Occurrences System Usage Guide:

Solution Description:

If you need the connection between other systems and your CRM to be fully dynamic and its logic to be controlled on the CRM side, this solution will be beneficial for your organization. Methods of type Post, Put, Delete, etc., can all be developed on the CRM side by a CRM Developer, and input and output changes can occur internally without the need for API development or modification.

Example:

1. A method needs to be called on the CRM side to search for the customer ID (unique field) and then update specific information.
2. A Contact needs to be created in CRM with specific data.
3. An Account needs to be created or modified in CRM.

All of these cases require a separate method to be developed for each, and a developer to spend a certain amount of time to perform this activity. Now, imagine that only one dynamic method is developed, but all these businesses are developed as Actions on the CRM side, and the CRM developer controls all the logic in these actions.

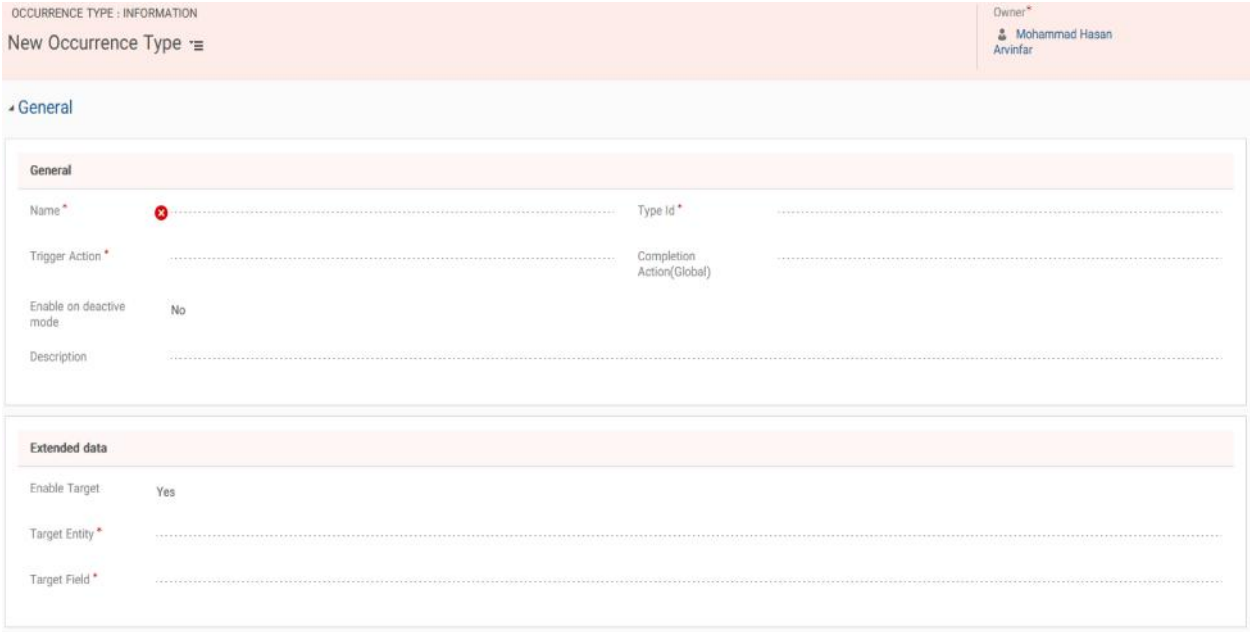
The Occurrences System does exactly this for you. Define a type and create a record from it. These records are operations that are performed and maintain your log.

Technical Development:

This solution consists of two entities and a plugin, with the main control of the system being handled by this plugin.

Occurrence Type Entity:

Each record of this entity defines the business type and logic, such as creating a customer, changing the specifications of an Order. In fact, the desired action is defined in this record, and the dynamic method creates another record of this type.



Name	The name of the logic or business you are defining
TypeId	A unique code that must be called by a dynamic method
Trigger Action	An action that controls the business and logic
Enable on deactive mode	It should be usable if the Type is disabled
Description	Descriptions
Completion	A Global action that will be executed after the successful execution of TriggerAction
Enable Target	If Yes, it means that TriggerAction has a Target
Target Entity	The entity on which the action is executed
Target Field	The entity field on which the action should be executed

Occurrence Entity:

This entity executes commands based on the specified Type and logs the result after successful completion. In fact, the dynamic method should create a record of this entity type and a specific field with the desired data. Successful creation of this record and the Done status code indicate successful execution of the commands.

Below are examples that fully and clearly explain the process execution path:

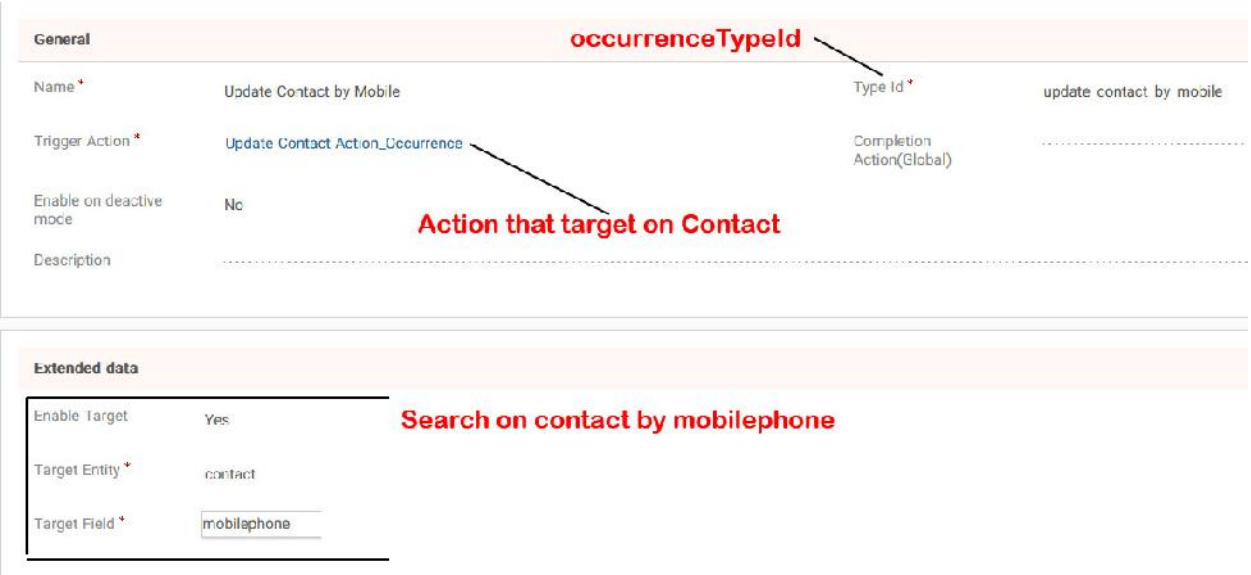
First Example:

Suppose you need to update the information of a Contact, such as searching for a **Contact** with the mobile number 09112320258 and changing the first name, last name, father's name, gender, and company information. As a result, you need to define a **JSON** that specifies the inputs.

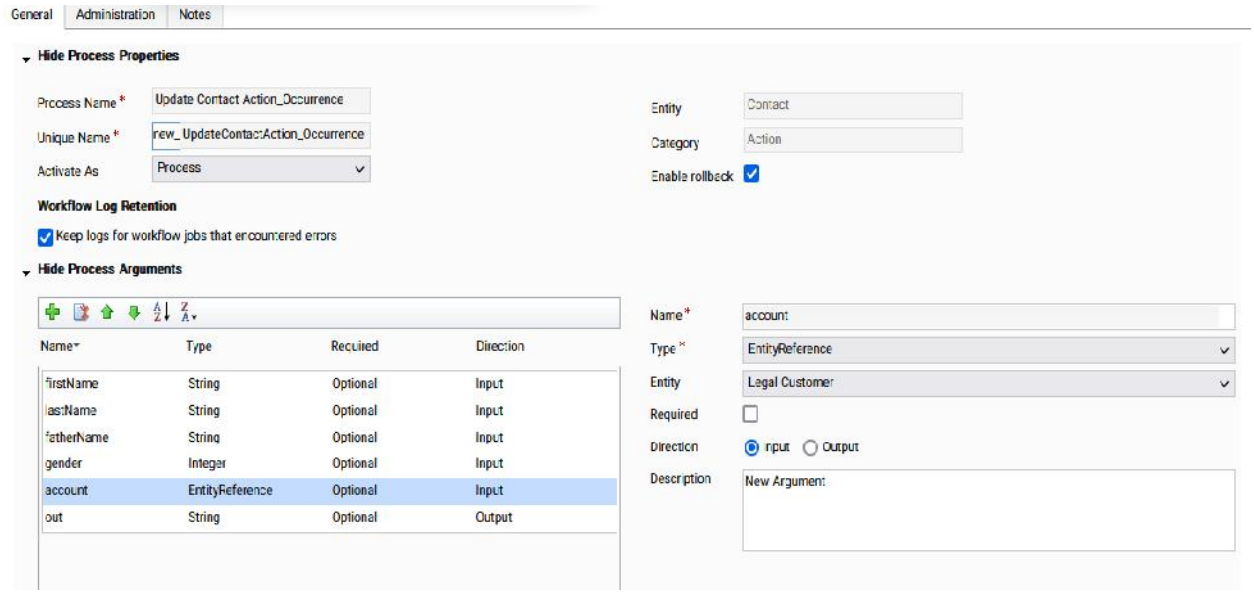
```
{
      "occurrenceTypeId":"update_contact_by_mobile",
      "targetValue":"09112320258",
      "firstName":"Alice",
      "lastName":"Portman",
      "gender":0,
      "account":"6A534348-6FBE-E811-80DB-005056B6C839"
}
```

In the **JSON** above, the **TypeId** related to **OccurrenceType** is specified, and a unique identifier is defined as the **Target** so that changes are made to that record. The rest of the information is also added to the Json as needed.

Occurrence Type Definition:



The action in OccurrenceType is as follows:

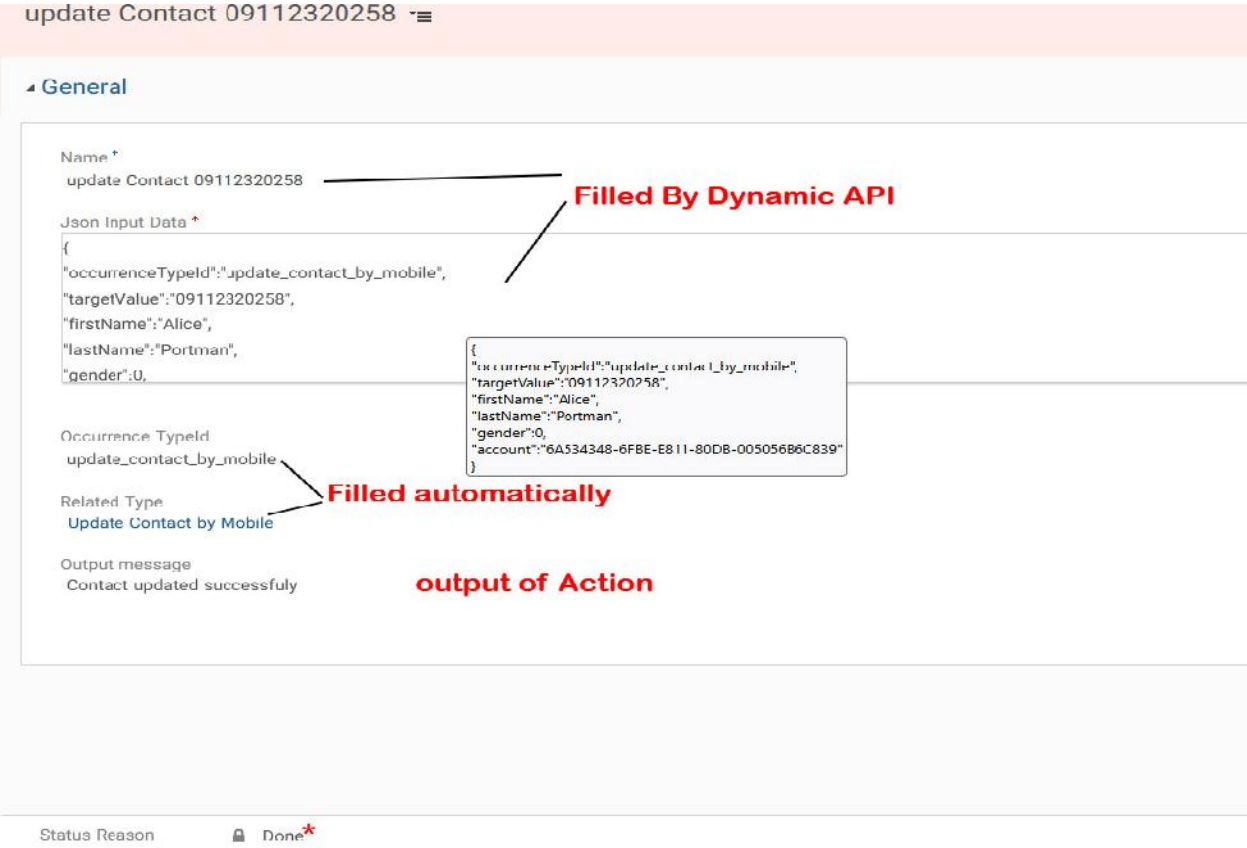


The **action parameters** must match the Json input names exactly, and it is better for the action parameters to be Optional. An output can also be defined for the action so that after successful execution, the output value is also logged.

After defining the OccurrenceType, an Occurrence record must be defined, and the defined Json must be placed in the new_**jsoninputdata** field. (This part must be created by the dynamic method). As soon as this record is created, the controlling Plugin is executed, extracts the **occurrenceTypeId** from the **JSON**, executes the desired Action, and after successful completion of the process, places the desired output in the **new_outputmessage** field and sets the record status to Done.

If the **CompletionAction** field is defined in the OccurrenceType, after successful execution of the process, that action is also executed and performs the desired process. (Async)

Note: If a parameter named self is defined in the action inputs, the Guid of the generated Occurrence record is placed in this parameter so that it can be used for linking records if needed.



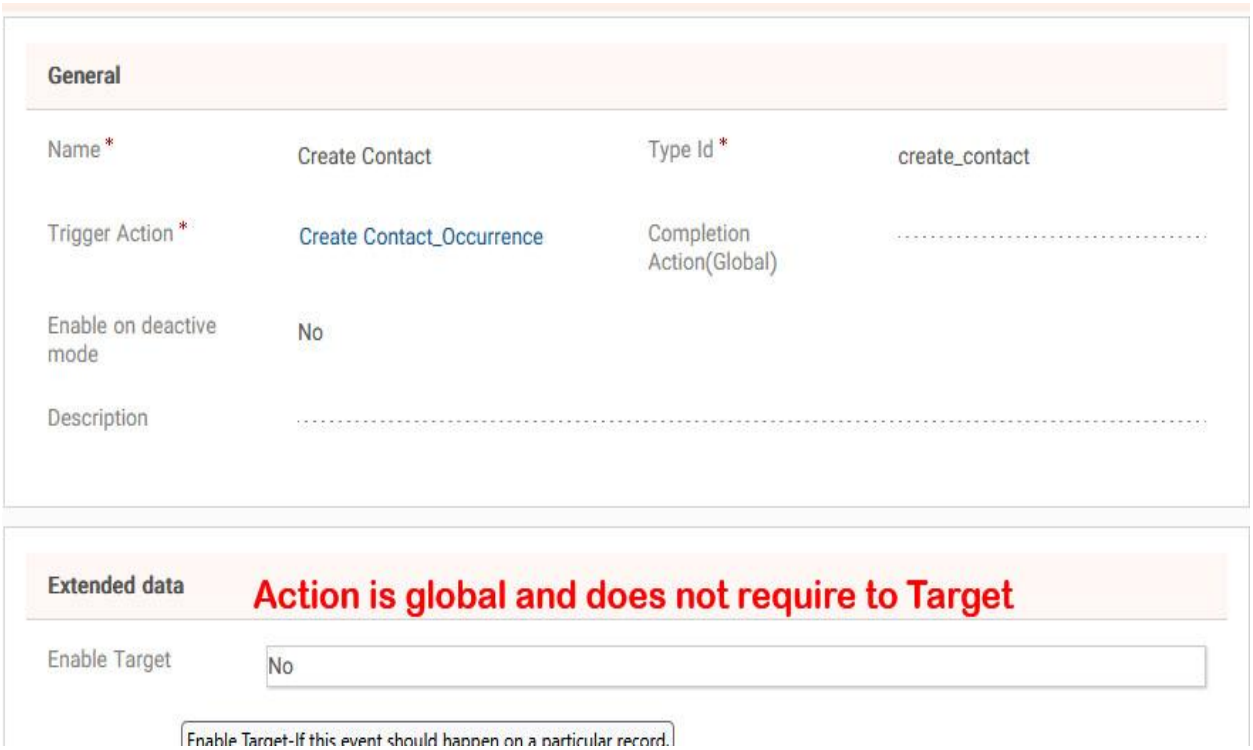
Second Example:

We need to create a Contact with custom values.

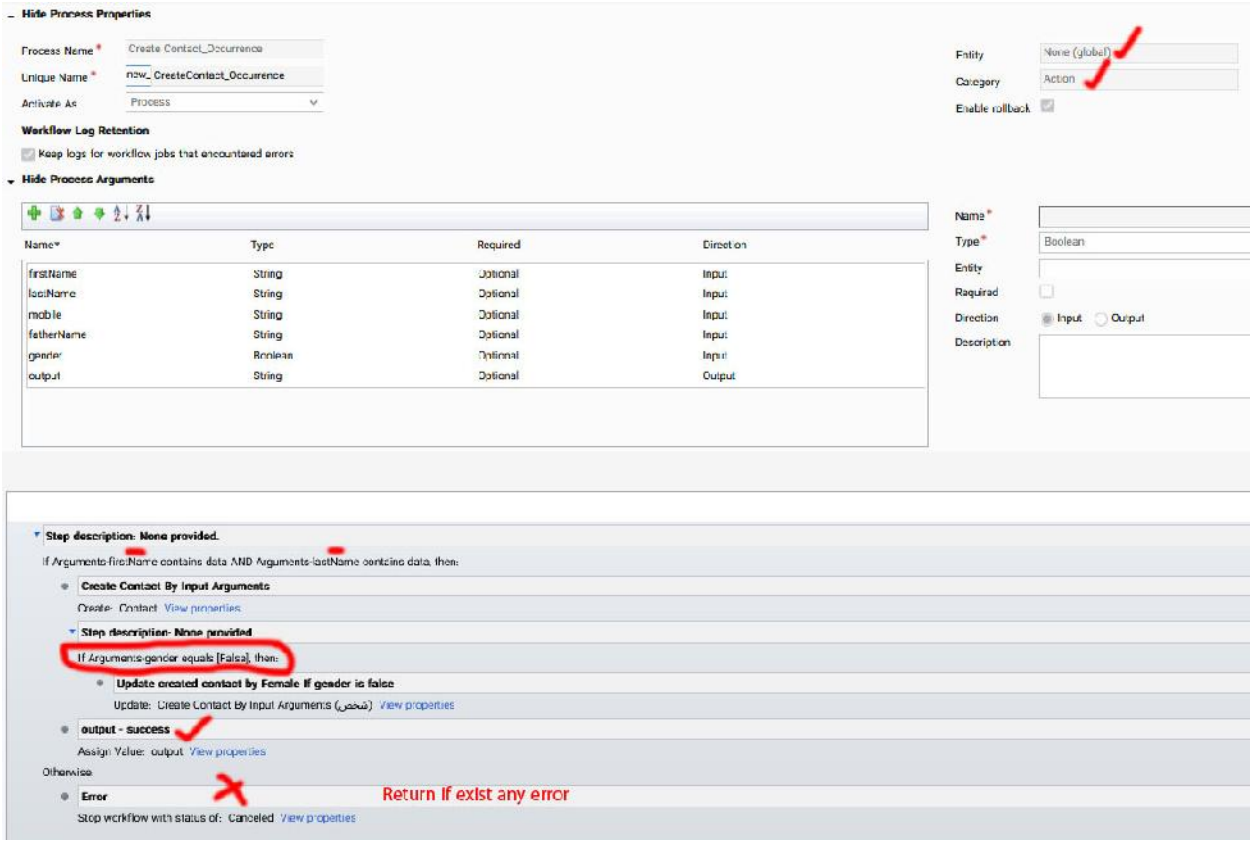
First, we implement the desired Json. It is clear that in this logic we do not need the Target field, and the desired action must also be Global.

```
{
      "occurrenceTypeId":"create_contact",
      "firstName":"David",
      "lastName":"Simon",
      "fatherName":"Jack",
      "gender":false,
      "mobile":"09112320258"
}
```

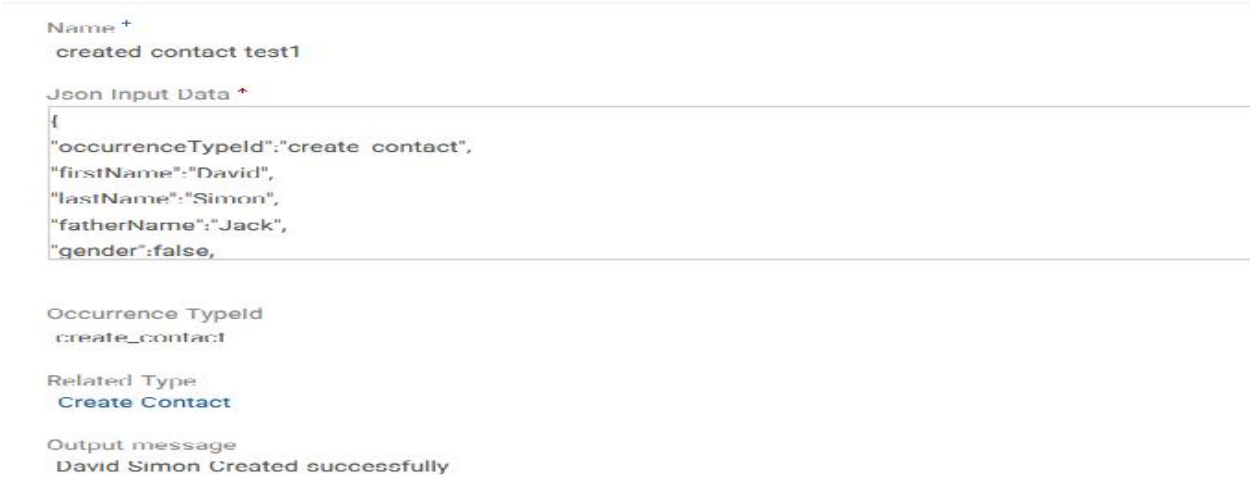
Create Occurrence Type with a unique ID



Define the desired action



Now we must transfer the created Json to the dynamic method parameter to create the Occurrence record, and after creating the Occurrence record, the customer is also created.



This system supports any standard Data Type that exists in CRM and even performs the required Cast in some cases, considering the destination DataType (action). For example, in the Json below, if the data type of the action parameters is DateTime, Money, Boolean, and the Json input values are as follows, the data type conversion is done automatically, and the error rate is reduced.

```
{
      "occurrenceTypeId":"xxxxxx",
      "firstName":"Tim",
      "lastName":"Marson",
      "birthDate":"1988-02-16T12:51:07.397Z",
      "deposit":”50000000”,
      "gender":7
}
```

- Convert String birthdate to DateTime
- Convert String deposit to Money
- Convert int gender to Boolean True

If you need more explanations or suggestions to improve this system, I am available through the following:

Mobile: +989112320258

Email: aria.rvn@gmail.com

LinkedIn :https://linkedin.com/in/ariarvn

GitHub :https://github.com/marvinfar