CloudCrossing BVBA Dahlialaan 1 2950 Kapellen Belgium



## FLOW INTRODUCED VARIABLES

## Flow Introduced variables

It can be that the input fields of a Screen Flow would be used as variables in the DataSource queries.

This is possible via Flow Introduced variables.

Let's take a flow as an example.

We want to output all opportunities with a close date in a certain period.

bex Defined Variable Test 2	
Start Date	
	â
End Date	
	i i

## The SOQL we want to run to select the data:

Data Source Name	Record Type	
Opportunities for Account	SOQL	
Description	Type  Type	
iOQL Builder Managed 🚯		
Child Data Source Settings		
Parent Data Source		
Grouping Field Name 🕕		
arent Query Field Name 🚯		
Query Settings		

This SOQL takes 2 parameters, the ":startdate" and the ":enddate". These parameters have to be set by the Flow.

Just as the default parameter ":recordId", the parameters must be identified by ":".

The flow:



Screen 1 will select the dates:

	Edit Scree	'n		
Screen Components			Screen Properties	2
kearch components	Apex Defined Variable Test 2		*Label	
			enter vour data	
<ul> <li>Input (23)</li> </ul>	* Start Date			
Address		t	* API Name	
Checkbox			enter_your_data	
Checkbox Group	* End Date		Description	
Currency		ŧ	1	
ate Date				
🙃 Date & Time	Pause	Previous		
Dependent Picklists			Ne Configura Frama	
Display Image			♥ Conligure Hame	•
🛃 Email			Show Header	
🚱 File Upload			Show Footer	
Long Text Area			> Control Navigation	
🚱 Lookup				
Multi-Select Picklist			> Provide Help	
Name				
Get more on the AppExchange				

These are Date fields that allow to select a date, in this case without a time.

Now we want to add these as parameters. So for every parameter, we must create a variable in Flow. The variable must be of type Apex-Defined and set the class cadmus\_core\_\_CadmuskeyValue.

Ner	v Resource	
*Resource Type		
Variable		•
* API Name		
startdate		
Description		
* Data Type		
Apex-Defined	<ul> <li>Allow multiple values (collection)</li> </ul>	
*Apex Class 🚺		
cadmus_coreCadmusKeyValue		
	Cancel Don	e

Do the same for the "enddate" parameter.

These parameters have to wrapped in first a collection and second in a parameter variable to pass on to the "Call PDF Butler" component.

The Collection variable:

See the type, but more important, indicate this is a collection.

	Edit Variable	
* API Name values		
Description		
* Data Type Apex-Defined	Allow multiple values (collection)	
*Apex Class  Cadmus_coreCadmusKeyValue		
		Cancel Done

Next, the variable that wraps the collection:

This one has to be of type: "cadmus\_core\_\_CadmusParameters"

Edit \	/ariable
* API Name	
Description	
*Data Type 🔹	
Apex-Defined * Apex Class Cadmus_coreCadmusParameters	
	Cancel Done

Assign the values to the variables:

Important to note, the CadmusKeyValue has to be set:

- 1) key: unique identifier and also maps on the parameter for the SOQL
- 2) value: this has to be specific for the type of the variable
  - a. valueString
  - b. valueDouble
  - c. valueDate
  - d. valueDateTime
  - e. valueBoolean

	Edit Assi	gnment	
*Label		* API Name	
set vars 1		set_vars_1	
Description			
			10
Set Variable Values			
Each variable is modified by the operator and value	combination.		
Variable	Operator	Value	
{!startdate.key}	Equals	▼ startdate	<b></b>
Variable	Operator	Value	
{!startdate.valueDate}	Equals	▼ {!Start_Date}	俞
Variable	Operator	Value	
{!enddate.key}	Equals	▼ enddate	亩
Variable	Operator	Value	
{!enddate.valueDate}	Equals	▼ {!End_Date}	亩
L Add Assignment			
			Cancel Done

Assign even more variables to fill the wrappers:

- add the variables to the collection
- set the collection to the "values" property

	Edit Ass	signment		
*Label		* API Name		
set vars 2		set_vars_	2	
Description				
				10
Set Variable Values				
Each variable is modified by the operator and value combination.				
Variable	Onevates		Value	
{Ivalues}	Operator Add	•	(Istartdate)	ŵ
Variable	Operator		Value	_
{!values}	Add	•	{!enddate}	
Variable	Operator		Value	
{!parameters.values}	Equals	•	{!values}	Ô
+ Add Assignment				
			Cancel Done	е

## Call PDF Butler

Add an action of type "Call PDF Butler"

Set at least following input values (these are hard coded for demo purposes).

Most important one is when the {!parameters} are set. This is the wrapper variable we set earlier.

	Edit "Call PDF Butler" core action	
Use values from earlier in the flow to set the inputs for the	"Call PDF Butler" core action. To use its outputs later in the flow, store them in variables.	*
*Label	* API Name	- 1
Call PDF Butler	Call_PDF_Butler	
Description		
		10
Set Input Values Store Output Values		- 1
A <sub>3</sub> Alternative Name	Don't Include	le
A <sub>a</sub> Country	Don't Include	le
A <sub>a</sub> Doc Config Id	$\sim$	)
a0J1v00000CysmvEAB	Include	
A <sub>a</sub> Doc Config Id As String	Don't Include	le
A <sub>a</sub> Language	Don't Include	le
A <sub>a</sub> Pack Id As String	Don't Include	le
A <sub>a</sub> PDF Action Type	Don't Include	le
A <sub>a</sub> Record Id		)
0060Y00000IhdX4QAJ	Include	
Aa Record Id As String	Dont Include	le
00 Run Async	Dont Include	ie
>_ Set of parameters that can be used in PDF Butler DataSources		)
{!parameters}		
A <sub>a</sub> Target Type	Don't Include	le
A <sub>a</sub> Timezone	Don't Include	le
		-