

CloudCrossing BVBA
Sint-Pietersvliet 7
2000 Antwerpen
Belgium



APEX Actionables - KEYVALUE

APEX Actionables:

New Data Source

Select a record type

- KEYVALUE
- PICKLIST TRANSLATION
- PICTURE LIST
Type for adding sending dynamic pictures to PDF Butler
- SOQL
- STATIC VALUES

Cancel Next

New Data Source: KEYVALUE

General

*Data Source Name Record Type KEYVALUE

Description Type

Child Data Source Settings

Parent Data Source

Grouping Field Name

Parent Query Field Name

Cancel Save & New Save

✕

New Data Source: KEYVALUE

General

*** Data Source Name**

Record Type

KEYVALUE

Description

Type ⓘ

Single sObject ▾

Child Data Source Settings

Parent Data Source

🔍

Grouping Field Name ⓘ

Parent Query Field Name ⓘ

Cancel

Save & New

Save

Data Source

SummaryTotals

Related List Quick Links ⓘ

DocConfig

Datasources (0)

General

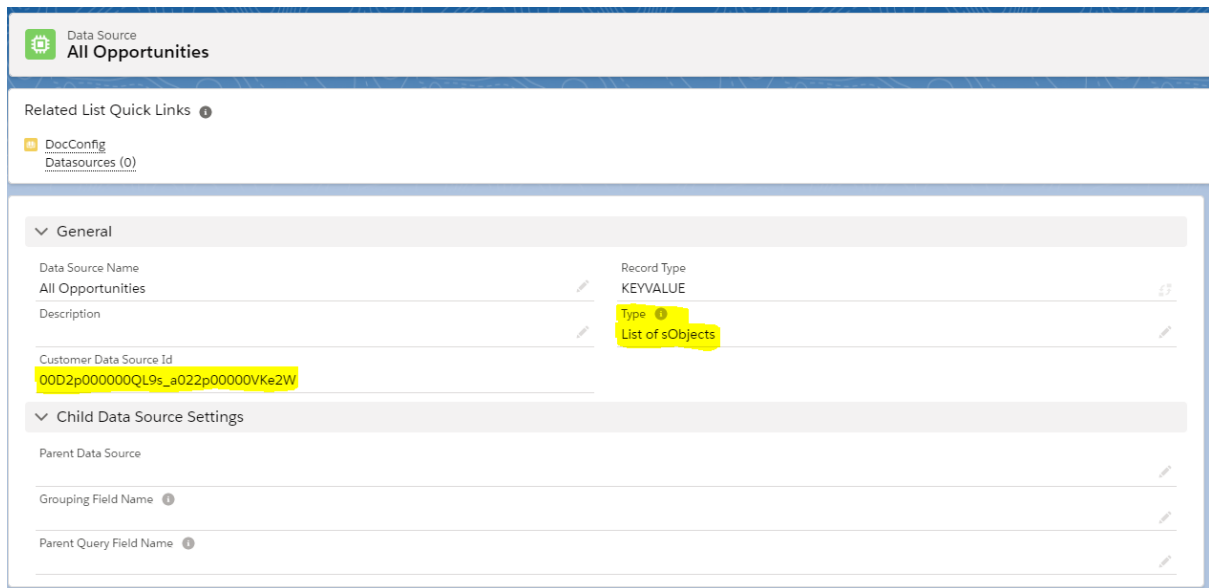
Data Source Name	Record Type
SummaryTotals ✎	KEYVALUE 🔄
Description	Type ⓘ
Customer Data Source Id 00D2p000000QL9s_a022p00000VKe2R	Single sObject ✎

Child Data Source Settings

Parent Data Source ✎

Grouping Field Name ⓘ ✎

Parent Query Field Name ⓘ ✎



These are actions that can run before or after the Document is generated via PDF Butler.

You can easily implement your own Actionables in APEX by implementing the following interfaces:

- cadmus_core.AbstractBeforeActionable
- cadmus_core.AbstractAfterActionable

Make sure your class is “global”!

AbstractBeforeActionable:

The class has 1 method to implement:

```
global class Actionable_CalculateStuff implements cadmus_core.AbstractBeforeActionable {
    global void execute(cadmus_core__Actionable__c actionable,
        Id docConfig,
        Id objectId,
        Map<String, Object> inputMap,
        cadmus_core.ConvertController.ConvertDataModel cdm) {
        //your custom stuff
    }
}
```

The most custom usage for the Before actionable is to add data to the inputMap. This data can be used for variables in SOQL Datasources or to be directly added to the PDF Butler.

- 1) Variables: just add a key/value to the Map. If the key matches a variable in a SOQL, then it will be used
- 2) Single DataSource: the data must always map to a DataSource in the PDF Butler App
Create a new KEYVALUE DataSource:

New Data Source

Select a record type

- KEYVALUE
- PICKLIST TRANSLATION
- PICTURE LIST
Type for adding sending dynamic pictures to PDF Butler
- SOQL
- STATIC VALUES

Cancel Next

Make sure that the DataSource is of Type “Single sObject”

▼ General

Data Source Name	My Single Key Value DataSource	Record Type	KEYVALUE
Description		Type	Single sObject
Customer Data Source Id	00D1t00000FRu9_a0Y1t000001npJi		

The DataSource will get an unique identifier to identify it over all environments. This key will be required to add the data to the inputMap

Here some example code to add the data for this DataSource

```
//Get DataSources
List<cadmus_core__Data_Source__c> dss = [SELECT Id, Name,
cadmus_core__CustomerDataSourceId__c,cadmus_core__Image_Name__c From
cadmus_core__Data_Source__c];

Map<String, cadmus_core__Data_Source__c> dsMap = new Map<String,
cadmus_core__Data_Source__c>();
for(cadmus_core__Data_Source__c ds : dss) {
    dsMap.put(ds.cadmus_core__CustomerDataSourceId__c, ds);
}

//DUMMY 1 ROW: shows haw to add text values
Map<String, String> dummy = new Map<String, String>();
dummy.put('test','test');
dummy.put('test2','test2');
inputMap.put(dsMap.get('00D1t00000FRu9_a0Y1t000001npJi').Id, dummy);

//Calculated fields: Shows how to add numbers
Map<String, Decimal> SummaryTotals = new Map<String, Decimal>();
SummaryTotals.put('NO_VAT', 12);
SummaryTotals.put('WITH_VAT', 12 * 0.21);
inputMap.put(dsMap.get('00D2p000000QL9s_a022p00000VKe2R').Id,
SummaryTotals);
```

- 3) List DataSource: the data must always map to a DataSource in the PDF Butler App Create a new KEYVALUE DataSource:
Make sure that the DataSource is of Type “List of sObject”

General	
Data Source Name	My List Key Value DataSource
Record Type	KEYVALUE
Description	Type List of sObjects
Customer Data Source Id	00D1t000000FRu9_a0Y1t000001npJi

Here some example code to add the data for this DataSource

```
//Get DataSources
List<cadmus_core__Data_Source__c> dss = [SELECT Id, Name,
cadmus_core__CustomerDataSourceId__c,cadmus_core__Image_Name__c From
cadmus_core__Data_Source__c];

Map<String, cadmus_core__Data_Source__c> dsMap = new Map<String,
cadmus_core__Data_Source__c>();
for(cadmus_core__Data_Source__c ds : dss) {
    dsMap.put(ds.cadmus_core__CustomerDataSourceId__c, ds);
}

//LIST
<SELECT YOUR DATA TO ADD> eg variable accnts
//create the list of maps
List<Map<String, String>> myMapsList = new List<Map<String, String>>();

for(Account acc : accnts) {
    Map<String, String> tempAccnts = new Map<String, String>();
    tempAccnts.put('Id',acc.Id);
    tempAccnts.put('Name',acc.Name);
    myMapsList.add(tempAccnts);
}
inputMap.put(dsMap.get('00D1t000000FRu9_a0Y1t000001npJi').Id, myMapsList);
```

AbstractAfterActionable

The most custom usage for the After actionable is to handle the documents. The documents are available via the wrapper.

This way, you can store the documents where you want if you do not want to use the OOTB storage capabilities, eg store the document linked to another object then the one that was used to initiate the PDF Butler call.

You can also integrate here with your own internal or external applications or define actions.

The documentation on the ConvertDataModel and DocGenerationWrapper can be found in "Call PDF Butler from APEX"

The class has 1 method to implement:

```
global class Actionable_AfterStuff implements cadmus_core.AbstractAfterActionable {
```

```
global void execute(cadmus_core__Actionable__c actionable,
    Id docConfig,
    Id objectId,
    Map<String, Object> inputMap,
    cadmus_core.ConvertController.ConvertDataModel cdm,
    cadmus_core.DocGenerationWrapper wrapper) {
    //your custom stuff
}
}
```

Add your KeyValue DataSource to the DocConfig.

In the DocConfig configuration page, you can add your KeyValue DataSource just like any other DataSource. We ofcourse, do not know the fields that you will add via APEX.

After adding, you will see a screen like this one:

For every field, just click 'Add field' to make sure you can use it. Make sure to also set the type correctly for correct formatting:

DataSource Reload from SFDC

Name
SummaryTotals

Unique identifier
00D2p000000QL9s_a022p00000VKe2R

Type
Single

Fields
Make sure to update the field types accordingly and validate that the fields in SFDC are copied correctly!

[Add field](#)

Name	Type
<input type="text" value="NO_VAT"/>	<input type="text" value="CURRENCY"/>
<input type="text" value="WITH_VAT"/>	<input type="text" value="CURRENCY"/>

[OK](#) [Cancel](#)

Use your actionable

Easiest way is to use it from the DocConfig.

Add a new Actionable via the Related list on DocConfig (if the list is not there, just update the Page Layout to add it)

Doc Config
Mail - Contact Lead Generation + Follow Edit

Related List Quick Links

- DocConfig Datasources (2)
- Actionables (0)**

Related	Details
Document Config Name Mail - Contact Lead Generation	Owner Igor Stuyver
Description	Record Type EMAIL
Document Title Contact Lead Generation	Delivery option BASE64
System Information	
Created By Igor Stuyver, 24/07/2019 15:20	Last Modified By Igor Stuyver, 11/08/2019 10:57
Custom Links	
Open PDF Butler	

The Actionable is of type "Run Class":

New Actionable

Select a record type

- DocuSign
Sent document via DocuSign
- AdobeSign
Sent document via AdobeSign
- Auto Email
Sents out your documents created by PDF Butler via mail. Uses a SFDC Template
- Email DocConfig
This actionable should be used when the DocConfig is of type EMAIL. This action will sent out the resulting email.
- Run Class
Run a class that is inherits AbstractAfterActionable or AbstractBeforeActionable

Cancel

Next

Next is to fill in the details, make sure your Actionable is Active and it will be run Before or After the document is generated by PDF Butler:

New Actionable: Run Class

Information

* Actionable Name

My Before Actionable

Record Type

Run Class

Doc Config

Mail - Contact Lead Generation

Owner

Igor Stuyver

Class

MyBeforeActionable

* When

BEFORE

Active



Pack

Search PDF Butler Packs...

Cancel

Save & New

Save

An alternative way is to use it with a PDF Butler Pack.