Learn Flow and Auto Create Orders

Version 9.19.2

Installation and Set-Up

Thank you for your download!

Don’t forget to check out Taveza Lightning Templates. Over 100 templates to build better Lightning Pages! One time charge of only **$125.** Thats it!

Taveza Templates

Also, should you need help with your Salesforce Org, please reach out to me to see how I may be of service.

Howard Stall

Whats New

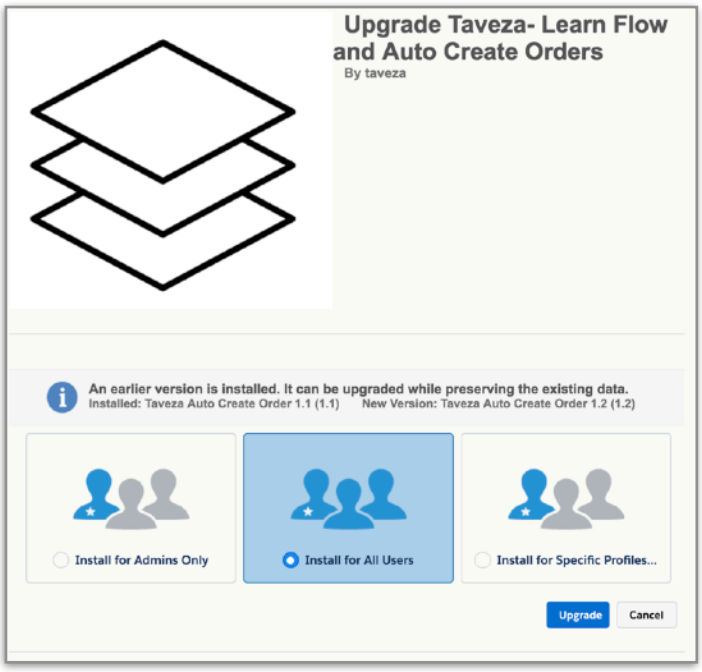
Release 9.19.2

Pass Records From A Tab Flow template was added to provide a working example of how to select records in a list view and pass them into a flow.

Preinstallation Steps

You must have Quotes enabled in your organization before the package will install. Goto Set Up, search on Quote, and click enable. You do not need to add the Quote related list, or do any additional set up steps. Nor do you have to use Quotes in your Org. There are Flow templates in this package that require the Quote functionality to be enabled prior to the installation starting.

Installation from the AppExchange



Install the package for all users during the set up from the AppExchange. Don't worry, this package has nothing that will run automatically or change the UI.

User Permissions To Use The Flows in This App

The primary user permission to utilize Flows it the Run Flows permission. This will need to be enabled for EVERY user that will interact with the flows in your Org. If you place a Workflow or Process builder rule on the Opportunity object to create orders on Closed Won, then any user that could Close Win an Opportunity would need the Run Flows permission.

The other permissions that you will need to be concerned with are Object and Field Level Security. Users must be able to read fields, edit, and create records. With out the proper permissions on the object and fields, your Flows may not perform or function as designed.

Manual Set-Up Steps

Custom Settings

Included in this app is set of custom settings, OrderFlowSettings, you can use in the Flows, and leverage as an example to learn how to include Custom Settings in your flows.

- Generate Contracts - If set to True, than a contract is generated if now has not been created on the Opportunity record

- GenerateFromQuote - This is used in combination with the MasterOpportunityFlow to help in auto selecting if you will be creating Orders form an Opportunity or a Quote. If you are using Quotes, go ahead and mark this as true

- GenerateSeperateOrders - If this is set to true, this will create an individual order for EACH product line on your Opportunity or Quote. If you have (5) product records, you will get (5) Orders

One final item is that as you work with each flow for the first time, Deactivate and Reactive the Flow. Then begin to create your own Flows or versions from the Template Flow you have selected to work with. This ensures that the Flow uses the latest API of Salesforce in your Org.

Purpose of This App

The goal with this app is to provide you a way to generate Orders the way you want to with out code. Secondly, provide you with a set of Flow Templates so that you can build new flows, modify these, and leverage the power of Salesforce's Lightning Flow Builder. This app and manual are meant for the non-developer, but developers are also welcome!

This app is NOT 100% complete. You will need to perform the following task:

- Determine which Flow will help you generate Orders that will fit your business processes
- You may need to modify the security model in your Org so that these flows function as you have designed
- You WILL have to modify the flows so that standard and custom fields are mapped correctly
- You will need to determine what action will start your Flow
- At some point you will be involved in troubleshooting your flows

If you are looking to learn flow, think of this manual and app as a guide. It will not provide ALL of the answers your are looking for. These templates are designed to provide you with examples. Very basic, but working flows covering all sorts of the most common use cases. As you dive in to these, remember, as long as you are working and learning in a Sandbox, no harm will come to you or your Org. Good old trial and error, along with some examples is an amazing way to learn. At any point, please feel free to reach out!

Flow Templates Included In The Package

Clone Opportunity With Products - This Flow will provide you with a solution and example on how to clone a Parent and its child records. In the Lightning UI, the actual ability to Clone an Opportunity with its products is not there. With some field mapping, and a quick button you can add this back to your Lightning experience. Or learn what to do so you can leverage this else where in your org.

Generate Contract - This Flow does just that, generates a Contract and attaches it to your Opportunity, Account and Order. If you have indicated that Contracts are to be generated using the custom settings, this action will be performed automatically when running some of the flows in this app. You can also launch this flow from a button or process if you desire this to operate on its own.

MasterOpportunityFlow - This Flow is used to show you how you can pass the Opportunity Record ID to a 'Master Flow' and using the Custom Settings, decide which one of (4) Flows to utilize. By using a single workflow rule or process, you can use logic inside of a Master Flow to determine which one to invoke. Saving you from having to build multiple workflow or processes with varying degrees of logic.

Opportunity To Order- You decide what action on the Opportunity creates an Order along with Order Lines from the Opportunity Products.

Opportunity To Order For Each Product- Same flow template as the Opportunity To Order, but you will get an Order for EACH product on the Opportunity

Quote To Order- Exactly like the Opportunity template for orders, except you will be able to use fields from Quotes and product from Quote Lines.

Quote To Order Per Product - You guessed it! Same as the previous templates, expect this uses fields from the Quote and generates an Order for each Quote line.

Opportunity Product Counter- An example of how to count child records and post a result to the parent. Yes, out of the box Salesforce can count Opportunity Product records. But, in the sprint of this app, the Opportunity object was chosen. This example can be modified to work on any object, and child, regardless of the relationship type.

Pass Records From A Tab- This flow illustrates how you can select a set of records on a tab, and pass those ids into a Flow. This is VERY useful to perform mass actions on list views with multiple record types.

Words Of Wisdom

Below are what I would consider the most popular items to remember as you start working with Flows. I have learned many the hard way. Mainly like most, I read the manuals, best practices, and look at Trailhead when I get stuck vs before you start. I too love a since of adventure, but please take a moment and read these.

- Yes, you can create new Flows and versions in production. Until you feel confident in using this new part of the 'Force', please stay in a sandbox. You can easily promote new Flows with change sets. Making changes is not a best practice, but yes you can do it

- As you create new versions of your flow, put in the description field what has changed. This WILL help you later understand why that version was created

- Don't be too quick to delete prior versions of your Flows. Yes, looking at a long list of inactive Flows or versions will mess with the worst of your OCD. Keep them there until you are certain the latest version has been working across a broad set of your user base

- When you use the Debug function, it WILL create, edit, delete records. Debug does not mean a 100% simulation of your flow's actions

- As you think about performing a record lookup, create, or edit. Think about all of the items you would have to do with a report or the data loader.

    - Specify the exact record to lookup - Id, and other criteria

    - When you create or edit a record, again, just like the data loader, all the fields need to be mapped

- Think about how the flow will operate as a USER. Does the USER have access to read, edit, or perform the create action you are designing? Remember, depending on the way flow is used, it utilizes the USERS permissions

- Think about how to stop a flow from running again accidentally. The Opportunity To Order flow updates the field on the Opportunity, Order Generated. In case the Opportunity is Closed Won, then Opened and corrected, and then Closed Won again. New Orders will not get generated if that field is already checked true. You can't solve for every use case, but think about how the flow is triggered and when it should not be

- You can launch a flow from a button, a link, or any URL based 'hack' you can think of. Flows can also be launched from process builder rules. Apex can also call a Flow.

- No, you don't need a screen in order to use flows in your Org. None of the Flow Templates in this app use screens to perform the actions. Think of using flow to perform a complex Quick Action thru a click or auto launched from behind the scenes.

- **SAVE FREQUENTLY**. Set a timer and **SAVE FREQUENTLY.** The universe has a knack of knowing when you have finally built or solved a problem in flow, and then the internet goes down. Your browser crashes, laptop runs out of power. The list goes on.

- If you use these templates, and intend for them to be kicked off behind the scenes, you will save them as Autolaunched flows

- If all of a sudden your flow doesn't read fields, deactivate it and re-activate it. At times when flows move between Orgs, the API version of the flow 'gets confused' . Deactivating and reactivating will clear that issue.

Kicking Off A Flow

There are several ways to put your finished flow to work. This is something you should think of ahead of time. Understanding how flows can be launched in Salesforce may have an impact on how you design your end to end process using flow.

1. From a Button- Review this Trailhead module for amazing details on how to launch a flow from a button, Launch Flow From A Custom Button. For creating Orders from an Opportunity or Quote, this should be done automatically using a Process. Now, let's say you wanted to have the Sales Rep create the Contract before they could Close Win an Opportunity. Using the GenerateContract Flow Template, you would create a new Flow and complete your mappings. We will name it OppContractGen

    - Create a button on the Opportunity, and add in the following URL: /Flow/OppContractGen? OpptyID={Opportunity.ID}

        - The URL can also be found on the Flow record in the URL Field. After the ? Is when the input parameters are added

    - Add the button to the page layout

    - Activate the Flow

2. From a Process using Process Builder - A very great example of this can be found here, on the Automation Champion website Launch Flow From A Process. However, in the sprit of this manual, let's assume you want to auto create orders on Closed Won of your Opportunity.

    - Create a Process on the Opportunity Object, when the record is created or Edited

    - Criteria will be when Opportunity stage is Closed Won

    - Action will be Launch A Flow

- Flow Variable will be OppID and the value will be the Opportunity ID

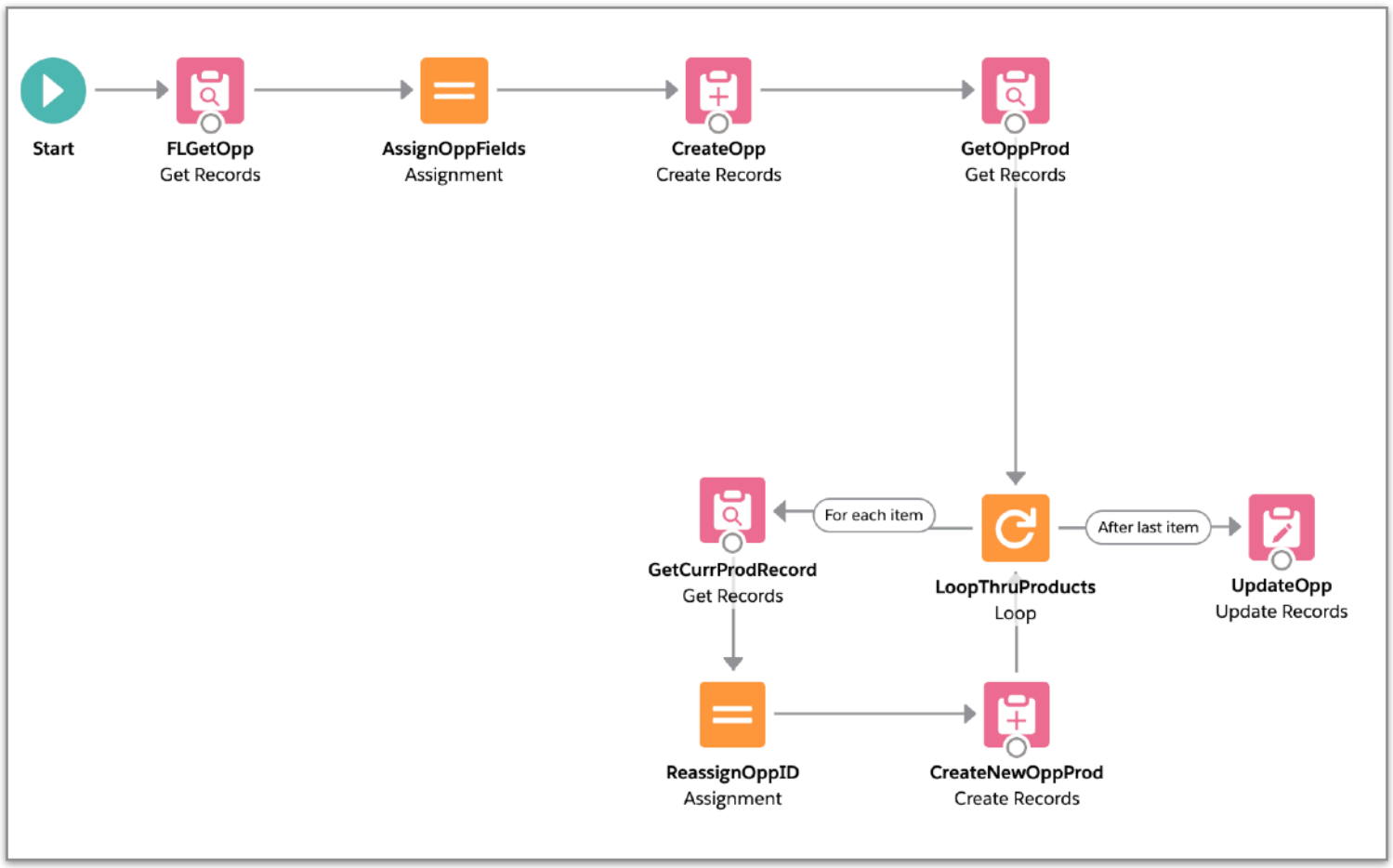
- Thats it! Save and Activate your process

3. From a Flow - Yes, you can launch a flow from a flow. If you plan your business processes correctly, and build flows that support smaller concepts. You can call a flow from a flow based on logic you design. Remember those elaborate flow charts in the meeting you drifted off in, well flow can support similar concepts.

4. From Apex- We would cover that in this app, but Apex can call up your flows, just like flows can call Apex.

Digging Into The Flows

Clone Opportunity With Products



Let's assume this flow is going to be kicked off from a Button you are going to add to the page layout. That button is going to launch the flow you have created by passing it the Opportunity ID of the record you are on. Boom! It then clones the Opportunity and its products. It does this by passing the Opportunity ID in a variable called **CurrentOppID**.

1. Go to the Clone Opportunity With Products Flow and click Save As. Give your Flow a new name, as this will now be your flow copied from the template. This keeps the template intact, while you goto town on your new flow.

2. **FLGetOpp**: Goto your new flow, and open it up. It should look like the screen above. First, we are going to ensure that you retrieve all the

field values from your current Opportunity record. The **FLGetOpp** elements is a Fast Lookup element that has been designed to get values for a SINGLE record. Double click on the **FLGetOpp** and you will see a label, API, and object. Pretty standard stuff. Next you will see the conditions. Remember what I said about the applying the concepts of building a report or using a the data loader? A Fast Lookup can pull in multiple records or one. It does what YOU tell it, so in this case, you need to add criteria to pull up a record by its ID. But wait, how does the flow know that? You will be using Variables, and in this case, a variable named **CurrentOppID** is being used to represent the ID of the current Opportunity record. This variable is only used for the ID. As you look more at this element, you see its set up to store only the First Record, and keep the values in a single variable. That single variable is called **CurrentOpp**. What that variable represents is where to find the current Opportunities values and the fields you have chosen to store. For each field you want to clone into a new Opportunity, you need to it ADD HERE. If you do not READ the records fields, you can NOT use them later on. So, take a moment and add all the fields from the Opportunity you will need copied onto a new record.

What happens if I chose separate variables? Great question. A lot more work. When you elect to use separate variables, you will need to create a variable for EACH field you want stored. For AccountID, you would need a variable called CurrentOppAcctID, and repeat for every field. The single **CurrentOpp** record variable used above stores all of the fields you have identified under its 'umbrella'.

3. **AssignOppFields**: Next we need to assign the current Opportunity records field vales to a new variable. Just like the dataloader, when you query data and then insert it, the step in-between is mapping. Think of this **AssignOppFields** (Assignment Element) as a mapping element. We will map the old to the new. So, you had a single record variable to hold the old elements, we will need a variable to hold the new elements. This is where the **NewOppRecord.XXX** came from on the left side of this assignment element. Just like you had to create a single record variable to contain what you looked up, you need a variable to map those values against. Once you create it on the Opportunity, you can then map items at the field level. Now, for every field you added in the Lookup element, ensure it's mapped in this Assignment element.

4. **CreateOpp**: This element will create our new Opportunity record based on the mapping we defined in our Assignment variable (**AssignOppFields)** to the **NewOppRecord** variable. There is nothing

to do here except make sure your users have permission to create Opportunity records.

5. **GetOppProd:** Now we are going to get the Opportunity Product records for the current Opportunity using a Fast Lookup similar to what we did before. Remember you need to define how to find the product records, so in this case just ones that belong to the current Opportunity. Otherwise Flow will get ALL of them. The difference on this lookup is that we will select All Records, and not first record, because we could have more than 1. Again, on this one make sure you list all the fields you want cloned below.

6. **LoopThruProducts:** This is a loop element, and its job is to process thru a set of records 1 by 1. It does to each record what you define in the steps after it. You create a set of actions, and then after the last action its only purpose in life is to go to the next record. Think of it as a traffic cop at an intersection. Once you turn left and go around the block, the cop doesn't know what you did. But if you come back around the block, he will direct you once again to turn left. This loop is working thru the set of Opportunity Products you looked up and stored in the record set in the previous element, the CurrOppProductRecords variable. As it picks a record to work with, Flow needs to assign those values to a variable that will 'drive it around the block'. Each record will run thru a set of actions, and Flow needs something to reference while it does that. So, you need a loop variable. Therefore, we have a variable called CurrentProdRecord to identify a single record.

7. **GetCurrentProdRecord:** So, now let's get the details of a single record from out list. We will be using a lookup element to get the fields from the CurrentProdRecord variable. As you recall, you will need to list out all the fields you need mapped out just like you have before.

8. **ReassignOppID**: With this assignment element we will learn a new 'trick'. Instead of creating yet another new record variable to map the old to the new. We are going to use the current record variable's values to make the new record. Just like as if you downloaded a set of records with the dataloader, cleared the ID and input the new Opportunity ID to insert a copy of records on a new Opportunity record. This is going to do just that. Wipe out the current ID, and replace the Opportunity ID with the one created a while back.

9. **CreateNewOppProd**: This create element uses the CurrentProdRecord we have been modifying to create a new Opportunity Product Record.

10. **Loop Continues**: The loop element continues until it runs thru the entire list.

11. **UpdateOpp**: This update element is used to update a field called Cloned From ID to allow you to track back where the Opportunity was cloned from. Make sure your users have edit access to this field.

Flow Recap:

1. Add all the fields you need to the FLGetOpp element

2. Assign the fields from Step 1 in the AssignOppFields element

3. Add all the fields you need to the GetOppProd element

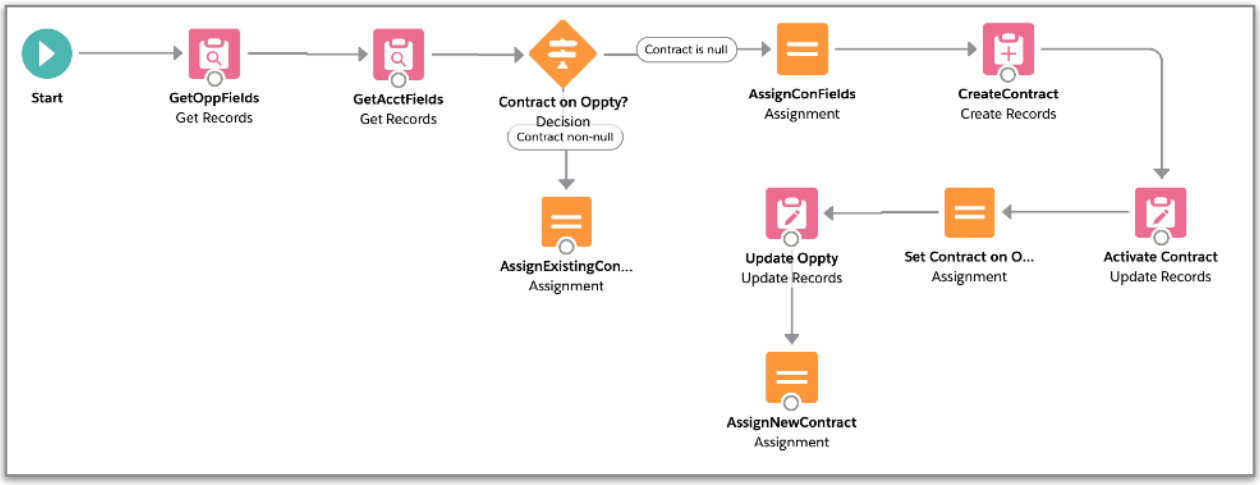
4. Add all the fields you need to the GetCurrProdRecord element

5. Ensure the field level security on the profiles that will use this flow are correct

6. Activate your flow

7. Create your button or process builder to use the flow

Generate Contract



With this flow, you need to decide if you want your users to click the create Contract button that you have created. Or if you want this to happen in the background. In order for this to happen in the background, you will need to ensure the custom setting OrderFlowSettings - GenerateContract is set to true.

Because you read everything about the last flow, the explanations on this are going to be a less. Some of the set-up steps are the same when it comes to getting record values and assignments of those values.

1. **GetOppFields** and **GetAcctFields** are the lookup elements that provide information from the Opportunity and Account. Ensure that all fields are listed in these two elements where you need their values.

2. **Contract On Oppty?**: This is a decision element and a perfect example of checking to see if the action you are about to perform has already been taken care of. This is especially true if you have a flow launched from a button. Users can easily click it multiple times, and create multiple records. This decision element could be moved to after the Start element in this flow, but as long as no record actions are taking place before the decision. All is ok.

3. **AssignExistingContract**: This assignment has been put in place so that the variable **EndContractID** can be passed to another flow if needed. In both outcomes, if there is a contract or one is created. That variable gets set and can then be used in other flows.

4. **AssignConFields**: Just like in the Opportunity Clone Flow, you will need to map your Contract fields to the Opportunity and Account fields that will provide the values for them. You will notice there is a variable called oContract to contain all of the values for the 'about to be created' record.

5. **CreateContract**: Nothing to do here. Flow creates the Contract from the variable you mapped values into from the previous step.

6. **Activate Contract**: This is a simple Update Element to change the Contracts status field.

7. **Set Contract On Opp**: This assignment element assigns the Contracts ID to the Contract field on the Opportunity. While you could just update the Opportunity using an update element, this is an example that shows you how you can assign values to a current record for it to then be updated. Just like you have now learned to do with Create elements.

8. **Update Oppty**: Simple update the Opportunity based on the assignment from the previous element.

9. **AssignNewContract**: This assignment element functions just like the other one, **AssignExistingContract** ensuring that the end variable of **EndContractID** is set and can be used else where if needed.

Flow Recap:

1. Add all the fields you need to the GetOppFields and GetAcctFields elements

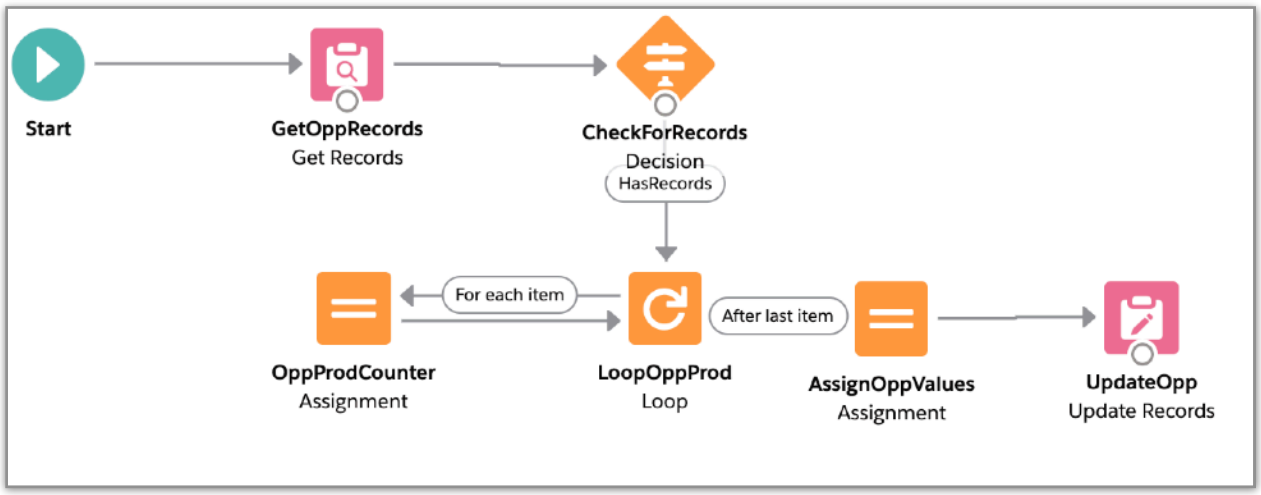
2. Assign the fields from Step 1 in the AssignConfields element

3. Ensure the field level security on the profiles that will use this flow are correct

4. Activate your flow

5. Create your button or process builder to use the flow if your not going to let the other flows call it as they are designed to in this app

Opportunity Product Counter



Here is a cool flow that provides an example of how to count child records, and place the sum of the records onto an object. This is very useful when you have run out of Roll Up Summary fields, or you have a standard lookup relationship.

When using flow to count records, you will need to put some thought into when the flow will run, because records will only be counted when the flow is invoked. Unlike, Roll Up Summary fields that automatically run on Salesforce in the background. A "count" flow will need an edit action, such as a Process triggered when fields change on the parent or child record.

If you would like to count records that meet a certain criteria, say when a date field is populated/edited, and the date is less than today. You would build a Process that triggers when the date field is changed, and is less than today to run the flow.

Should you just want a count of child records, you will build a Process that runs when a record is created. In addition you will need to also think about when a record is deleted. You would need to build a flow that deletes the record from which it is called, and then count records. You could easily override the delete button on the object in question to do this.

The only time your ‘count’ Flow would not run is if there were actions that deleted records using the API.

In the above example, let's walk through what has been built.

1. **GetOppRecords**: This element is designed to get all of the records you would like to count. This is where you would also add the criteria as to which records to query and count. Just like you would define with a Roll Up Summary field.

2. **CheckForRecords**: This decision is placed in there so that your Flow can stop if there are no records. This prevents it from generating errors when no records are found.

3. **LoopOppProd**: Typical loop element that will run thru the list of records you have defined in the previous steps.

4. **OppProdCounter**: In this Assignment element you will notice that there is a variable called XXX. As the loop element runs through the list, this assignment element increments the variable by 1. If your list of child records contains 8, the end value of this variable would be 8.

5. **AssignOppValues**: This assignment takes the variable with the final count, and assigns its value to a field on the Opportunity object.

6. **UpdateOpp**: As in the other examples, this element updates the record with the values you have assigned.

Flow Recap:

1. Add the criteria you need that will query the records you would like in the GetOppRecords element

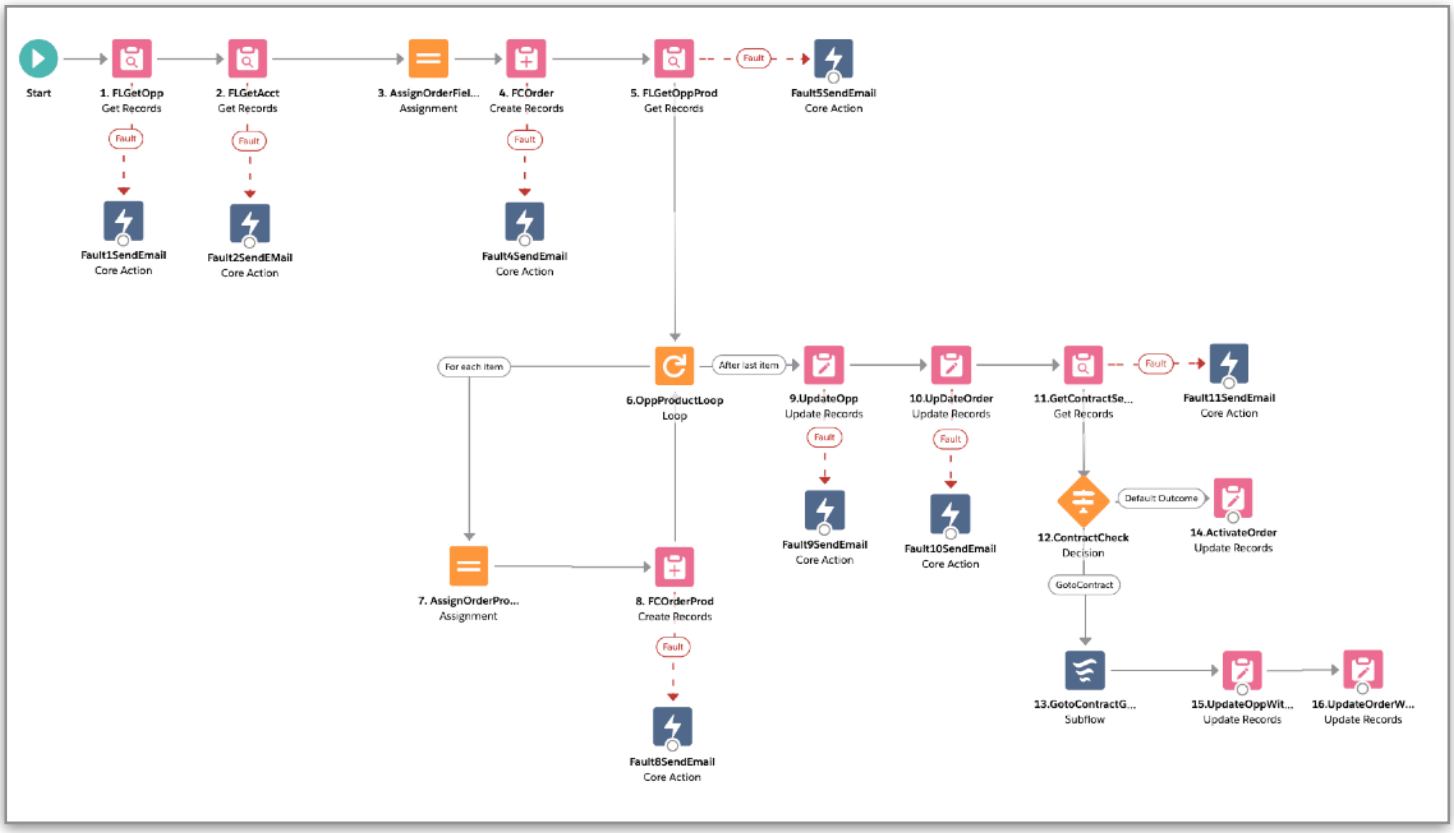
2. Decide where you want the results of your count placed on your record, and update AssignOppValues element

3. Ensure the field level security on the profiles that will use this flow are correct

4. Activate your flow

5. Build your process to trigger the flow

Opportunity With Order



At this stage in the manual, you should be catching on that some of the actions you need to perform for each flow are the same. Define your fields (Lookup), map them (Assignment), and then update or insert (Create).

Since you are starting to master that part of flow, the next item to think about is the concept of placing faults in your flows. This flow has a fault built in for each element that allows it. Each fault sends a custom email to provide the user with more details on where the user was when the flow encountered an error. If you recall, flow does send a system generated email when something fails, but often that email is very generic and can be hard to figure out where exactly things broke down.

Faults can offer you an enhanced troubleshooting capability as you roll flows out in Orgs with many variations of profiles and permission sets.

In this flow you will also notice we start to check for if a Contract is to be generated by checking the Custom Settings for the Org. If a Contract is required, the Flow continues on to the Contract Flow. Rather than duplicate the contract creation process in each Flow, you can have one Flow to perform this function and use it when required. Leaving the Flow you started and coming back.

Flow Recap:

1. Add the fields in the **FlGetOpp** and **FLGetAcct** elements that you need to populate fields on your newly created order

2. Map the fields in the **AssignOrderFields** assignment

3. Add the fields in the **FLGetOppProd** element that you need to create your Order Product records

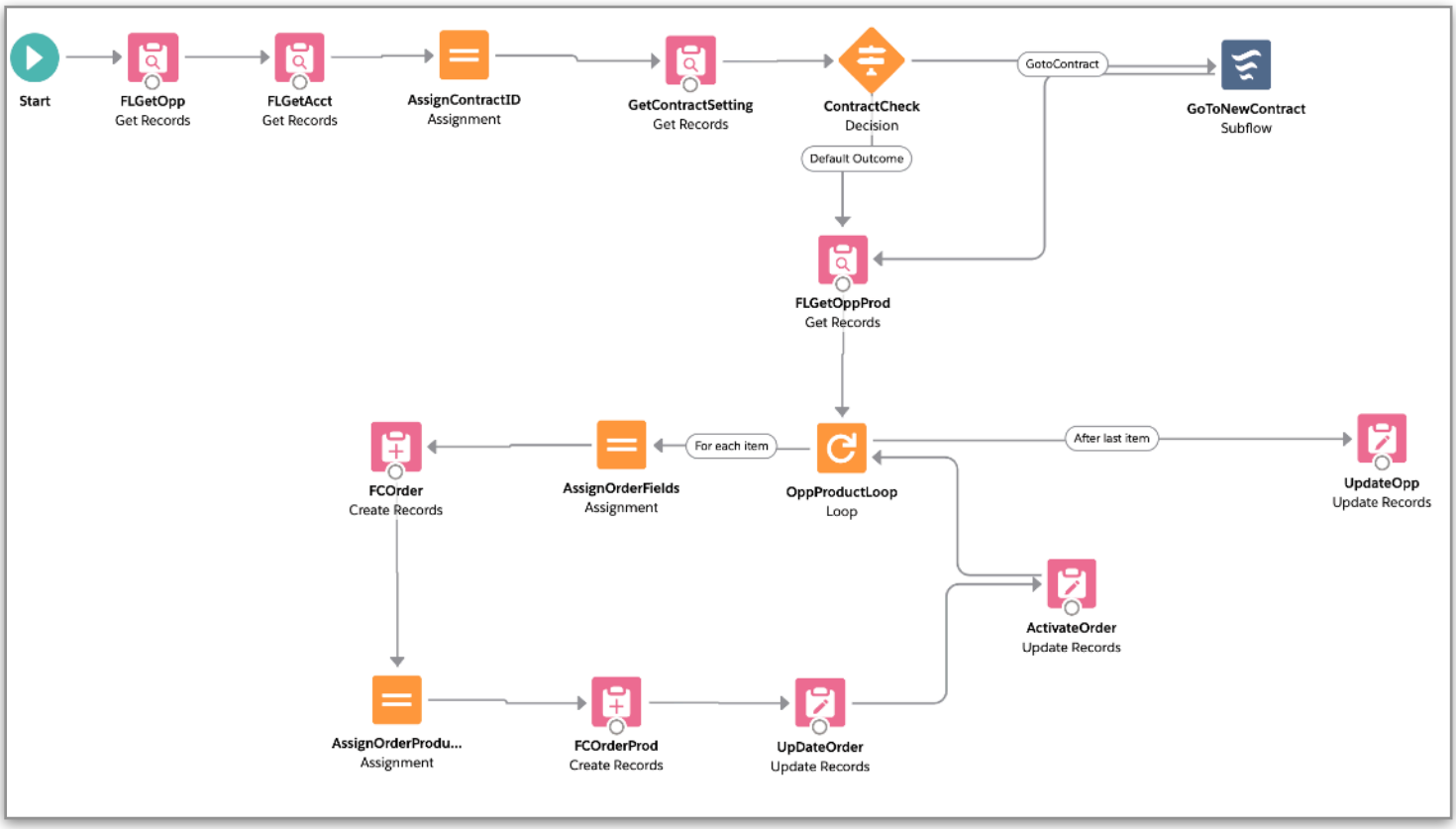
4. Map the fields in your **AssignOrderProduct** element

5. Ensure the field level security on the profiles that will use this flow are correct

6. Activate your flow

7. Build your process to trigger the flow. Alternatively, you can launch this flow from a button if you would your users to launch this flow manually

Opportunity To Order For Each Product



This Opportunity to Order flow is exactly the same as the previous one, except it will generate an Order for each product. If you notice the elements to create the order are also in the same 'loop' as creating the products under the Order. This is a perfect example of how you can perform multiple actions while working through a list of records with actions in a loop.

Flow Recap:

1. Add the fields in the **FlGetOpp** and **FLGetAcct** elements that you need to populate fields on your newly created order

2. Map the fields in the **AssignOrderFields** assignment

3. Add the fields in the **FLGetOppProd** element that you need to create your Order Product records

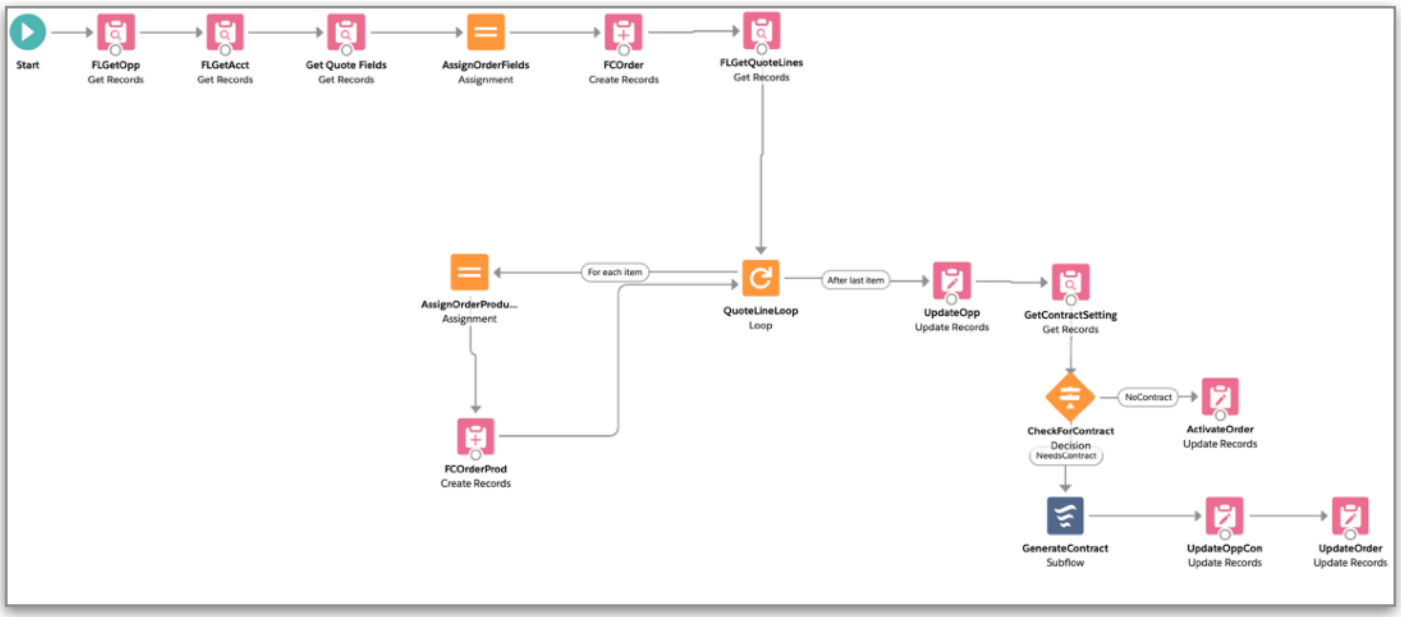
4. Map the fields in your **AssignOrderProduct** element

5. Ensure the field level security on the profiles that will use this flow are correct

6. Activate your flow

7. Build your process to trigger the flow. Alternatively, you can launch this flow from a button if you would your users to launch this flow manually

Quote to Order



If you are using Quotes, this is the flow for you! Just like the Opportunity to Order flows, this one utilizes the Quote and Quote Line items. Same exact 'process' of reading values, mapping, and creating an Order and its Order Lines.

Flow Recap:

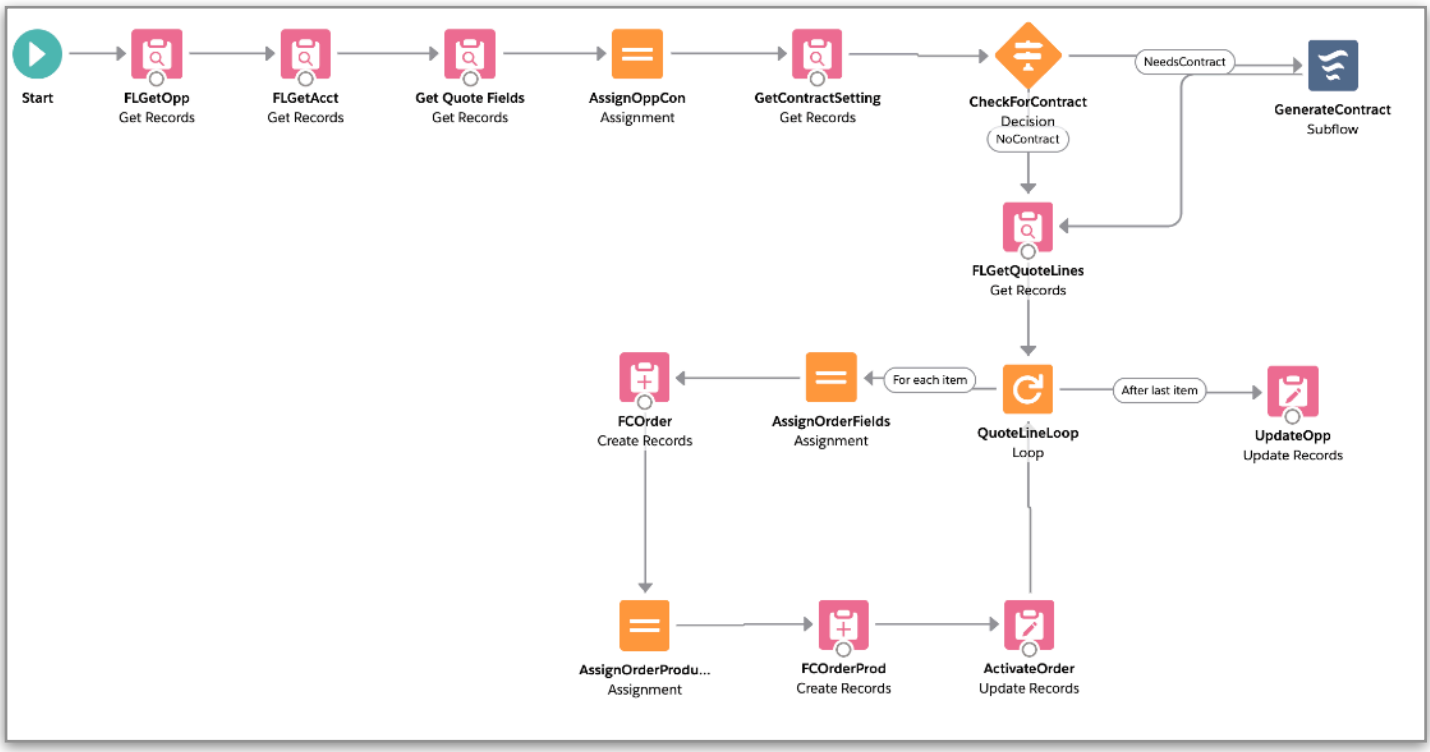
1. Add the fields in the **FlGetOpp, FLGetAcct, Get Quote Fields** elements that you need to populate fields on your newly created order
2. Map the fields in the **AssignOrderFields** assignment
3. Add the fields in the **FLGetQuoteLines** element that you need to create your Order Product records
4. Map the fields in your **AssignOrderProduct** element

5. Ensure the field level security on the profiles that will use this flow are correct

6. Activate your flow

7. Build your process to trigger the flow. Alternatively, you can launch this flow from a button if you would your users to launch this flow manually

Quote to Order per Product



Lastly, this flow generates an Order for each Quote Line item. This flow operates just like the Opportunity To Order Per Product flow. The steps for setting up this flow are below, and should be familiar to you by now.

Flow Recap:

1. Add the fields in the **FlGetOpp, FLGetAcct, Get Quote Fields** elements that you need to populate fields on your newly created order

2. Map the fields in the **AssignOrderFields** assignment

3. Add the fields in the **FLGetQuoteLines** element that you need to create your Order Product records

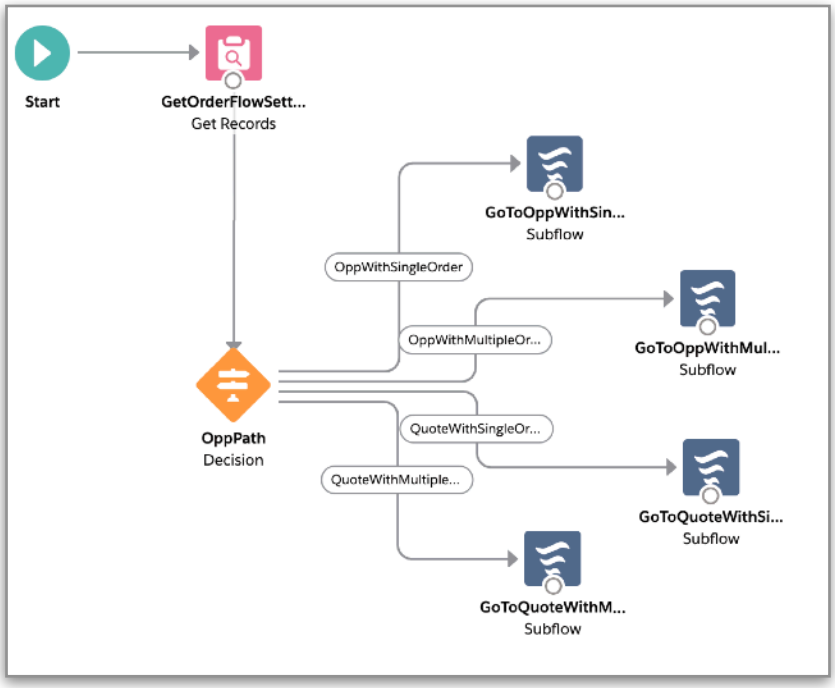
4. Map the fields in your **AssignOrderProduct** element

5. Ensure the field level security on the profiles that will use this flow are correct

6. Activate your flow

7. Build your process to trigger the flow. Alternatively, you can launch this flow from a button if you would your users to launch this flow manually

Master Opportunity Flow



Having a master flow is a term that is commonly used to describe a flow that is called upon, and then it determines which flow to use next. In this example, we pass along the Opportunity Id to the flow. It then queries the custom settings for the app, and determines which flow to use to generate orders. This is an example of building flows that perform and can operate independently of each other. The master acts like an 'If-Then' statement except you have built the 'If' and the 'Thens' as elements in a flow.

While this type of master flow concept will not prove that useful for creating Orders since you really only need one of these. Knowing that this concept can be applied else where in your future flows is important. This will help you to avoid:

- Atomic flows - everything happens in a single flow, and parts should be separated out

- Multiple processes built for multiple flows - While this is not a very bad thing, it is nice to have a single process builder trigger a single flow to reduce events firing in the background

- Reduce workflow rules by combining their conditions into your flows - This also helps narrow down changes in your org

Pass Records From A Tab

The working example of this flow can be found on the Account Tab. There is a button called FlowExample that when added to the list view, will show you how you can use a screen flow to update a set of records. Select the list of records in the list view, and then click on the FlowExample button. This flow does NOT need a screen flow, you can tie the button to an auto launched flow to perform actions in the 'background'.

This flow leverages an Apex Class and a Visualforce page in order to pass selected ids from the tab to your flow. This requires some set up on your side before you can create a flow to work with the records selected.

1. Decide which object you want to pass Ids from, and modify the Apex Class below and save it in your Org. The class included with the package only works with Accounts.

- Change the highlighted text to the object's API name you require and save it as a new Apex Class

```
public class PassSelectedIdsToFlowVFController {

    public string[] SelectedListRecordIDs{get;set;}

    public PassSelectedIdsToFlowVFController(ApexPages.StandardSetController
listcontroller){
        SelectedListRecordIDs = new string[]{};
        for(Account chl : (Account[])listcontroller.getSelected()){
            SelectedListRecordIDs.add(chl.Id);
        }
    }
}
```

- With the test class, you need to change the name highlighted in blue to match the name of the Apex Class you just saved

```
@isTest
public class PassSelectedIdsToFlowVFControllerTest {
    public static testmethod void PassSelectedIdsTest(){
        List <Account> checklistLBList = new List<Account>();
        // insert Account
        Account checklistLB = new Account(name='TestAccount');
        insert checklistLB;
        checklistLBList.add(checklistLB);
        ApexPages.StandardSetController sc = new
ApexPages.StandardSetController(checklistLBList);
        sc.setSelected(checklistLBList);
        PassSelectedIdsToFlowVFController testPassSelectedIds = new
PassSelectedIdsToFlowVFController(sc);
    }
}
```

2. Inside the flow, you will need to decide what you want to happen while looping thru the records. In this flow template, it is updating the billing street address based on the text from the screen that appears. This is purely for use with the example. If you decide to build a new flow, be sure to make the variable for the ids to passed into a **text** variable.

3. You will need a Visualforce page that can be tied to your button.

- Change the name after the standardController to match the object you are working on.
- Make sure the extension are is the same name as your Apex class you saved from above
- After flow interview name, input the name of your flow
- If you have changed the name of the text variable in your flow from SelectedTabRecords, then input that name

```
<apex:page standardController="Account" recordSetVar="chl"
extensions="PassSelectedIdsToFlowVFController"
lightningStylesheets="true">

    <flow:interview name="Pass_Records_From_A_Tab">
        <apex:param name="SelcetcedTabRecords" value="{!
SelectedListRecordIDs}" />
    </flow:interview>

</apex:page>
```

4. Now just create a button on your object, direct it to the above visualforce page, and expose it on the list view.

Helpful links for Flows

Below is a set of helpful links to resources to continue your flow education.

https://automationchampion.com/learning-flow/

https://unofficialsf.com/

Process Builder App