

CloudCrossing BVBA
Dahlialaan 1
2950 Kapellen
Belgium



BULK BUTLER

BULK Butler

With the PDF Butler Batch add-on it is easy to schedule and start batch jobs.

From the interface you will get a great overview on all runs of the batch.

You can also start or schedule the batch via the buttons on the page

The screenshot shows the PDF Butler Batch Info interface. At the top, there is a search bar and navigation tabs including 'PDF Butler', 'Get Started', 'Data Sources', 'Doc Configs', 'Pdf Butler Admin', 'PDF Butler Packs', and 'Batch Infos'. The main section is titled 'Batch Info' and 'Create Opportunity Overview For All Accounts'. It features a 'Batch Performance Report' with a horizontal bar chart showing 'Record Count' on the x-axis (0 to 3) and 'Start Time' on the y-axis (ranging from 3/03/2019 to 17/09/2019). A legend on the right indicates 'Processed' counts: 12 (blue), 44 (black), 18 (green), 24 (teal), and 22 (yellow). Two blue arrows point to the 'Schedule batch' and 'Run Batch (1 time)' buttons in the top right corner. Below the chart, there are sections for 'Related Details' and 'Batch Runs (3+)'. The 'Batch Runs' section lists two runs: BR-0083 (Start Time: 9/11/2019 10:10, End Time: 9/11/2019 10:10, Time Elapsed: 0.13) and BR-0082 (Start Time: 8/11/2019 10:10, End Time: 8/11/2019 10:10, Time Elapsed: 0.13).

A Batch Info record:

The detailed view of a Batch Info record shows the following information:

- Batch Name:** Create Opportunity Overview For All Accounts
- Owner:** Igor Stuyver
- Cron:** 0 10 10 * * ?
- Emails:** igor.stuyver@pdfbutler.com
- SOQL:** SELECT Id, Name FROM Account
- Doc Config:** Account Opportunities
- Batch Size:** 5
- Batch Apex Class:**
- Count SOQL:** SELECT Count() FROM Account
- Delivery Option:** ATTACHMENTS_OVERWRITE
- Created By:** Igor Stuyver, 27/02/2019 20:28
- Last Modified By:** Igor Stuyver, 9/11/2019 15:33

Information on all runs of the batch can be found related to the Batch Info:

A Batch Run record:

A Batch Run will supply the stats and give an overview of any errors occurred:

The Batch errors will be created for every record in error. An example:

Create a BatchInfo:

- Name
- Cron: when scheduling this will indicate when to run. For a single launch, this is not important

The following are some examples of how to use the expression.

| Expression | Description |
|--------------------|---|
| 0 0 13 * * ? | Class runs every day at 1 PM. |
| 0 0 22 ? * 6L | Class runs the last Friday of every month at 10 PM. |
| 0 0 10 ? * MON-FRI | Class runs Monday through Friday at 10 AM. |
| 0 0 20 * * ? 2010 | Class runs every day at 8 PM during the year 2010. |

- Emails: who to notify when batch has ran. Overview and status of batch and possible errors will be mailed. Leave empty when there is nobody to notify
- SOQL: this is the input data for the batch. This query will select all records that are the starting point of the context, data selected by the DataSources.
- DocConfig: DocConfig to run for every record returned from the SOQL
- Batch Size: Salesforce will not run all records returned by the SOQL in 1 go. For performance and governor limits the batch will be split in small parts. It is recommended to set this to small chunks so that the limits in queries, from the DocConfigs related DataSources, are not hit and that the transaction of inserting the generated documents remains small. It is not set that a bigger batch size will run faster than smaller.
We recommend small batch size between 5 and 10.
- Batch Apex Class: An APEX class that implements “cadmus_batch.Batch_ICadmusBatch”. See further for more information
- Count SOQL: this is normally the same query as in the SOQL field but not selecting and fields but counting the number of records eg: “SELECT Count() FROM Account”
Make sure that this is a count-query. The Count() function cannot have any parameters.
With bigger batch jobs, you can follow up on the status and see how many records to process and how many are already done.
- Delivery Option: The Delivery Option from the DocConfig will not be used. Indicate how the generated documents are saved related to the record.
 - o ATTACHMENTS
 - o ATTACHMENTS_OVERWRITE
 - o FILES
 - o BASE64

Extending BULK Butler

cadmus_batch.Batch_ICadmusBatch

As everything PDF Butler, the batch is also extensible. We cannot cover all possible use-cases so we let you extend so it does exactly what you need.

```
global class YourActionBatch extends cadmus_batch.Batch_ICadmusBatch {
    global Batch_NoActionBatch() {}
    global override void beginExecute(List<sObject> items){
        //Your implementation here
    }
}
```

```

        //This will be executed when the chunk (batch part according to "Batch Size" setting) is
        started. The list is the sObjects the SOQL retrieved. To handle these, just cast them to the type you
        queried via the SOQL eg: Account acc = (Account)items.get(0);
    }

    global override void inExecuteLoop(sObject item, cadmus_core.ConvertController.ConvertDataModel
data){
        //Your implementation here
        //This will be executed when running through the list of items processed in this batch chunk.
        For every record processed, this will be executed.
        //In this part you can set extra information on the ConvertDataModel, eg the Alternative to use:
        data.alternativeName = "<YOUR ALTERNATIVE>"
        // To handle the item, just cast them to the type you queried via the SOQL eg: Account acc =
        (Account)items;
    }

    global override void endExecute(List<sObject> items,
List<cadmus_batch.Batch_ProcessDocConfigs.DocGenResultData> results){
        //Your implementation here
        //This will be executed when all records in the chunk (batch part according to "Batch Size"
        setting) are processed. At this time you can decide what to do with the documents that are generated. Eg
        save to another record in SFDC, email, ...
    }

    global override void beginFinish(){
        //Your implementation here
        //the batch is finished, you might want to initialize another task, inform users, connect to a
        backend system
    }

    global override void endFinish(){
        //Your implementation here
        //the batch is finished and really at the end you want to take an action, you might want to
        initialize another task, inform users, connect to a backend system.
        //information on the batch and the status can be retrieved from the Batch Run sObject
    }
}

```

Anywhere in this class, you can get information from the current batchInfo or batchRun like this:

In the below, we get the Pack Id from the batchInfo:

```
this.batchInfo.cadmus_batch__Pack__c
```

In the below, we get the Pack Id from the batchRun:

```
this.batchRun.cadmus_batch__Docs_To_Process__c
```

Code example that update a field on the records processed:

```
global class YourBatchClass extends cadmus_batch.Batch_ICadmusBatch
{
    global override void beginExecute(List<sObject> items) {}
    global override void inExecuteLoop(sObject item,
cadmus_core.ConvertController.ConvertDataModel data) {}

    global override void endExecute(List<sObject> items,
List<cadmus_batch.Batch_ProcessDocConfigs.DocGenResultData> results)
    {
        List<Opportunity> opps = new List<Opportunity>();
        for(cadmus_batch.Batch_ProcessDocConfigs.DocGenResultData res : results)
        {

            Opportunity opp = (Opportunity) res.item;
            opp.Batch_Process__c=true;

            opps.add(opp);

        }
        update opps;
    }
    global override void beginFinish() {}
    global override void endFinish() {}
}
```

cadmus_batch.Batch_ProcessController

You might want to call your batch programmatically. This can be done by directly calling into BULK Butler key controller class.




Methods:

- `startBatch(Id batchInfoId)`: start a batch immediately and run it 1 time. The supplied `batchInfoId` is the Id of the `BatchInfo` record created that holds all information the batch requires to run.
- `schedulebatchMethod(Id batchInfoId)`: schedule a batch according to the Cron job settings in the `BatchInfo` record. The supplied `batchInfoId` is the Id of the `BatchInfo` record created that holds all information the batch requires to run.

Run Pack with BULK Butler

You can also run Packs with BULK Butler, for instance if you want to create multiple documents in 1 go or add some Pre or Post processing eg sent out via email (see that the Batch Size is set to 1).

Make sure the Pack field is on the Page Layout and just pass on a Pack instead of a DocConfig:

| Related | Details |
|--|--|
| Batch Name | Owner |
| test validation |  Igor Stuyver |
| Cron | |
| NA | |
| Emails ¹ | |
| igor.stuyver@pdfbutler.com | |
| SOQL | |
| SELECT Id FROM Account LIMIT 5 | |
| Doc Config | |
| Pack ¹ | |
| Sent Mail | |
| Batch Size | |
| 1 | |
| Batch Apex Class | |
| Count SOQL ¹ | |
| SELECT Count() FROM Account | |
| Delivery Option | |
| FILES | |
| Created By | Last Modified By |
|  Igor Stuyver , 9/11/2019 20:28 |  Igor Stuyver , 10/11/2019 0:03 |

Starting a batch from a flow:

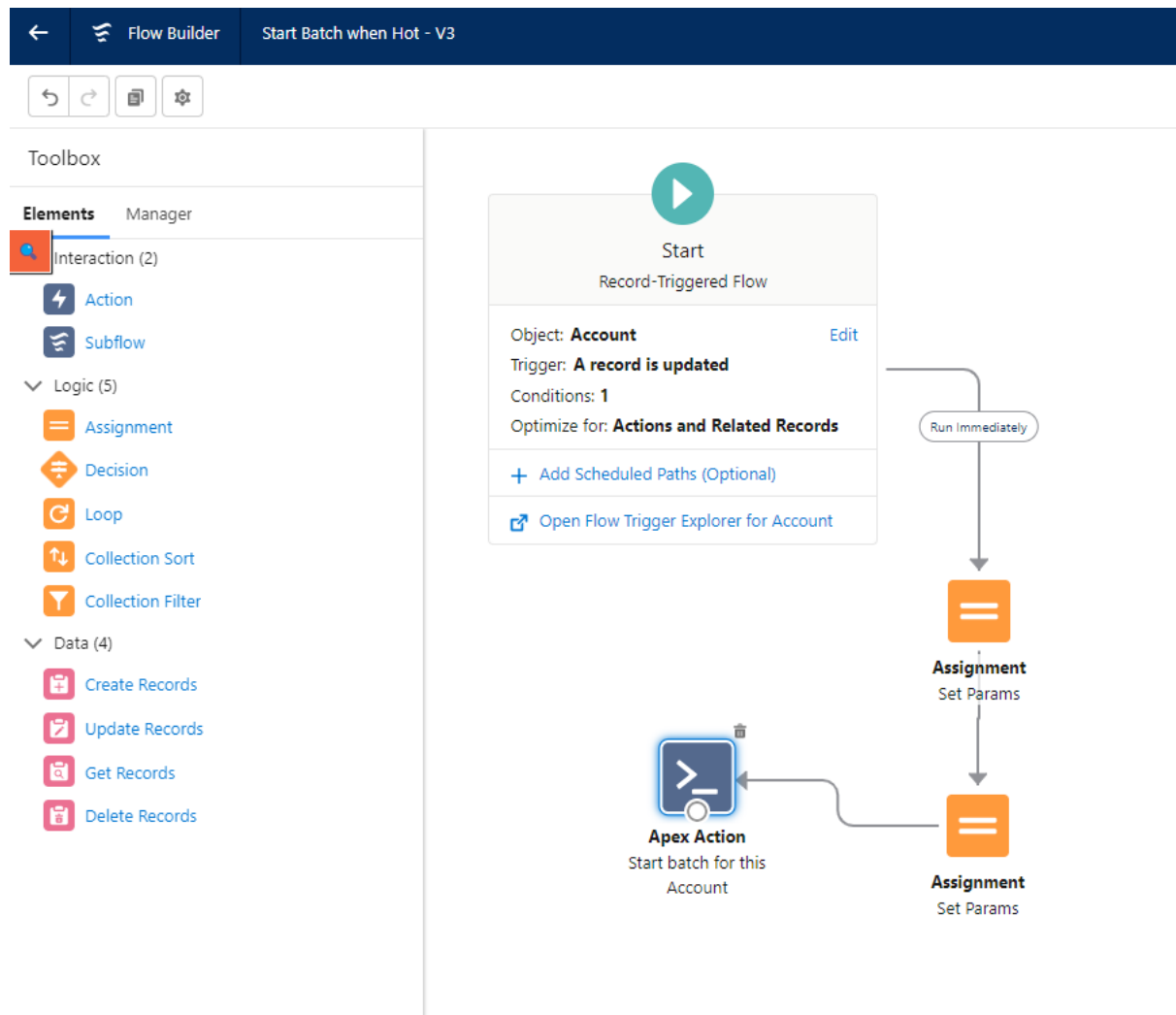
When you need to process multiple records when a certain event occurs, you can launch BULK Butler to run a batch.

With parameters

You want to launch the batch from a flow but want to filter the records processed in the batch in relation to the record you are working from or a list of records.

This example can be done from any flow but below we will use a record triggered flow.

See the flow below:




The flow uses parameters to pass on and sets these in the Assignment steps




More information on how to add these to the DataSource can be found here:


<https://www.pdfbutler.com/academy.html?q=Flow%20Introduced%20variables>

- 1) set the parameter value
in this case we set the a parameter for the AccountId
create the variables in the Flow:

Edit Variable



KeyValue 

* Data Type 
Apex-Defined  Allow multiple values (collection) 


* Apex Class 
cadmus_core__CadmusKeyValue




Availability Outside the Flow


Available for input
 Available for output

Edit Variable



Parameters 

* Data Type 
Apex-Defined  Allow multiple values (collection) 

* Apex Class 
cadmus_core__CadmusParameters

Availability Outside the Flow


Available for input
 Available for output

2) Manage the assignments:









- a. Assignments of values: Here we assign the
 - i. Key: this is the name of the parameter as it will be used in the SOQL for the Batch Info record


Edit Assignment



Set Params (Set_Params) 

Set Variable Values

Each variable is modified by the operator and value combination.


| Variable | Operator | Value | |
|---|----------|---|---|
|  Key KeyValue > key  | Equals | accid |  |
|  Value KeyValue > valueString  | Equals |  Record \$Record > Account ID  |  |



- b. Add all variables to the Parameters variable. This implies that you can introduce multiple variables to be passed on!

Edit Assignment

Set Params (Set_Params2) 

Set Variable Values

Each variable is modified by the operator and value combination.

| Variable | Operator | Value |
|---|----------------------------------|---------------------------------------|
| <input type="text" value="Parameters > values"/> | <input type="text" value="Add"/> | <input type="text" value="KeyValue"/> |

3) add a Flow Action:


New Action

Filter By

- All
- Users
- Group
- Sales leads
- Task
- Price books
- Feed Item
- Chatbots
- Case
- Account
- Contact

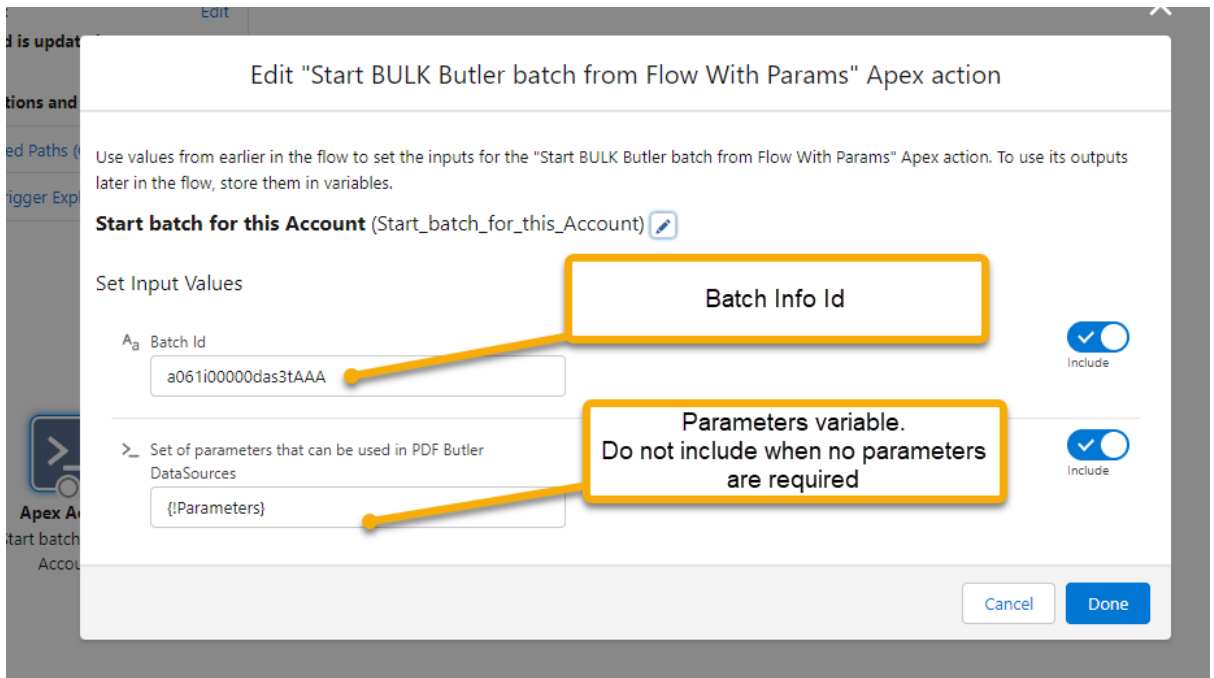
Action

- Start **BULK** Butler batch
apex-cadmus_batch__Batch_StartBatchInvocable
- Start **BULK** Butler batch from Flow With Params
apex-cadmus_batch__Batch_StartBatchWithParamsInvocable



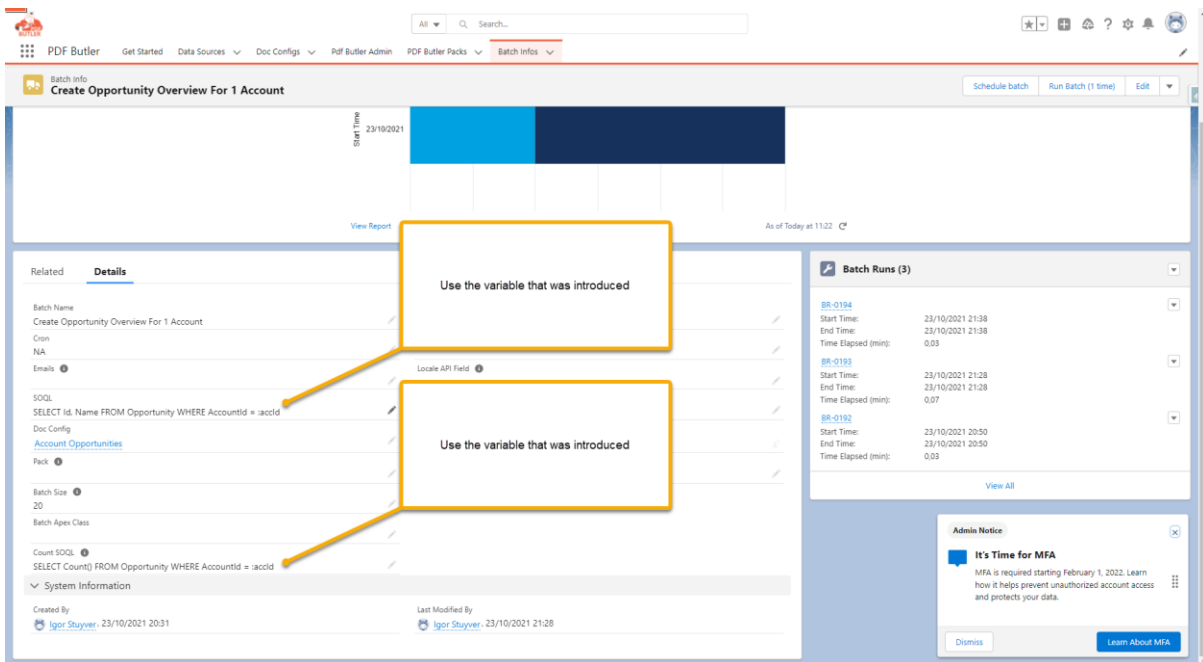
Lights, camera, action!
Select an action to configure.

And configure it



Next is to make sure our Batch Info SOQL uses this variable:

The variable that we used was called "accId", in the SOQL we must use it by prefixing it with a ":". So use this with ":accId" in the SOQL



Using a List of records as variable

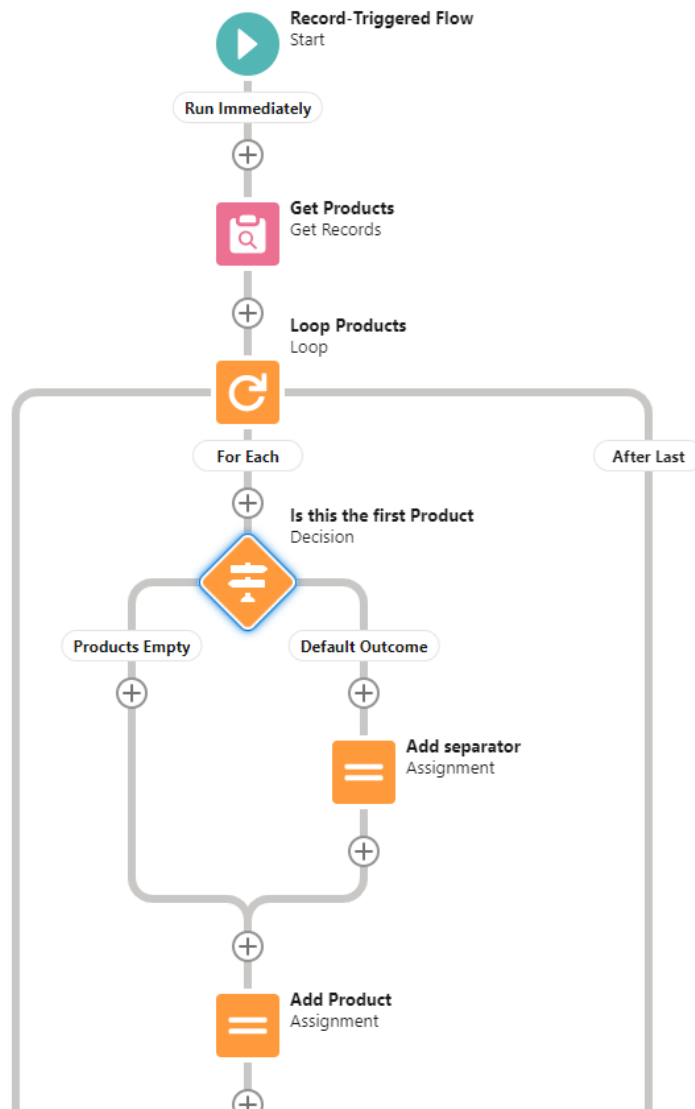
The idea is not to add thousands of records to the variable. When you want to process a large amount of records. The SOQL should select these via a stage, status, checkbox, ...

This is for passing a limited number, let's say 100 records, to the batch for processing.

The variable must look like this: "a061i00000das3tAAA; a061i00000das3tAAA; a061i00000das3tAAA; a061i00000das3tAAA"

So the values must be split by "; ". Do not forget to add the space after the semicolon.

In Flow, you can use it like this to set it up:




When looping over the products, every product Id is added to the value of the variable.


When this is the first product, do not add a separator, otherwise ... add it.

- Decision

Edit Decision

Is this the first Product (Is_this_the_first_Product) 

Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

OUTCOME ORDER 

Products Empty

Default Outcome

OUTCOME DETAILS

*Label: Products Empty

*Outcome API Name: Products_Empty

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource: productIds > valueString

Operator: Is Null

Value: True

+ Add Condition

When to Execute Outcome


If the condition requirements are met

Only if the record that triggered the flow to run is updated to meet the condition requirements

Cancel Done

- Add separator

Edit Assignment

Add separator (Add_separator) 

Set Variable Values

Each variable is modified by the operator and value combination.

Variable: productIds > valueString

Operator: Add

Value: ;


+ Add Assignment

Semicolon and a space

Cancel Done

- Add product Id field

Edit Assignment

Add Product (Add_Product) 

Set Variable Values

Each variable is modified by the operator and value combination.

Variable: productIds > valueString

Operator: Add

Value: Current Item from Loop Loop_Products...

+ Add Assignment

Cancel Done