Asynchronous Process Manager / Creator

About the Application

1.Post Installation Steps
- 1.1 Schedule background processes

2.Async Job System
- 2.1 Manage Async Job Parameters
- 2.2 Start/Stop Async Job Service
- 2.3 Monitor Async Jobs
- 2.4 Run Async Job Immediate
- 2.5 Abort Async Job

3. Adding Jobs Into the Queue Using Custom Code
- 3.1 Use Async Job with Custom Code

4.Async Job Template
- 4.1 Define Async Job Template
- 4.2 Run Async Job Template
- 4.3 Export/Import Async Job Template

5.Other Services
- 5.1 Database Service Class
- 5.2 Trigger Framework
  - 5.2.1 Create new trigger
  - 5.2.2 Add Custom Process to Run by the Trigger Framework

6.Additional Components
- 6.1 How to Use Filter Data Component
- 6.2 How to Use Field Setup Component

About the Application

The application provides base services for managing background processing and some advanced services for adding background processes using declarative tools.

Main capabilities:

- Engine for running/managing background processes
- Tool for building process templates

Additional capabilities, which exposed for usage

- Service for managing DML transaction, including checking FLS
- Utility for building filter criteria per object/fields
- Trigger framework for custom processes

1.Post Installation Steps

1.1 Schedule background processes

1.Go to tab **MBA Settings**
2.Click button **Start Scheduler**

- This action will schedule a background scheduler to run jobs that enter the queue, note that the user who is running the jobs will be the user who starts the scheduler, therefore please make sure that this user has the right permissions.

1.2 Set your support email

(Optional) Set email address that will be used to report application errors

1.Navigate to Salesforce Setup-> Custom Settings
2.Click button **Manage** next to **Org Settings**
3.Click button **Edit**
4.Set the address in the field **Support Email** and click **Save**

2.Async Job System

The Async Job System allows the application to run background processes that sometimes require heavy calculation and therefore cannot run in synchronous mode.
This section describes how to view and manage the activities of the Async Job System.

2.1 Manage Async Job Parameters

- It is advised to keep this setting with their default values. If you still need to change any of the parameters, please read this section and understand its impact.
  1.Go to tab **MBA Settings**
  2.Setup the following parameters per your need
  - **Run Method** - can be either **Smart** or **Fixed Interval**. In the latest you will also need to set also the **Interval (M)** and this will define the minutes interval for the scheduler job, in smart run the system will reschedule itself and set its own interval based on estimation and report of the latest processes. If choosing **Smart** run you should also set the **Maximum Interval (M)** which is the maximum delay in minutes between runs and the **Tolerance (M)** which is the acceptance minutes that a new async process can wait or it may run immediately without the scheduler. For example, if the scheduler will run again in 10 minutes and the Tolerance is 5 minutes, then adding a new job will run it immediately.
  - **Inactivity Warning (M)** - set the amount of time (in minutes) without inactivity that the system will assume that the scheduler is not running and alert.
  - **Maximum Batch Jobs in Parallel** - The maximum batch jobs in Salesforce that can be run in parallel is 5. In this input you can set up the maximum batch jobs that Async Job System may occupy. It is advice to keep the number lower than 5. This way the Async Job System won't occupy all the available batches.
  - **Maximum Future Jobs Per Run** - every time the scheduler will run it will run future jobs up to this input. Maximum number of futures allowed in Salesforce per transaction is 50.
  - Next 5 inputs can be used to control the priority that the Async Job System processes its jobs. Each job has Priority input (low/medium/high/urgent) and the creation date. In those inputs you can provide scale numbers for each type to control the order jobs are processed. For example, providing high value for the **Create Date Scale** will give a higher impact for the creation date. This way, a low priority job can be executed before a high priority job, if it was created earlier.
    Low Priority Scale
    Medium Priority Scale
    High Priority Scale
    Urgent Priority Scale
    Creation Date Scale
  3.Click button **Save**

2.2 Start/Stop Async Job Service

1.Go to tab **MBA Settings**

2.In section **Async Scheduler Settings** use the buttons **Start Scheduler** or **Stop Scheduler**. Only one of these buttons should be available, depending on the current state.

2.3 Monitor Async Jobs

1.Go to tab **Async Job**

2.Select specific list view, for example **All** or **Queued**.

- You can define additional list views.

3.Go to specific Async Job record by clicking the Async Job Number

- If you are system administrator you might have privileges to update data in the Async Job record. Such updates should be avoided, as it is possible that the Async Job System is processing this data.

2.4 Run Async Job Immediate

1.Go to a specific Async Job record in status Queued.

2.Click button **Run Now**.

3.You will either see a message that the job is now being processed or a message indicating that the Async Scheduler is about to run and you should wait until it will do the job.

2.5 Abort Async Job

1.Go to a specific Async Job record in status Queued.

2.Click button **Abort Async Job**

3.You will either see a message that the job was aborted or a message indicating that the job is already in progress and therefore cannot be aborted.

3. Adding Jobs Into the Queue Using Custom Code

There are 2 options for adding jobs into the code:

- Writing custom code
- Using Async Job Template

This section describes the first option.

3.1 Use Async Job with Custom Code

- Note that this module requires development skills in Salesforce.

In order to run your custom code with the system, write an apex class that implements the interface **IRunAsyncJob**. In this interface you will need to implement the **run** method which will be invoked by the async job engine.

- Note that the class must be **global**

Example: develop a job that receives Lead Id as parameter, search if the lead already exists in the system as contact. If yes it will send a message to the submitter, otherwise it will automatically convert the lead.

```
//************************************************************
//    Convert Lead in Apex
//
//  search if the lead already exists in the system as contact.
//  If yes it will send a message to the submitter,
//  otherwise it will automatically convert the lead.
//************************************************************
global with sharing class AsyncJobConvertLead implements
mba_services.IRunAsyncJob {

    global mba.AsyncJobServices.AsyncJobResult
run(mba_services__Async_Job__c asyncJob){
        String userEmail;

        try{
            String outputMessage;

            //Get job parameters
            map<String, object> m_params = (map<String, object>)
JSON.deserializeUntyped(mba_services__Job_Params__c);
            String leadId = String.valueOf(m_params.get('leadId'));
            userEmail = String.valueOf(m_params.get('userEmail'));

            //Query the Lead
            Lead leadRecord = [select Id,Name,Email from Lead where Id =
```

```
:leadId];

            //search for existing contact. Here we search by the lead email.
            list<Contact> contactList = [select Id,Name,Email from Contact
where Email = :leadRecord.Email];

            if(contactList.isEmpty()){
                //Convert the lead

                Database.LeadConvert lc = new Database.LeadConvert();
                lc.setLeadId(leadRecord.id);
                LeadStatus convertStatus = [SELECT Id, MasterLabel FROM
LeadStatus WHERE IsConverted=true LIMIT 1];
                lc.setConvertedStatus(convertStatus.MasterLabel);
                Database.LeadConvertResult lcr = Database.convertLead(lc);

                if(lcr.isSuccess()) {
                    outputMessage = 'Lead was converted. New contact Id; ' +
lcr.getContactId();
                }
                else{
                    outputMessage = 'Failed to convert lead. Error: ' +
lcr.getErrors()[0].getMessage();
                }
            }
            else {
                //return message
                outputMessage = 'Seems there are already ' +
contactList.size() + ' with the lead email. Please verify and convert or
disqualify the lead manually';
            }

            //Close the job
            return new mba_services.AsyncJobServices.AsyncJobResult(true,
outputMessage, userEmail);
        }
        catch(Exception ex){
            return new mba_services.AsyncJobServices.AsyncJobResult(false,
ex.getMessage() + '. ' + ex.getStackTraceString(), userEmail);
        }
    }
}
```

2.Use apex code in order to add an Async Job that will execute the above class.

```
map<String, object> m_params = new map<String, object>();
m_params.put('leadId', leadId);
m_params.put('userEmail', UserInfo.getUserEmail());

mba_services.AsyncJobServices.addAsyncJob (
   'Auto Convert Lead',
   'Apex',
   'AsyncJobConvertLead',
   'High',
   JSON.serialize(m_params),
   null,
   true);
```

4.Async Job Template

With the Async Job Template you can easily define simple processes and schedule it to run with the Async Job System.

4.1 Define Async Job Template

1.Go to tab **Async Job Template**
2.Click button **New**
3.Set up the following input:

Async Job Template Name - free text
Description - free long text
Status - set either Test or Live. In the status Test, when the template will be run it will be in test mode and no changes will be committed.

- It is recommended that at first a new Async Job Template will be with a status Test. Only after running it in Test mode and confirming it is working as expected change the status to Live.

Rollback on Error - Check this option will rollback process changes if at least one error occurs. Keep uncheck to allow partial success.
Is Batch - check this option if your process should support large volume of data
Batch Size - if it is a batch process, set the batch size. The processing records are split into groups according to the batch size. For example, if the process should process 1000 records and the batch size is 200, then the records will be split to 5 groups of 200.

4.Click button **Save**
5.Click button **Set Actions**
6.Repeat the following steps per each action that you want to execute under the process:

6.1 Click **Add Action**

6.2 Set the Action Title/ Action Type and click the icon 

6.3 Dialog will be open to set additional input according to the **Action Type**:

- For action type **Get Records:**
  -Set the Related Object
  -Optionally set Filter. For more information see How to Use Filter Data Component.
  -Optionally define order for the query using the inputs: Order By and Order Direction.
  -Optionally limit the number of records for the query using the Limit input.
- For action type **Update Records**:
  -Set Data Source, it can be either query or previous Get Record action. In case the Data Source is Query, then set also the same input as described in the **Get Records** action.
  -Set the Fields Setup. For more information see How to Use Field Setup Component.

- For action type **Delete Records**:
  -Set Data Source, it can be either query or previous Get Record action. In case the Data Source is Query, then set also the same input as described in the **Get Records** action.
- For action type **Insert Records**:
  -Set Data Source, it can be either query or previous Get Record action. In case the Data Source is Query, then set also the same input as described in the **Get Records** action.
  -Set the Object to Insert
  -Set the Fields Setup. For more information see How to Use Field Setup Component.

6.4 Click button **Close**

7.Click button **Save**

4.2 Run Async Job Template

- Running the Async Job Template can modify your organization data. Use it with caution and verify your process before running it.
- You can monitor the job progress in the Async Job System. See section Monitor Async Jobs

1.Go to Async Job Template record
2.Click button **Create Async Job**
3.Set up the following input:
Job Priority - select the priority for the job
Notify Email - email address to notify once the process run complete
Time to Run - can run the job at a specific date and time.
Is Repeated - check this option to set repeated job or uncheck to run it once
Repeated Type - if the job is repeated, set the repeated type. Can choose either Daily, Weekly, Monthly, Hourly, Minutes
Repeated Interval - if the job is repeated, set the interval. This number context is according to the Repeated Type. For example, if it is repeated hourly, then interval 8, means that the job will run every 8 hours.
4.Click button **Next**

4.3 Export/Import Async Job Template

If you are using sandbox, then you might want to define the Async Job Template in the sandbox, test it and then create the same in production. In order to save manual work, you can export the record(s) from the first environment and import it to the second environment.

- Export Multiple Async Job Templates
    - Go to the tab Async Job Templates
    - Using the checkbox select the Async Job Templates that you want to export
    - Click the button **Export Templates**
    - You should receive an email with a file attached (.json file). The file can be used later for the import process.
- Import Multiple Async Job Templates
    - Go to the tab Async Job Templates
    - Click the button **Import Templates**
    - Click the button **Upload Files** and select the file that you received during an export process
    - The screen will show the Async Job Templates in the file. Using the checkbox select the templates that you want to import.
    - Click the button **Continue**
- Export Single Async Job Template
  Go to a specific Async Job Template record and click the button **Export**. You should receive an email with a file attached (.json file). This file will hold the template settings.
- Import Single Async Job Template
  Go to the tab Async Job Templates and click the New button.
  Click the button **Import**, a dialog box will be opened.
  Click the button **Upload Files** and select the file that you want to import.

➔ Note. If the exported file was created using the multiple method, then it cannot be imported using the single method.

5.Automation Templates

Using Automation Template you can configure to run Async Job Template from a trigger context.

5.1 Create Automation Template

1.Go to tab **Automation Template**
2.Click button **New**
3.Set up the following input:

- **Name** - free text
- **Description** - free text
- **Related Object** - object in context
    - If the required object is not listed, please refer to section Trigger Framework and first add a trigger on the relevant object.
- **Order** - number. In case you using multiple automation on the same object, you can control their order
- **Insert** - checkbox. Whether this automation should be running on insert.
- **Update** - checkbox. Whether this automation should be running on update.
- **Delete** - checkbox. Whether this automation should be running on delete.

4. Click the button **Save**.

- **Entry Condition** - condition that defines when a record should be entering this process.
- **Old Condition** - condition that defines the old state of a record that will be entering this process.

After completing the Automation Template setup you can click the button **Activate**

5.2 Adding Job under Automation Template

The template defines the event, next you can set the action that should be running when the event occurs.

1.In the Automation Template that was created, click the button **Add Job**.
2. Set up the following input:

- **Name** - free text
- **Type** - selection. to type supported:
    - **Async Job Template** - run specific Async Job Template
    - **Blocker** - create criteria to block the process
- **Log Level** - selection. logs that will be reported from the process
    - Use the option **All** with caution and only for debugging, as it will create detail log on every event that the process runs.
- **Order** - number. In case of multiple jobs you can control their order
- **Active** - checkbox to activate this job.

In case the Type is Async Job Template, use the input to search the relevant template, in addition set:

- **Threshold** - number. If you are working with a bulk of records, define the maximum number of records that will be running synchronously, above the threshold it will be running in the background process. ,
- **Run in Background** - checkbox. You can set this job to run in the background, regardless of the **Threshold.** This will improve the user experience and the performance.
-

In case the Type is Blocker, set the following input:

- **Blocking Criteria** - condition that when a record applies to the blocker will fire. For more information see How to Use Field Setup Component.
- **Blocking Message** - text. Message to be displayed when the blocker apply.

6.Other Services

- The next sections require development skills
- Although the application doesn't restrict you from adding changes directly in the production environment, it is advised to have any code changes first in your sandbox, test it and then deploy to production.

6.1 Database Service Class

DatabaseManager is a service class that provides common methods for running DML actions. Using the service ensures that security aspects are enforced and errors reported properly to the caller.

Signatures are

- runDMLAction (action, list-of-records)
- runDMLAction (action, list-of-records, parameter-maps)
    - action options are: 'insert', 'update', 'delete', 'upsert'
    - parameter-maps options are:
        - **allOrNone** (Boolean, default true), indicate if error in one record should fail the entire DML process or other records can processed
        - **throwSecurityException** (Boolean, default false), will throw custom exception of type SecurityMBAException in case the user missing any permission
        - **skiptSecurityCheck** (Boolean, default false), set it to true to skip the security check

Examples:

```
List<Account> accountList = [SELECT Id FROM Account];

//some logic that modify the accountList

mba_services.DatabaseManager.DatabaseResult insertResult =
                        mba_services.DatabaseManager.runDMLAction('insert',
accountList, new Map<String, object>{'allOrNone'=> false});

if(insertResult.isSuccess() == false){
      String errorMessage = insertResult.getAllErrorMessages('\n');

      //report to log
}
```

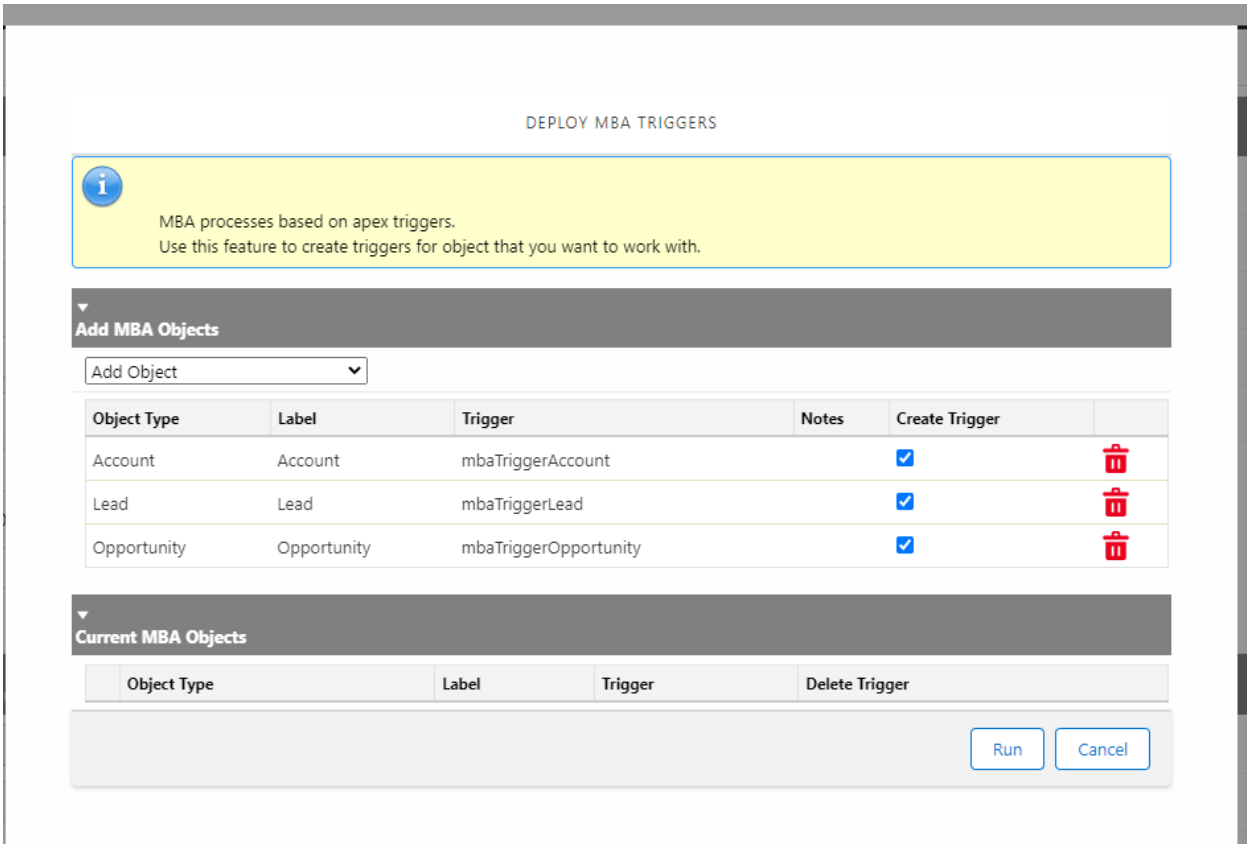
6.2 Trigger Framework

Trigger framework provides better control and monitor of the processing during apex triggers. With the application you can easily build such a framework and add a custom process that will be running by the framework.

- To allow the app adding new triggers in your org, first enable this option by navigating to Salesforce Setup ->
  Custom Setting->
  Click **Manage** next to **Org Settings**
  Click **Edit**
  Tag the field **Enable Trigger Deployment**
  Click **Save**

6.2.1 Create new trigger

- Go to the tab **MBA Settings**
- Click button **Deploy MBA Triggers**
- Using the drop down list add objects that you want to create triggers for them
- Click **Run**
  - Deployment process will generate apex trigger and apex test class per each selected object



- Note, if you working in sandbox you can generate the apex trigger and apex test class manually, using the following pattern (Account object is used in this example)

Apex Trigger:

```
trigger mbaTriggerAccount on Account (before insert, before update, before
delete, after insert, after update, after delete){
     mba_services.TriggerHandler.runTrigger ('Account');
}
```

Apex Test Class

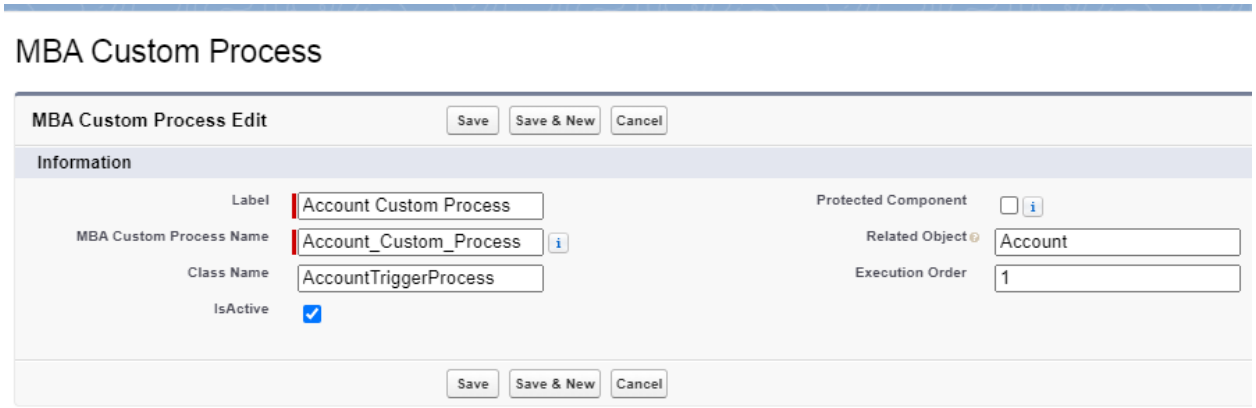
```
@IsTest
private class mbaTestAccount{
  testMethod static void runTest(){
        Test.startTest();
        mba_services.TestManager.createRecord('Account', null, true);
        Test.stopTest();
  }
}
```

6.2.2 Add Custom Process to Run by the Trigger Framework

For adding such process you need to add 2 items:

1. **Custom Metadata record**

In Salesforce Setup, got to **Custom Metadata Types** ->
Click **Manage** next to **MBA Custom Process**
Click **New** and fill the following fields:
Label: Descriptive name
Class Name: Apex class that should be running
IsActive: on/off flag
Related Object: Object that related to this process
Execution Order: in case, you have several processes on this same trigger this field allows to control the execution order.



2. **Custom apex class**

Create apex class that implements the interface **mba_services.IRunTriggerMBA**, you can use the following example as pattern

```
global with sharing class AccountTriggerProcess implements
mba_services.IRunTriggerMBA{

      //Invoked from trigger
    global mba_services.AsyncJobServices.TriggerRunResult run(
        String objectType, String triggerAction, boolean isAfter,
list<sObject> l_new, list<sObject> l_old, map<Id, sObject> m_new, map<Id,
sObject> m_old){

        mba_services.AsyncJobServices.TriggerRunResult runResult = new
mba_services.AsyncJobServices.TriggerRunResult(true, '');

        try{
            //Custom Logic
        }
        catch(Exception ex){
            runResult.isSuccess = false;
            runResult.message = 'Exception: ' + ex.getMessage() + '.
Details: ' + ex.getStackTraceString();
        }

        return runResult;
    }

      //Optional process settings
    global map<String, object> getProcessSettings(){
        return null;
    }

      //Optional process settings
    global map<String, object> handleSettingButonClick(String actionName,
map<String, object> inputs){
        return null;
    }
}
```

7.Additional Components

7.1 How to Use Filter Data Component

The Filter Data Component is widely used in many sections in the application. It is mainly, but not only, used to construct logical criteria for specific object type.

The following section describes how to use the component.

1. Add criteria.

Click the icon  to add new criteria.
You will notice a new line being added. In the **Field** input you can either choose field from the dropdown list or use the icon  to open the lookup window and select the field. Select the **Operator** from the dropdown list and finally add the **Value** to compare the field to.

- You can add several criteria lines. In case, there is more than one line you will need to set a relation between them in the **Criteria Logic**. For example, if you have 3 criteria lines, the criteria logic might be **1 AND (2 OR 3)**

2. Remove criteria

Click the icon  next to the criteria line you want to remove.

3. Validate criteria

Click icon  to validate your criteria. If there is an error it will be displayed above the criterias.

4. Clear Criteria

Click icon  to clear the criteria.

7.2 How to Use Field Setup Component

The Field Setup component provides the ability to map values to fields of a specific object type.

The following section describes how to use the component.

1. Using the dropdown **Add Field** select the field that you want to map. This field will be added to the list of fields in the screen.
2. Select the Source for the value. The Source might be either Value (specific value) or Formula for more complex calculations.
3. Set the Value input. In case the Source is Formula, then use the icon  to select merge fields from the object or the icon  to use a function.