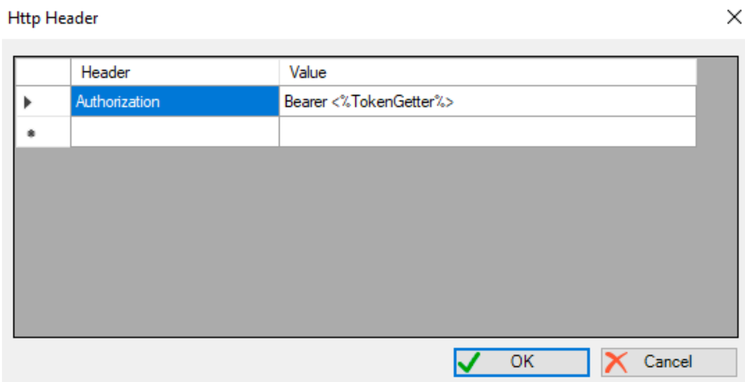
Copy Code:

https://[YOUR SalesForce address].salesforce.com/services/data/v46.0/query?q=[Query statement]

In our example data comes from the table *Order*:



Also important is to set your Http Header as following:

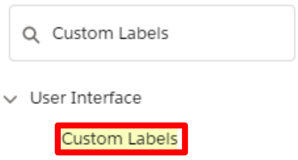


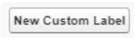
Set up a dox42 Call from SalesForce directly using Security Tokens from SalesForce

Set up SalesForce labels

Use Custom Labels to store Client_Id and Client Secret of your connected App.

Go to settings and search for “Custom Labels”



Click on “New Custom Label”: 

Parameter	Value
Name	C_Client_Id
Description	Client-Id
Value	[your client Id]

Click on “Save & New”: 

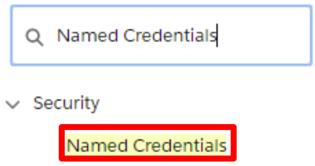
Parameter	Value
Name	C_Client_Secret
Description	Client-Secret
Value	[your client Secret]

Click on “Save”: 

Set up SalesForce “Named Credentials”

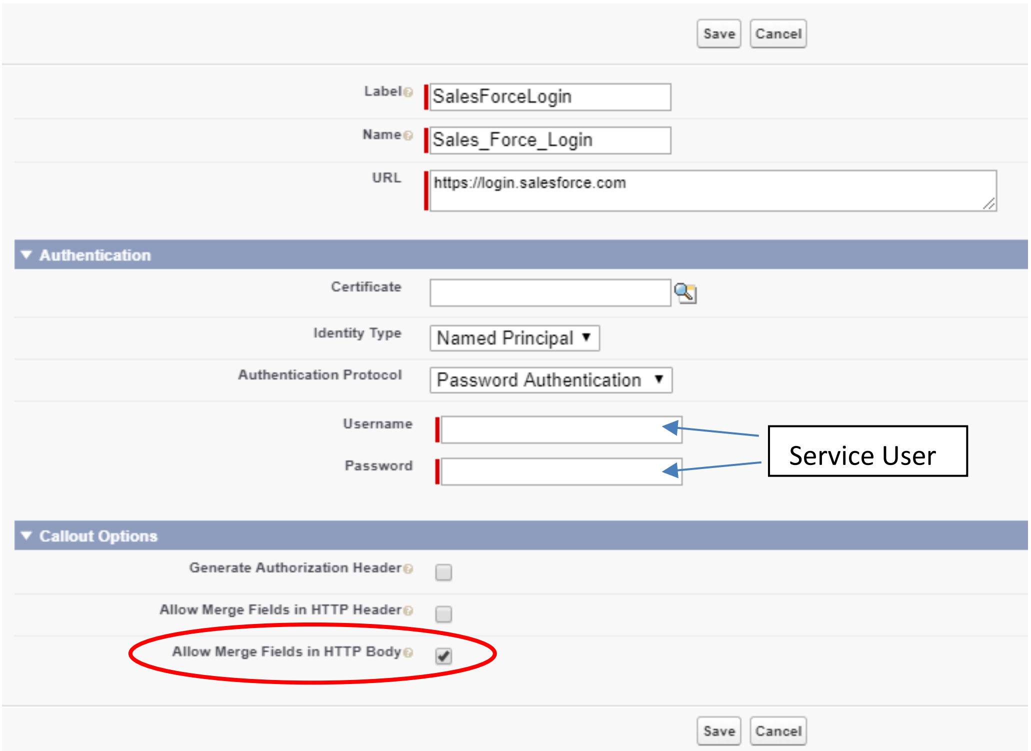
Use Named credatials in SalesForce to store your Service User Credentials encrypted.

Go to settings and search for “Named Credentials”



Click on “New Named Credential”:

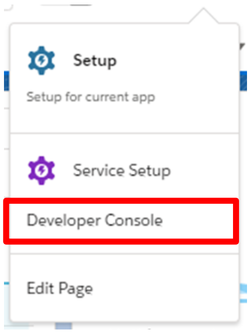
In the setup use the following parameters (You need to use a Service User):



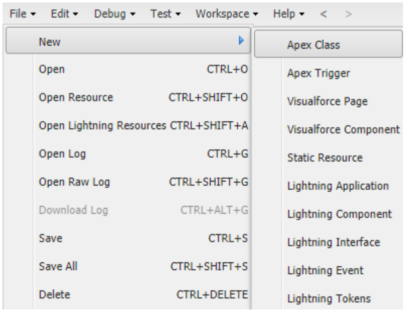
Create Visualforce Page

Click on the gear:

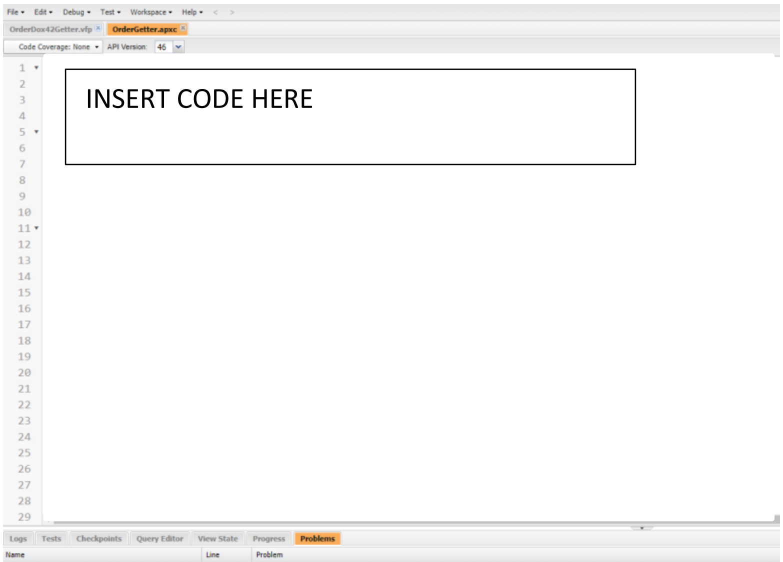
Click on “Developer Console”:



Click on File >> New >> Apex Class



[Apex Class-Name can be chosen freely]



Copy code

(replace all [ ] with your data, insert your parameters in the dox42 REST link with splitting the string at the correct position and adding your parameter out of SalesForce to it):

public with sharing class **[Your chosen Apex Class name]** {

//Get selected Order from VisualForce Page via Constructor

private **[Name of table that is in use]** o1;

public **[Your chosen Apex Class name]** (ApexPages.StandardController stdController) {

```
        o1 = ([Name of table that is in use])stdController.getRecord();
    }
    //Method that is called by the VisualForce Page "popUp"
  public PageReference redirectToDox42() {
     //Client Id set in a Custom Label
     String clientId = Label.C_Client_Id;
     //Client Secret set in a Custom Label
     String clientSecret = Label.C_Client_Secret;
     //String where token is extracted to
     String token;
     //Setting up connection for requesting token
     HttpRequest req = new HttpRequest();
     req.setMethod('POST');
     req.setEndpoint('callout:Sales_Force_Login/services/oauth2/token');
     string bdy = 'grant_type=password' +
        '&client_id=' + clientId +
        '&client_secret=' + clientSecret +
        '&username={!$Credential.Username}'+
        '&password={!$Credential.Password}'+
        '&format=xml';
     req.setBody(bdy);
     Http h = new Http();
     HttpResponse res = h.send(req);

     //Extracting token out of XML response
     String storage = res.getBody();
     storage = storage.substringAfter('<access_token>');
     token = storage.substringBefore('</access_token>');
```

//Call dox42 Server with handing over Order-Id and token

String dox42Url= '**[Your dox42 Rest Link]**';

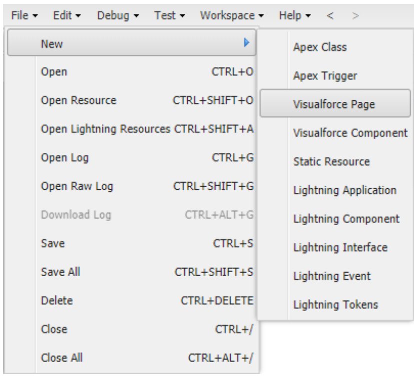
PageReference newFloodOnlyUrl = new PageReference(dox42Url);

return newFloodOnlyUrl;

}

}

Click on File >> New >> Visualforce Page



[Visualforce Page-Name can be chosen freely]

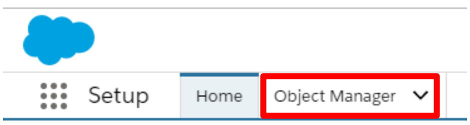
Copy code:

<apex:page standardController="**[Table you want to use e.g. Order]**" extensions="**[Name of your Apex class e.g. OrderGetter]**" action="{!redirectToDox42}">

</apex:page>

Set up SalesForce Action

Click on “Object Manager” in the settings:

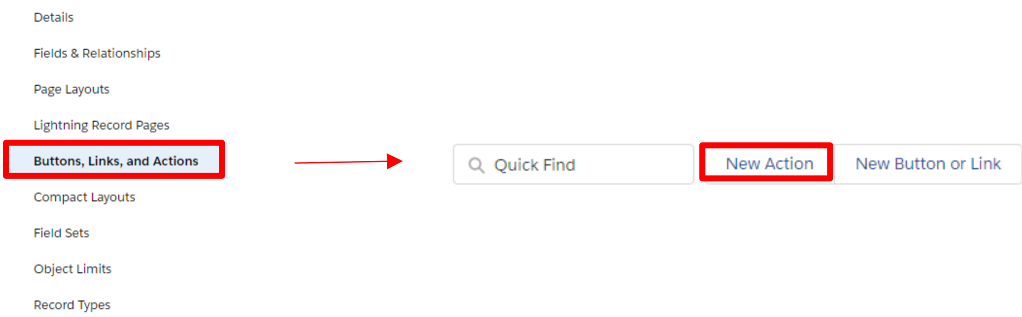


Then click on your label you wish to implement dox42:

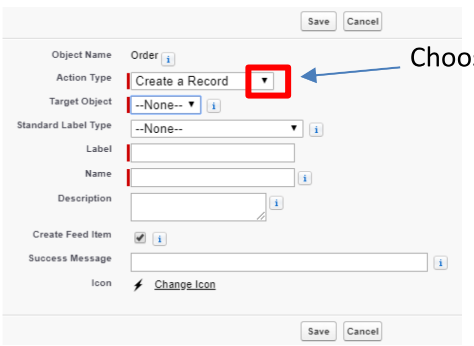
(In this example “Order”)

Image	Image
Individual	Individual
Lead	Lead
Macro	Macro
Messaging Session	MessagingSession
Messaging User	MessagingEndUser
Opportunity	Opportunity
Opportunity Product	OpportunityLineItem
Order	Order
Order Product	OrderItem
Price Book	Pricebook2
Price Book Entry	PricebookEntry
Product	Product2
Product Consumption Schedule	ProductConsumptionSchedule
Quick Text	QuickText
Recommendation	Recommendation
Scorecard	Scorecard
Scorecard Association	ScorecardAssociation
Scorecard Metric	ScorecardMetric
Social Persona	SocialPersona
Task	Task

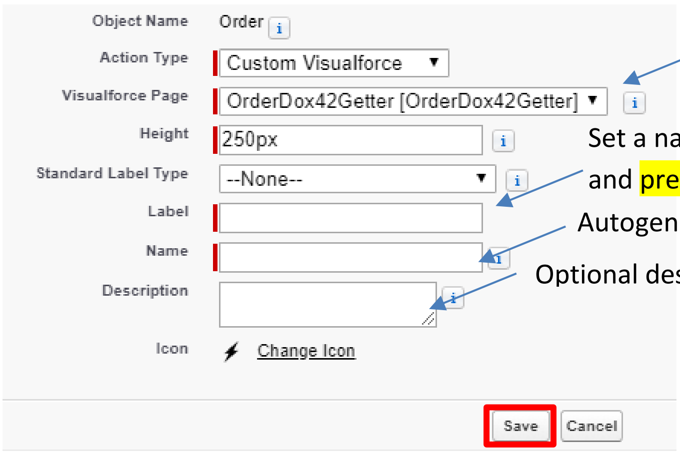
Click on “Buttons, Links and Actions” and on “New Action”:



Setup:



Choose ”Custom Visualforce”



Choose your Visualforce Page

Set a name for your button and press “tab”

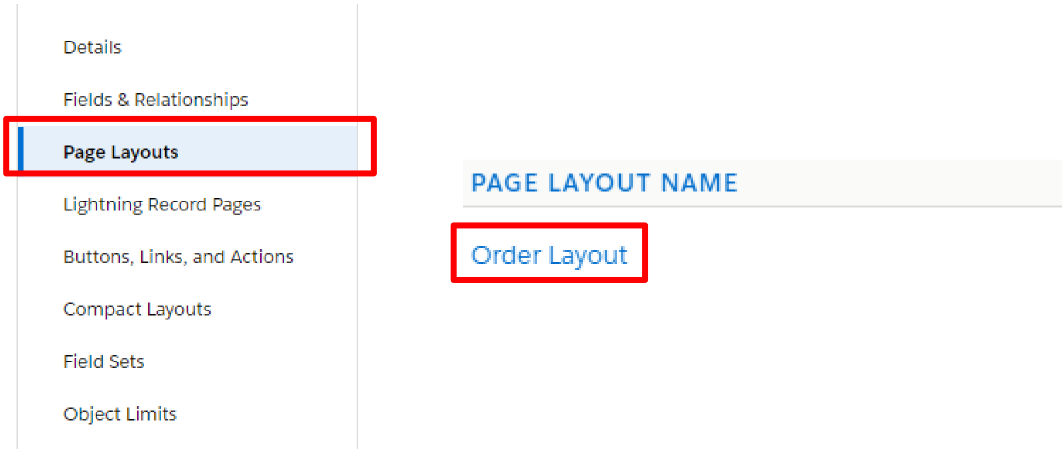
Autogenerated by pressing “tab”

Optional description

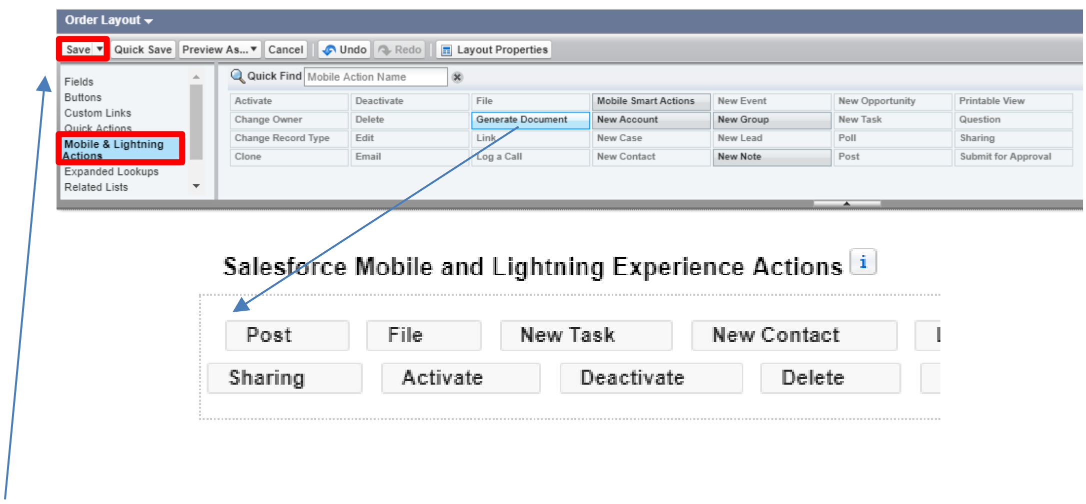
Click on Save

Next go back to the label you want to edit and click on “Page Layouts”, select your layout you would like to add the action to.

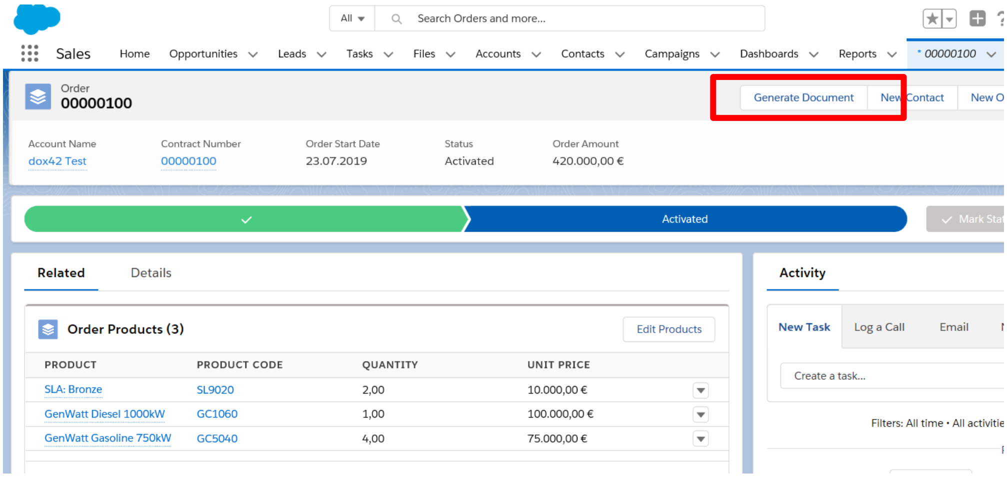
(e.g. “Order Layout”)



Select “Mobile & Lightning Actions”
Drag and drop your created action into this field.
(It is recommended to put it on first place.)



Click on Save.


Search Orders and more...
Sales
Home
Opportunities
Leads
Tasks
Files
Accounts
Contacts
Campaigns
Dashboards
Reports
* 00000100
Order
00000100
Generate Document
New Contact
Account Name
dox42 Test
Contract Number
00000100
Order Start Date
23.07.2019
Status
Activated
Order Amount
420.000,00 €
Activated
Related
Details
Activity
Order Products (3)
Edit Products
PRODUCT
PRODUCT CODE
QUANTITY
UNIT PRICE
SLA: Bronze
SL9020
2,00
10.000,00 €
GenWatt Diesel 1000kW
GC1060
1,00
100.000,00 €
GenWatt Gasoline 750kW
GC5040
4,00
75.000,00 €
New Task
Log a Call
Email
Create a task...