Writing Data to a SOAP Web Service

Table of Contents

Introduction

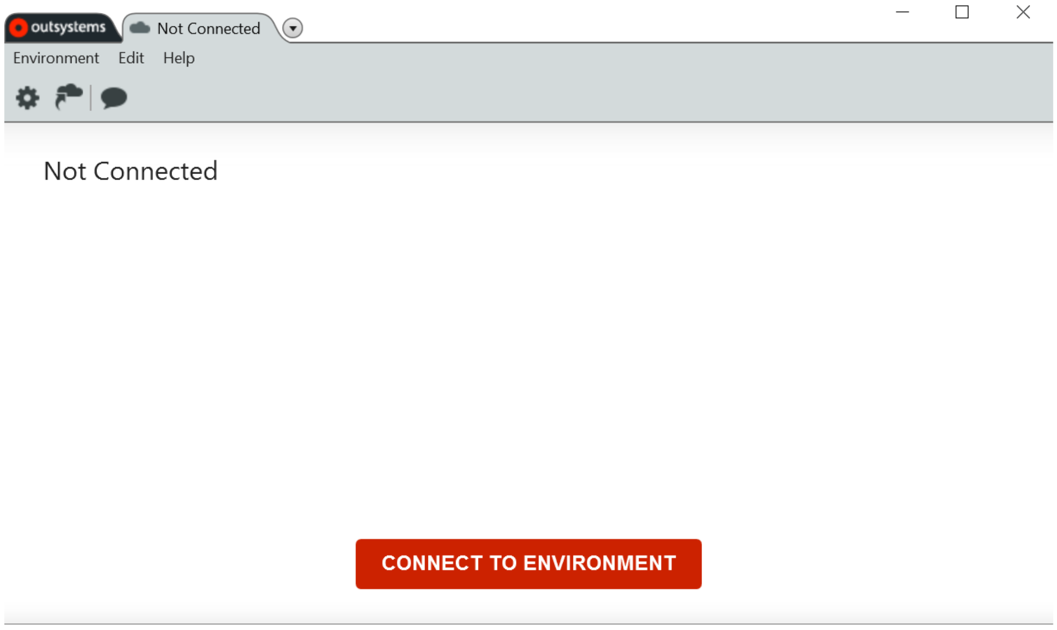
In this lab, we are going to integrate with a SOAP Web Service to allow to manipulate data in an external system. We are going to use two SOAP methods to insert and update information of Contacts in the external system.

Connect to an Environment

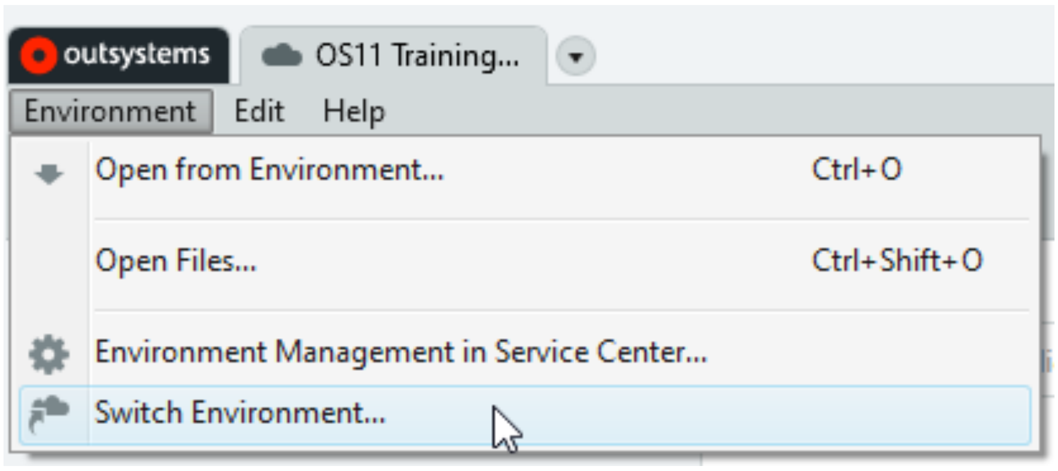
When we open Service Studio for the first time, we will need to connect to an **environment** where the OutSystems platform server generates, optimizes, compiles, and deploys OutSystems applications.

1) Open Service Studio and access the **Connect to Environment** dialog. This can be done in two ways.

   a) If you are not logged in to any environment, click on **Connect to Environment**.



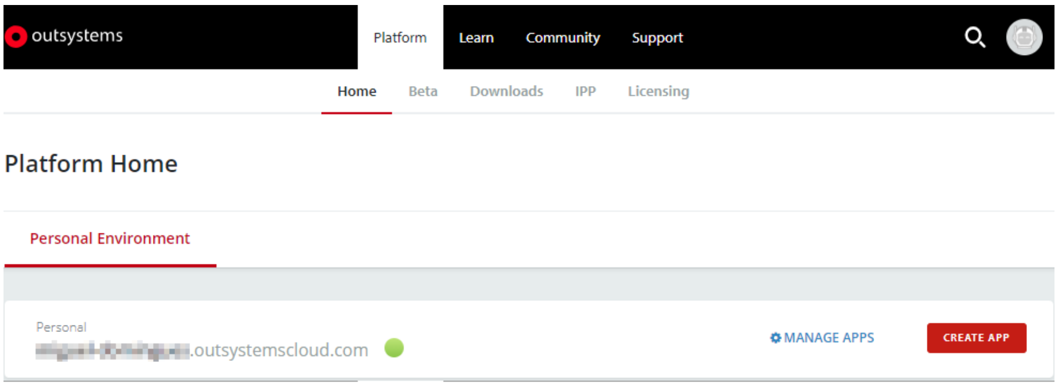
b) If you are already logged in to an environment, select the **Switch Environment...** option from the Environment menu at the top.


outsystems
OS11 Training...
Environment
Edit
Help
Open from Environment... Ctrl+O
Open Files... Ctrl+Shift+O
Environment Management in Service Center...
Switch Environment...

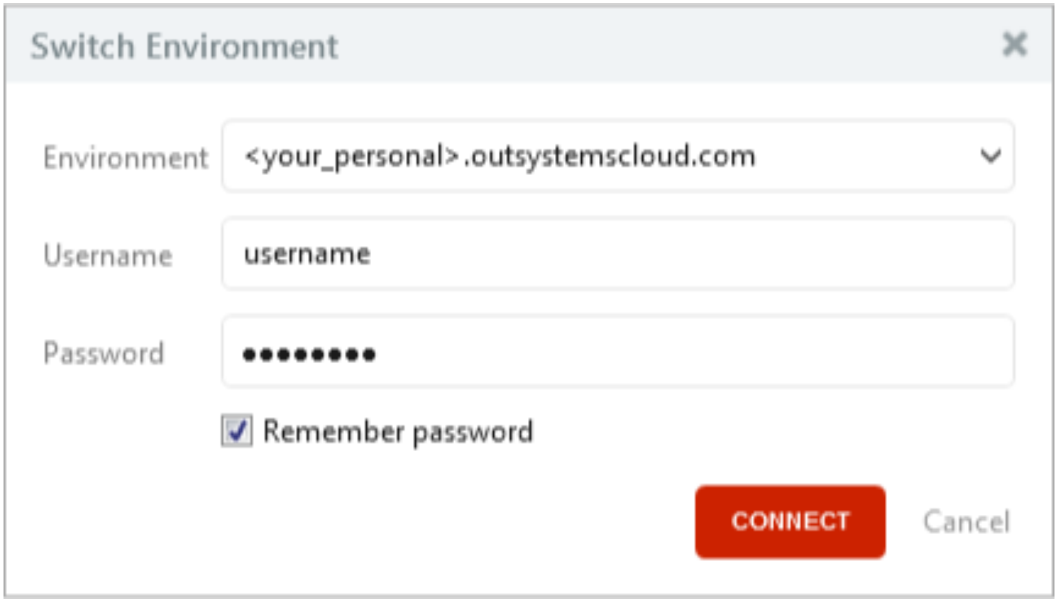
2) Connect to your OutSystems personal environment.

a) If you are using your Personal Environment, you can find its address in the OutSystems website and log in.

b) Under the **Platform** tab and then under the **Personal Environment** tab the environment address (or **Server Address**) can be found.



c) Back in Service Studio, use that Environment and login with your OutSystems community email (username) and password.



Environment: <your_personal>.outsystemscloud.com

Username: username

Password: ●●●●●●●●●

[✓] Remember password

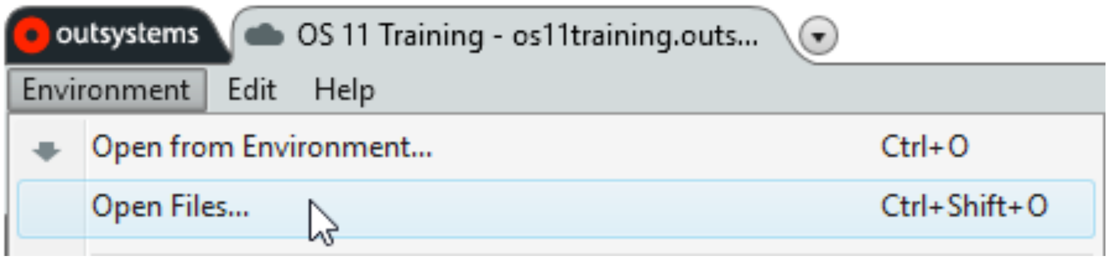
Get to know the scenario

Install the Contacts application

If you have completed the **Retrieving Data from a SOAP Web Service** exercise lab before, you can skip this section and jump to the next page. Otherwise, follow the instructions below to install the quick start application.

Open and publish the **SOAP Contacts - Writing Data.oap** in your personal environment. The oap file can be found in the Resources folder.

1) In the Applications, open the Environment menu and select Open Files...


outsystems
OS 11 Training - os11training.outs...
Environment
Edit
Help
Open from Environment... Ctrl+O
Open Files... Ctrl+Shift+O

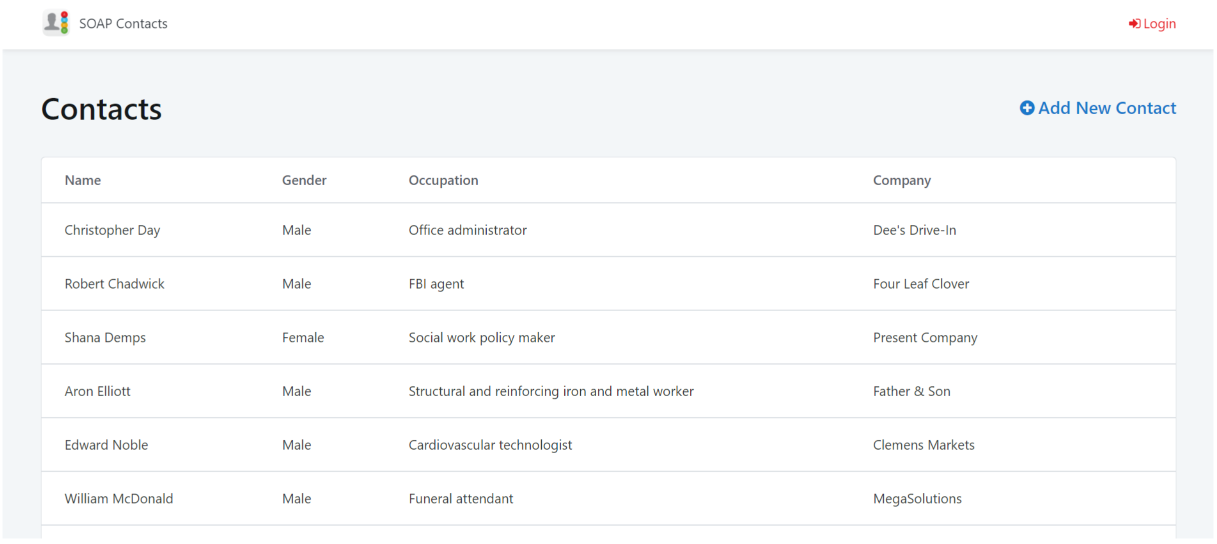
2) In the Open dialog, change the File Type dropdown option to **OutSystems Application Pack (*.oap)** and then open the SOAP Contacts - Using Data.oap.

3) Click Proceed when asked.

4) Wait for the installation to complete and then proceed.

Business Case Overview

The SOAP Contacts application is a simple application, only containing a couple empty Screens, Preparation and some Screen Actions, also empty. These elements are where we'll create the logic to get the Contacts and display the list on the screen. This quick start application is simply to speed up the setup part and start right away working with the external SOAP web service.



SOAP Contacts

Login

Contacts

Add New Contact

Name	Gender	Occupation	Company
Christopher Day	Male	Office administrator	Dee's Drive-In
Robert Chadwick	Male	FBI agent	Four Leaf Clover
Shana Demps	Female	Social work policy maker	Present Company
Aron Elliott	Male	Structural and reinforcing iron and metal worker	Father & Son
Edward Noble	Male	Cardiovascular technologist	Clemens Markets
William McDonald	Male	Funeral attendant	MegaSolutions

Consume a SOAP Web Service Method to Create a Record

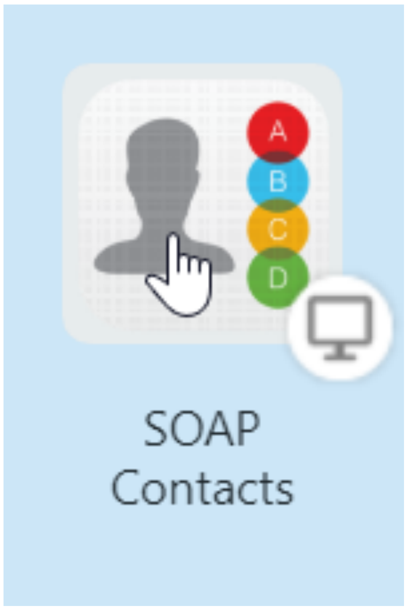
In OutSystems, integration with SOAP Services can be straightforward. OutSystems helps us to generate all the methods and data structures needed to integrate with an external system.

Before you consume any SOAP Web Service it's important to gather all the information you need from the SOAP Web Service documentation. Information such as the expected structures and some examples may be really useful when consuming an external service. In this lab, the documentation for the external SOAP Web Service is available here.

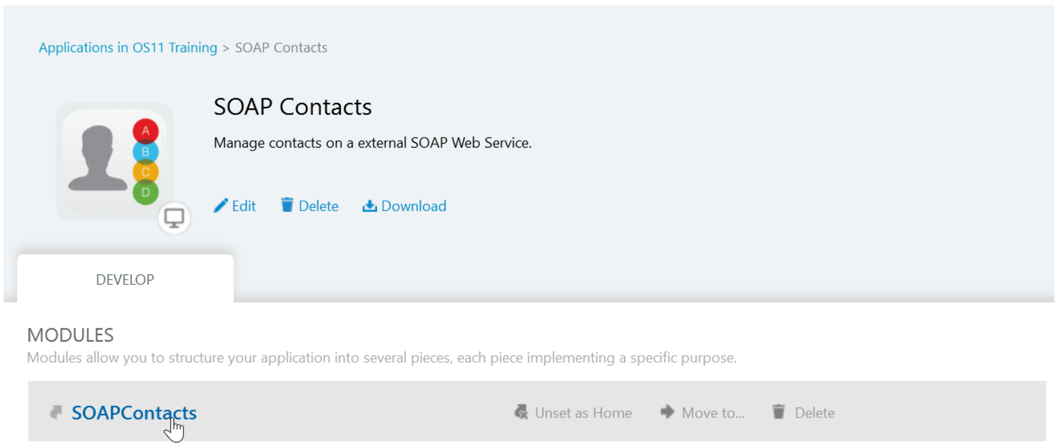
In this section we will extend the integration with the external SOAP Web Service. The extended integration will allow to create data on the external web service thru a form provided in our application.

1) Open the Contacts module.

   a) In the Applications list, locate the **SOAP Contacts** Web application and open it.



b) Open the **SOAPContacts** module.


Applications in OS11 Training > SOAP Contacts

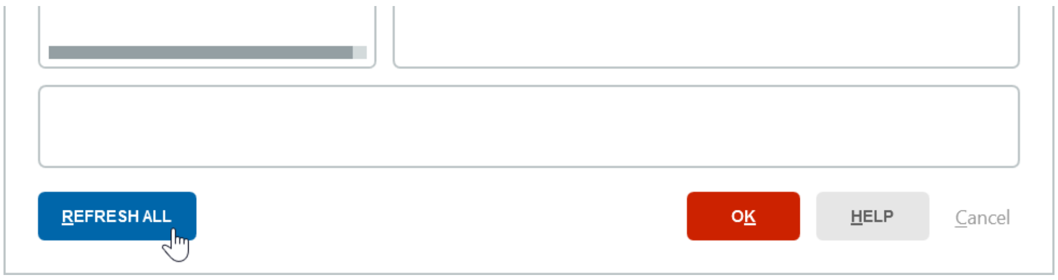
SOAP Contacts

Manage contacts on a external SOAP Web Service.

MODULES

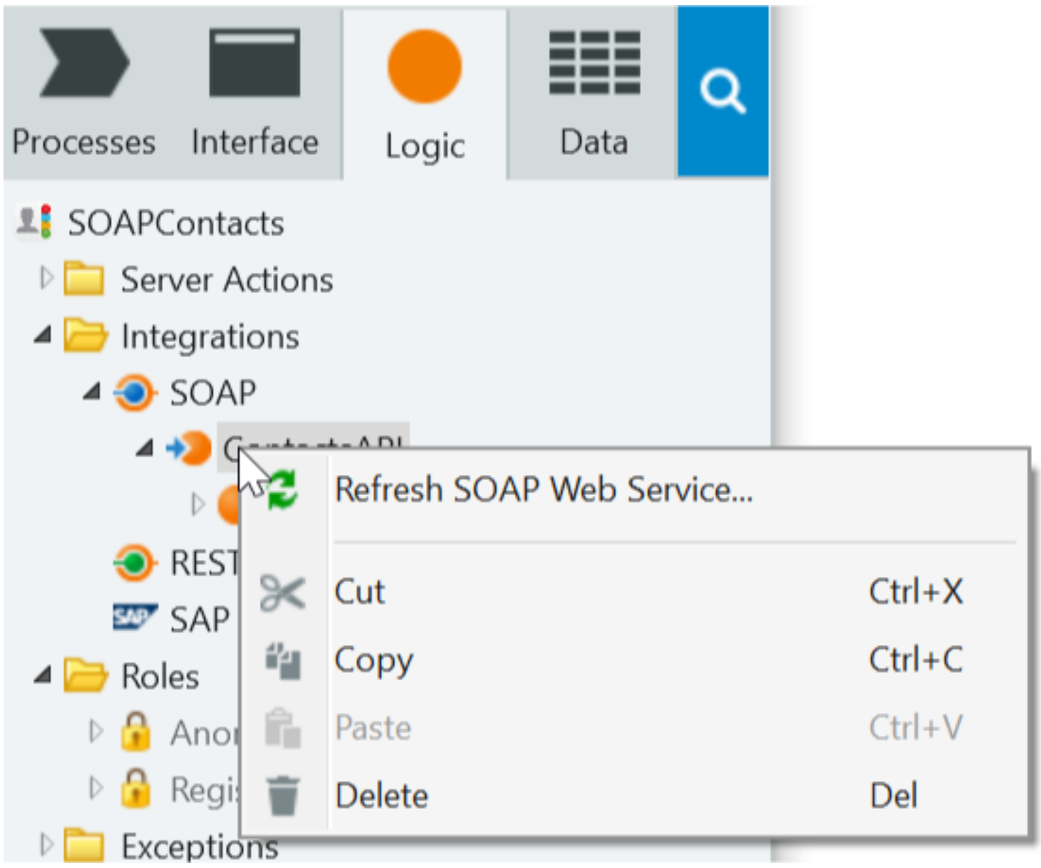
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

c) If a dialog appears asking to update references, click Yes to open the Manage Dependencies window. There, click on **Refresh All** and then OK.



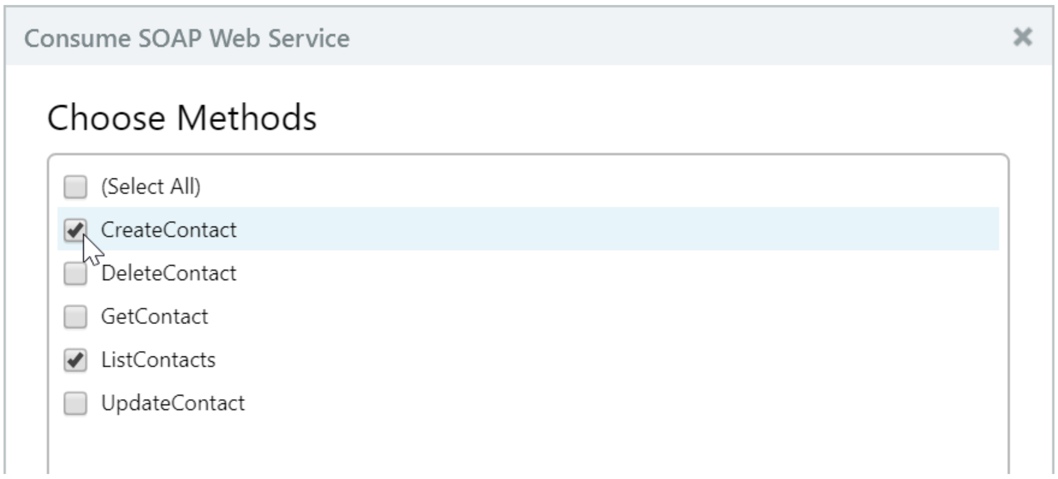
3) Consume the method to create a Contact in the external SOAP Web Service.

a) Switch to the Logic tab and, in the Integrations folder, right-click the **SOAP** element and select *Refresh SOAP Web Service...*



b) If a dialog appears asking whether to proceed, click **Yes**.

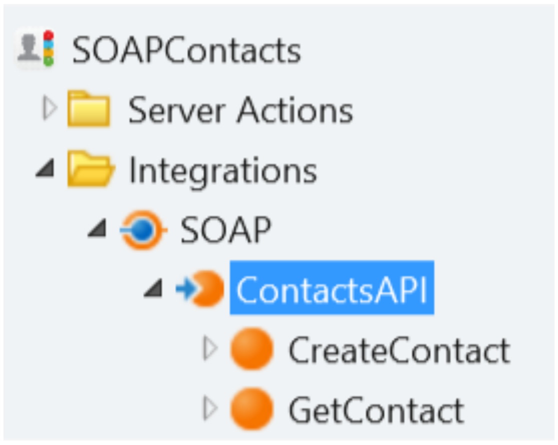
c) In the next dialog *ListContacts* should be already selected. Select **CreateContact** and press *Finish*.


Consume SOAP Web Service

Choose Methods

- [ ] (Select All)
- [x] CreateContact
- [ ] DeleteContact
- [ ] GetContact
- [x] ListContacts
- [ ] UpdateContact

d) Change the name of the SOAP Integration from v1 to ContactsAPI.



NOTE: This step is not mandatory, but it allows developers to have custom or more familiar names in their integrations.

Add Contact User Interface

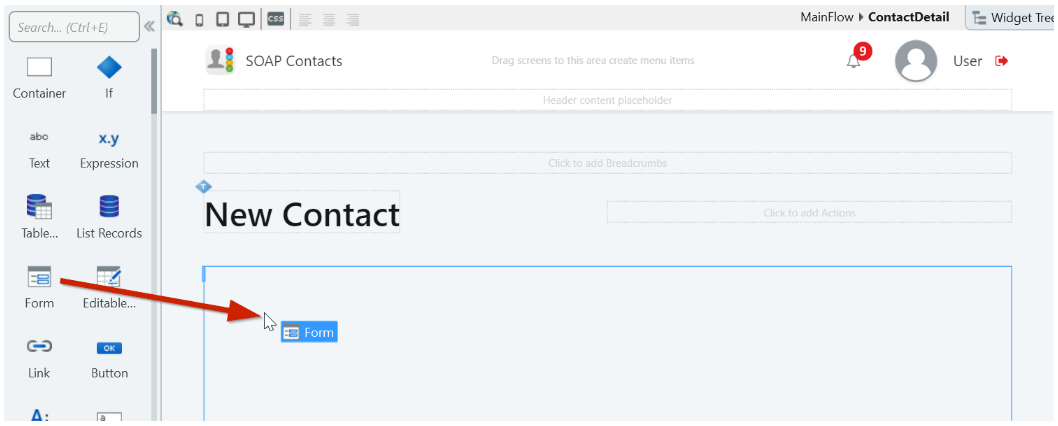
In this section we will modify the existing empty ContactDetail screen to allow creating a new contact. In our sample app, the ContactDetail screen has already been created. It has an input parameter (Email) that will be later used to edit Contacts. There's already a link from the main Contacts screen to add a new Contact, which navigates to the ContactDetail screen.

Let's now modify the ContactDetail screen and the Save screen action to create a new contact in the external system.

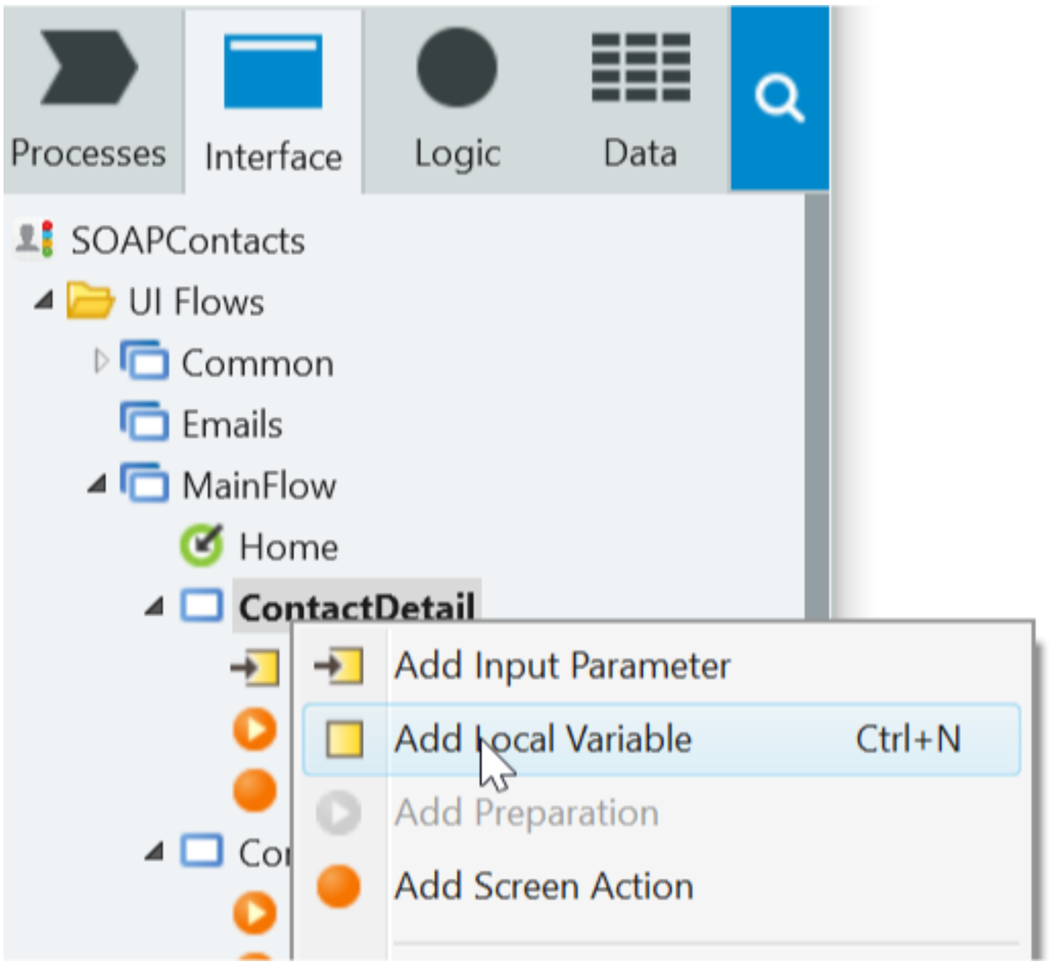
1) Create a new Form in the ContactDetail screen with the required inputs for the CreateContact SOAP Web Service method. Define the logic to invoke the SOAP Web Service method inside the Save screen action of the same screen.

    a) In the Interface tab, open the **ContactDetail** Web Screen.

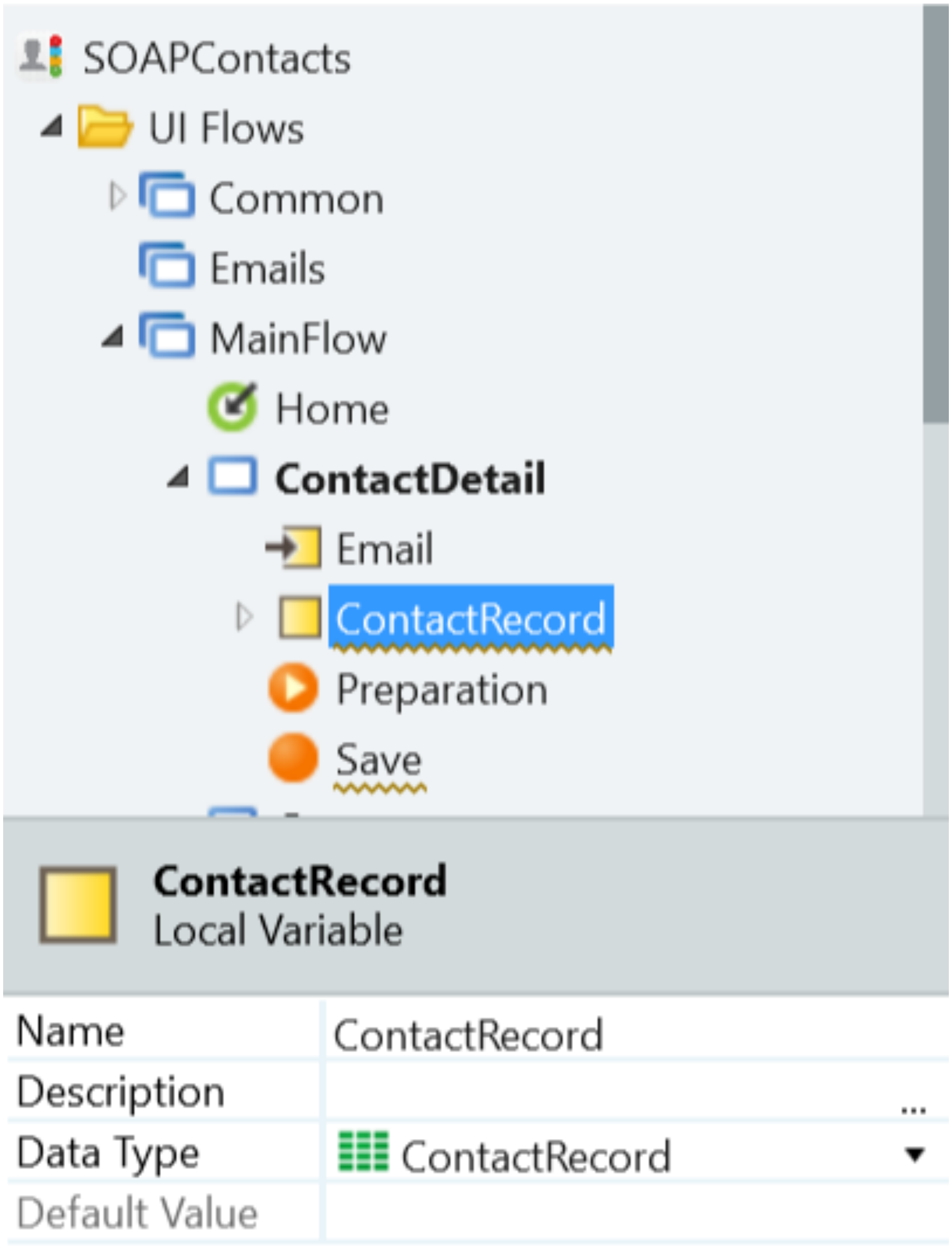
    b) Drag a **Form** widget and drop it in the MainContent area of the screen.



c) Right-click on the **ContactDetail** screen and select *Add Local Variable*.



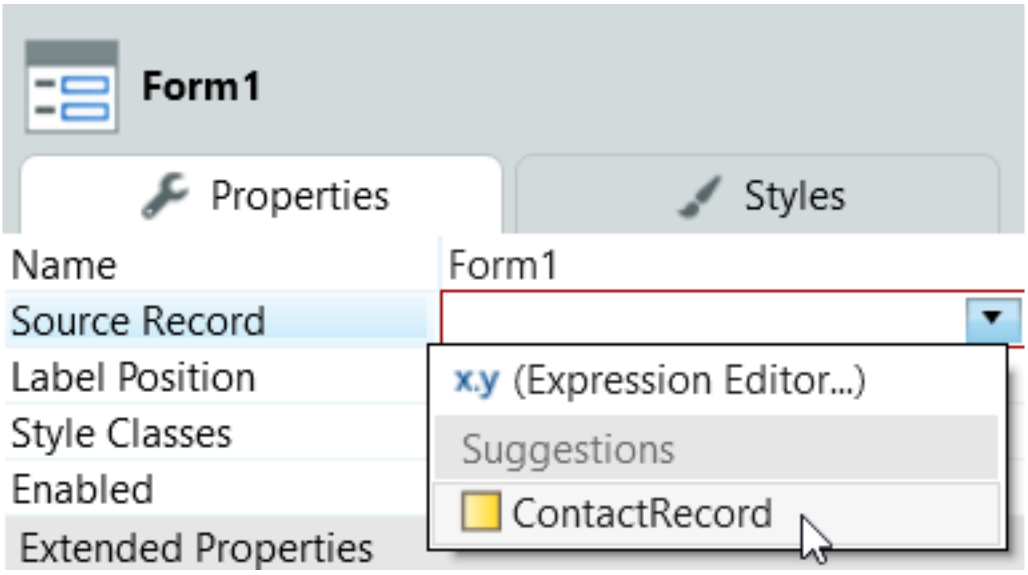
d) Set the Variable local variable **Name** to *ContactRecord*, and make sure the **Data Type** is set to *ContactRecord*.



ContactRecord
Local Variable

Name	ContactRecord
Description	
Data Type	ContactRecord
Default Value	

e) Select the Form on the screen and then set the **Source Record** property to the *ContactRecord* local variable.

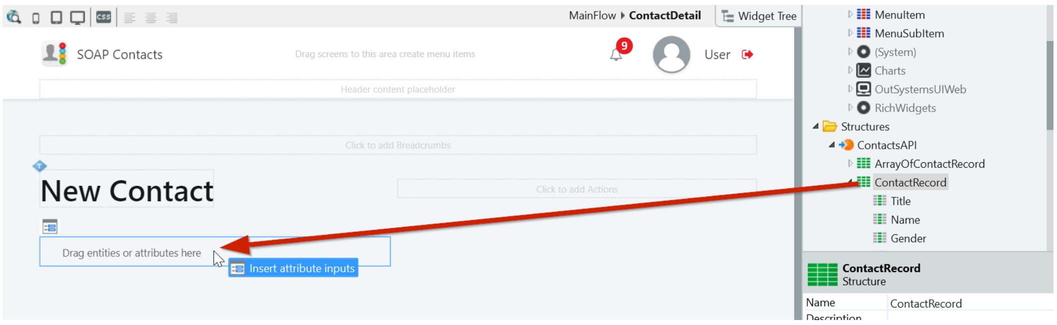


Form1

Properties | Styles

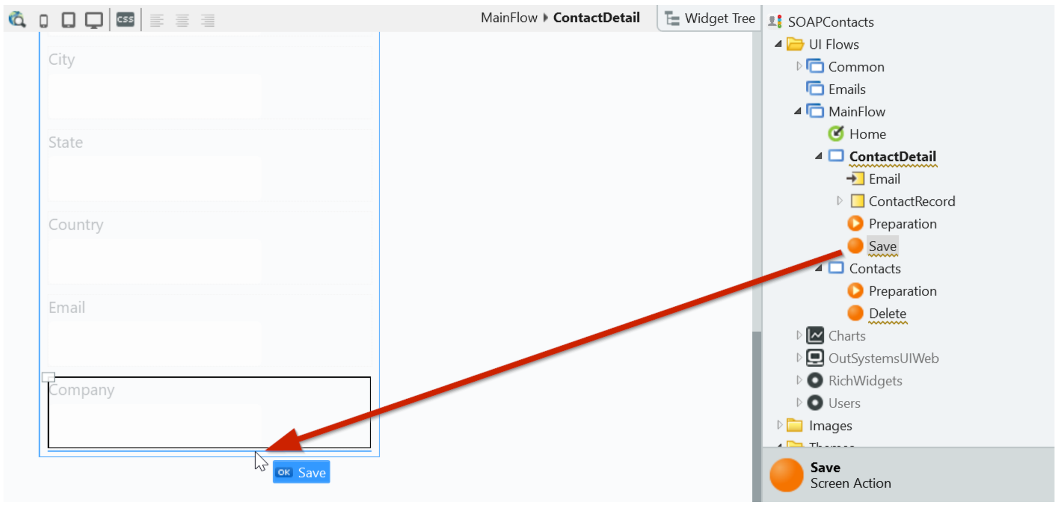
Name	Form1
Source Record	
Label Position	
Style Classes	
Enabled	
Extended Properties	

f) In the Data tab, drag the *ContactRecord* structure to the Form area.



NOTE: This drag and drop action will create the Label and input for each of the attributes. Each input will match the actual Data Type of the attribute. For instance, the Birthday input will display a Calendar in runtime. We won't go further in terms of customizing the form, and will focus on actually creating a Contact using the SOAP Web Service.

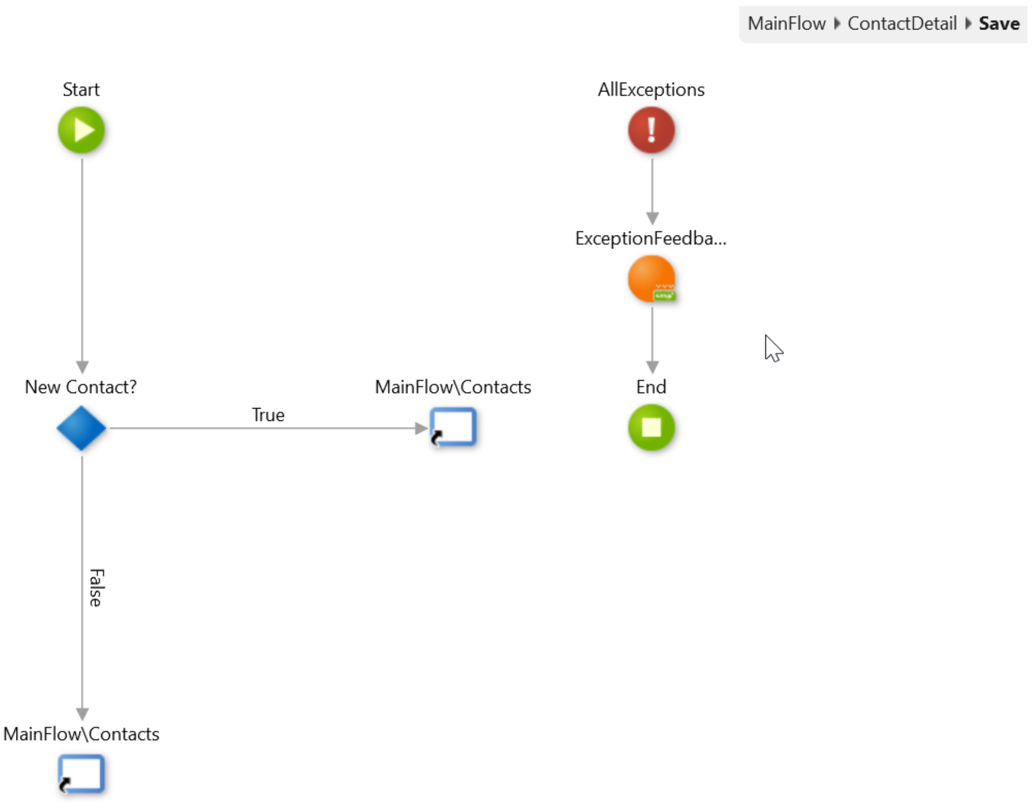
g) In the Interface tab, drag the **Save** screen action from the *ContactDetail* screen and drop it at the bottom of the Form.



NOTE: Dragging a Screen Action into the Screen will automatically create a Button, and will also set its label to the Screen Action name and its Destination to the Screen Action.

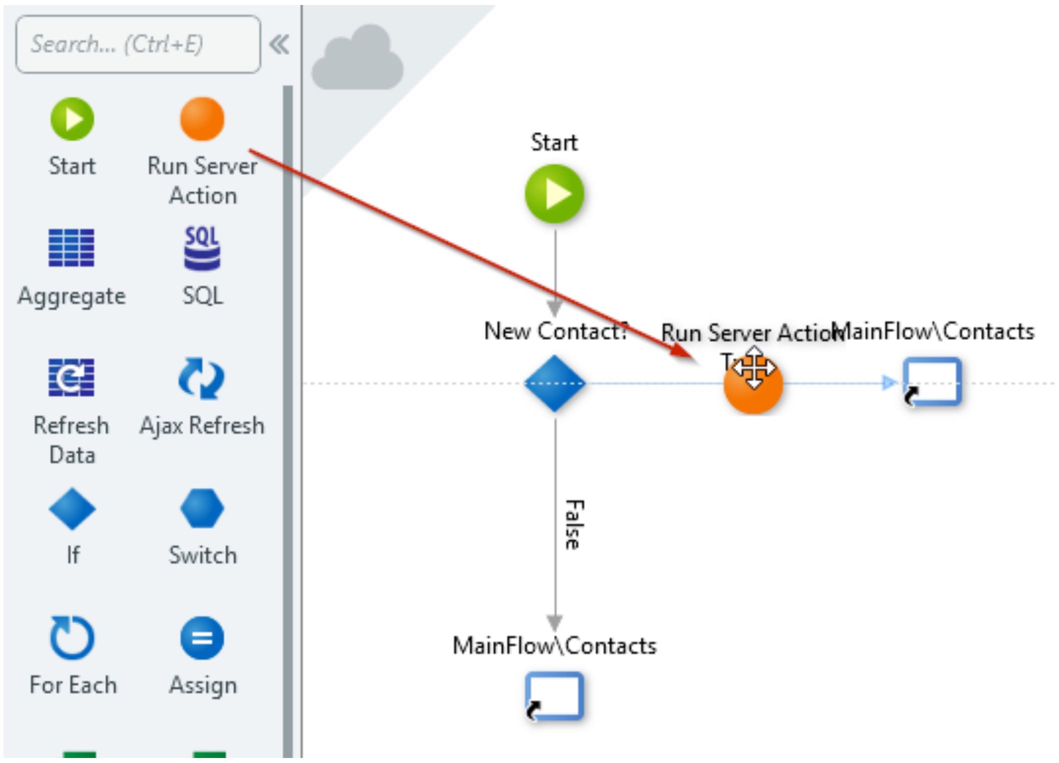
3) Create the logic inside the Save screen action that will invoke the SOAP Web Service method to create a new contact based on the inputs entered in the Form.

   a) Double-click the button to open the **Save** screen action, or open under the *ContactDetail* screen.

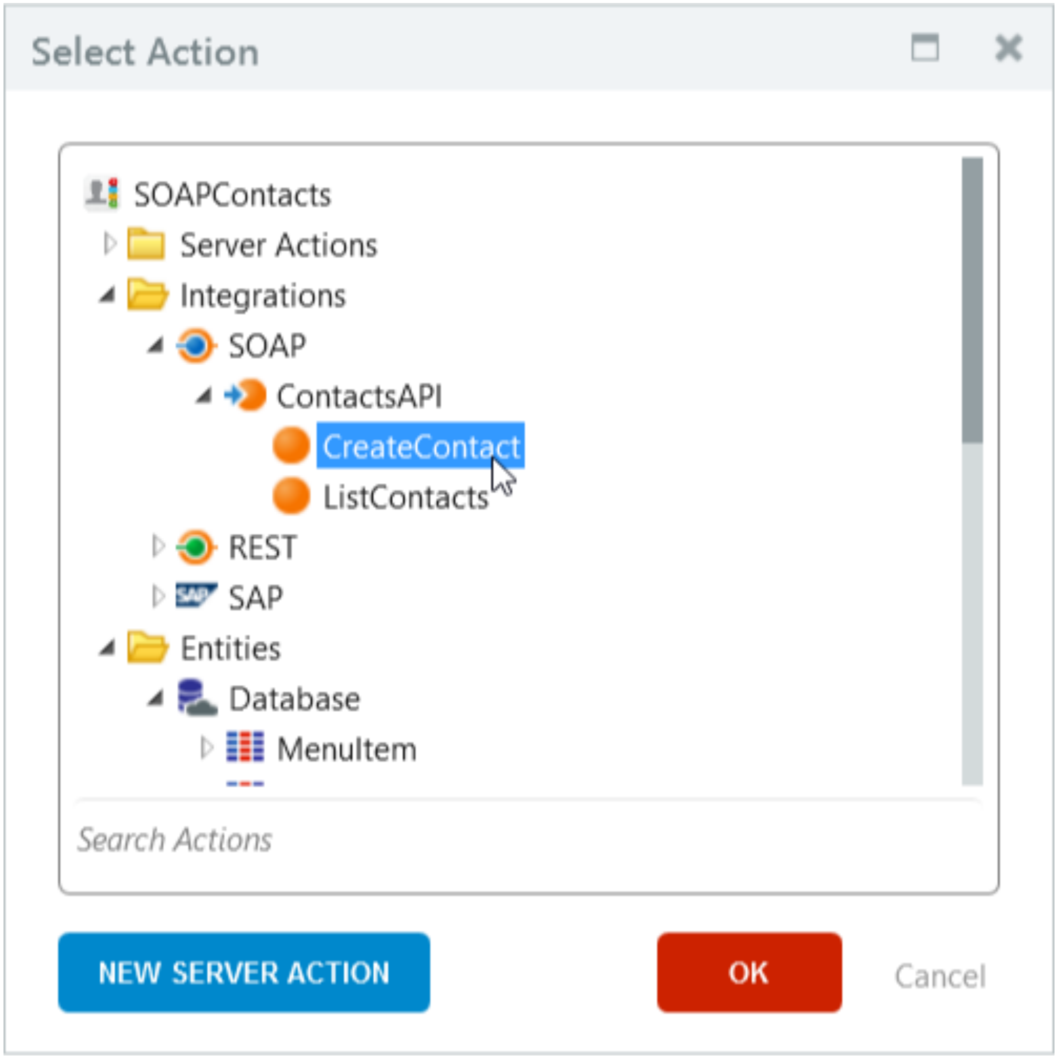


NOTE: This action already has some logic defined. The *New Contact?* It verifies if the Email input parameter of the screen is empty or not. When empty, it means that we will be adding a new contact. Later on, we will implement the logic regarding updating existing contacts.

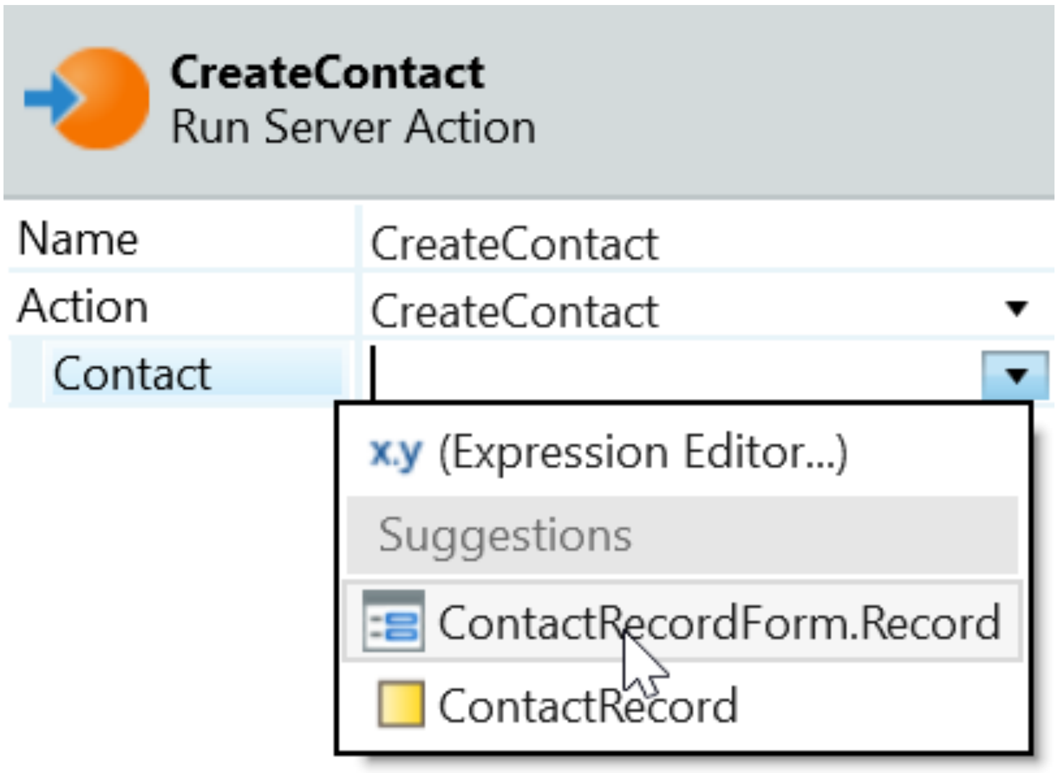
b) Drag a **Run Server Action** and drop it on the True branch.



c) In the **Select Action** dialog, select the *CreateContact* action under Integrations, SOAP, and *ContactsAPI*. Click OK.



d) Set the **Contact** parameter to *ContactRecordForm.Record*.



CreateContact
Run Server Action

Name	CreateContact
Action	CreateContact
Contact	

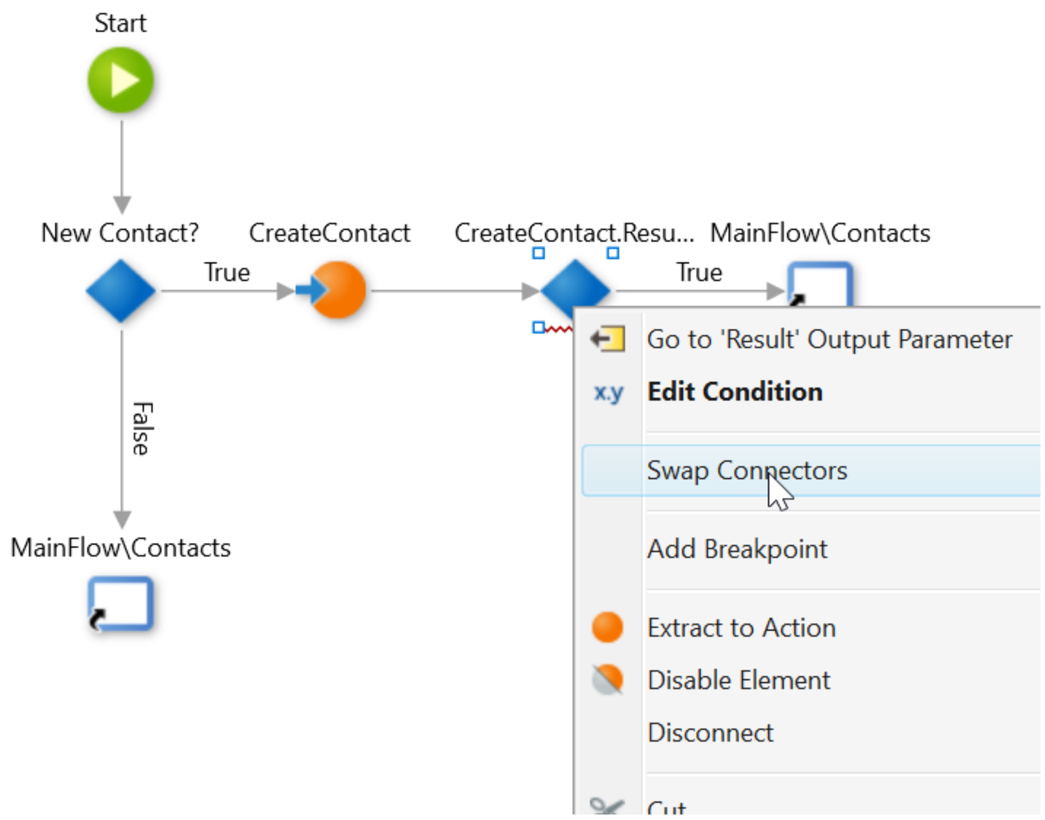
4) Validate if the Contact was successfully created. If not, show a feedback message with the error message.

a) Drag an **If** and drop it between *CreateContact* and *MainFlow\Contacts*.

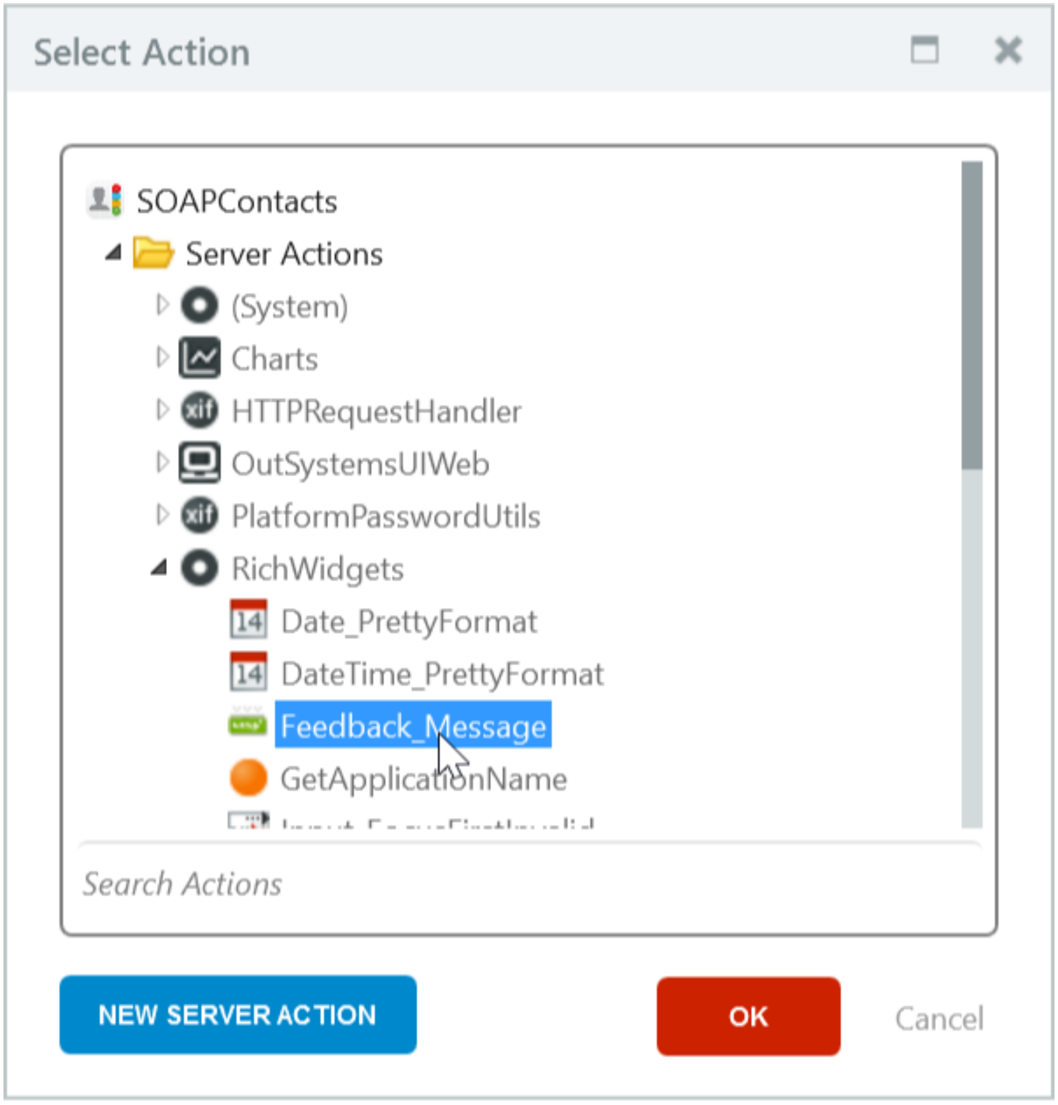
b) Set the **Condition** of the If to

```
CreateContact.Result.Success
```

c) Right-click the **If** and select *Swap Connectors*.



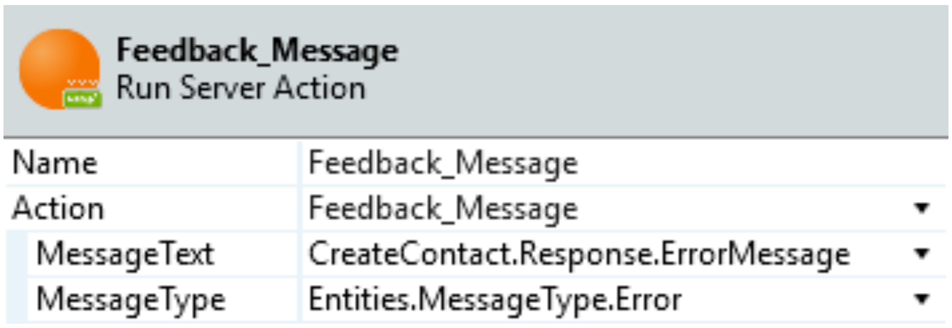
d) Drag a **Run Server Action** and drop it below the If, then select *Feedback_Message*.



e) Create the *False* branch connector from the If to the Feedback_Message.

f) Click on the Feedback_Message and set its parameters to:

MessageText: CreateContact.Result.ErrorMessage
MessageType: Entities.MessageType.Error

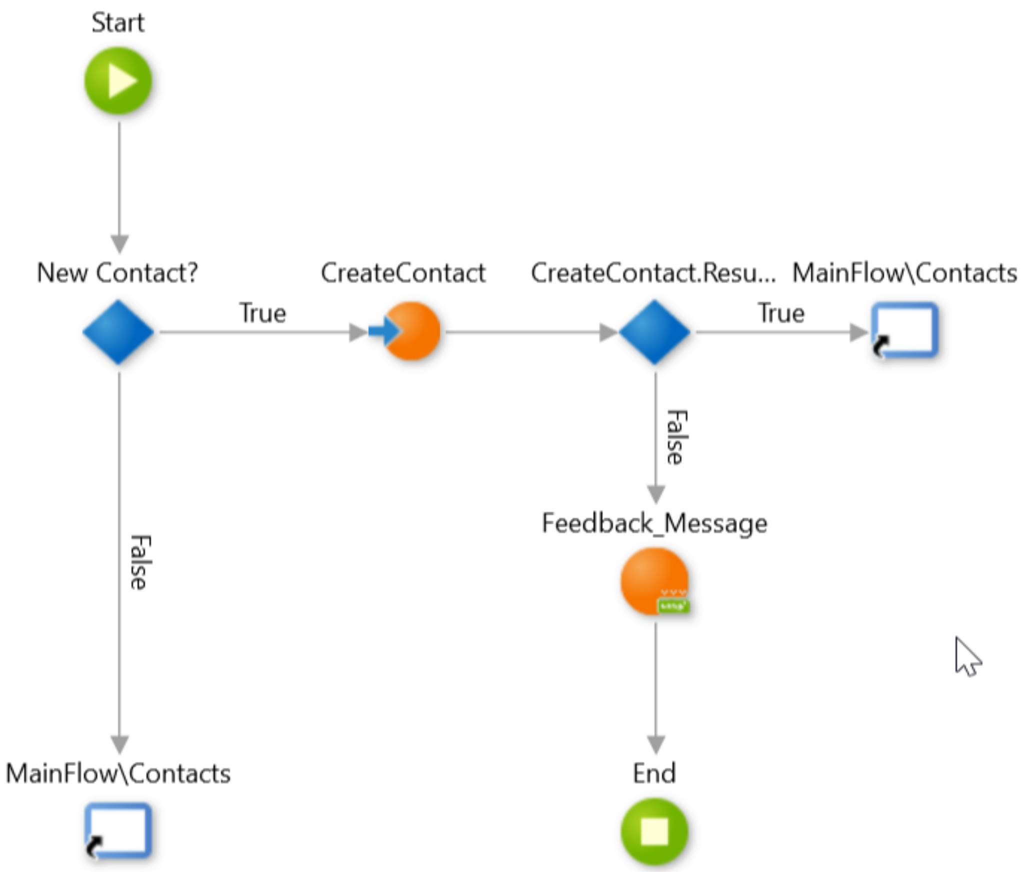


Feedback_Message
Run Server Action

Name	Feedback_Message
Action	Feedback_Message
MessageText	CreateContact.Response.ErrorMessage
MessageType	Entities.MessageType.Error

g) Drag an **End** and drop it below *Feedback_Message*. Connect both.

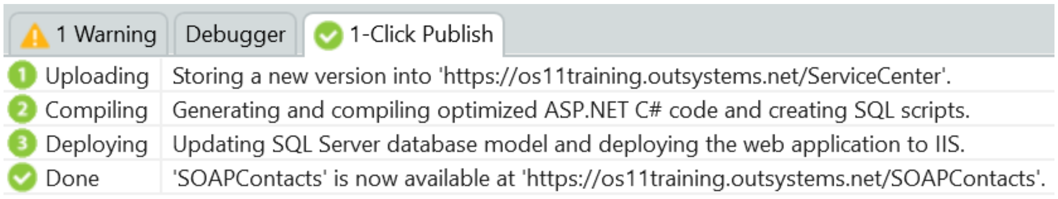
h) The flow should look like this:



5) Publish the application using 1-Click Publish button and verify that the publish completed successfully in the 1-Click Publish Tab.

a) Click on the **1-Click Publish** button to publish the module to the server.

b) Verify in the 1-Click Publish tab that the publishing process was successful.



Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.
Compiling	Generating and compiling optimized ASP.NET C# code and creating SQL scripts.
Deploying	Updating SQL Server database model and deploying the web application to IIS.
Done	'SOAPContacts' is now available at 'https://os11training.outsystems.net/SOAPContacts'.

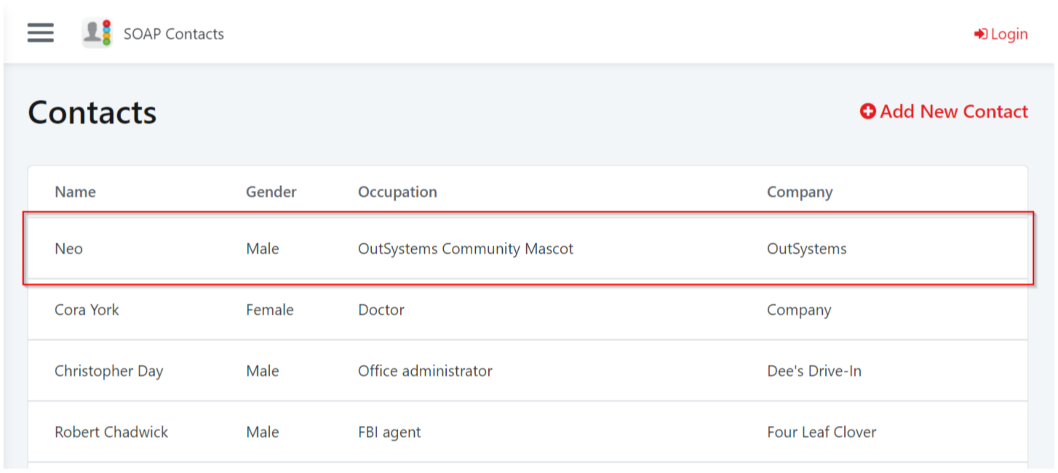
c) Preview the app in browser by clicking on the **Open in Browser** button.

d) Click the **Add New Contact** link at the top right of the screen.

e) Fill in the form with some Contact information like, for example:

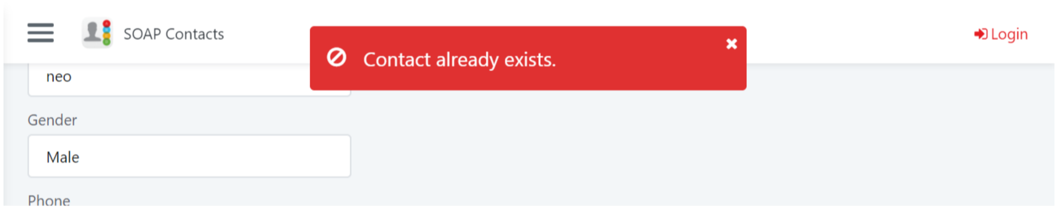
Name: Neo
Gender: Male
Birthday: 2000-01-01
Occupation: OutSystems Community Mascot
Email: neo@outsystems.com
Company: OutSystems

f) Click *Save* and you should be redirected to the Contacts screen and see the new contact.



Name	Gender	Occupation	Company
Neo	Male	OutSystems Community Mascot	OutSystems
Cora York	Female	Doctor	Company
Christopher Day	Male	Office administrator	Dee's Drive-In
Robert Chadwick	Male	FBI agent	Four Leaf Clover

g) Try to enter another contact with the same Email. Upon clicking Save, an error message should appear.



Contact already exists.

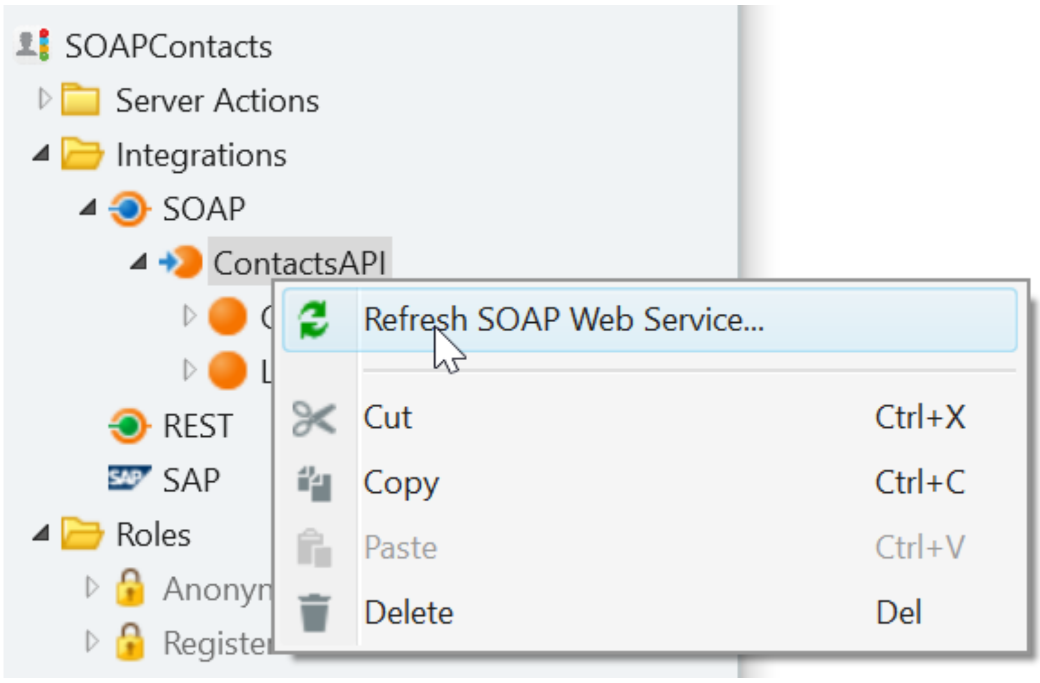
Consume a SOAP Web Service Method to Update a Record

In this section we will start working towards having the ability to update existing contacts. First we will start by adding a new method to our Web Service that will provide this functionality. Later on, we will modify our user interface and logic to allow users to update existing contacts.

To enable updating contacts an extra method will also be needed besides the one to update. It's the one to retrieve a single Contact based on a given Email address. It will allow to retrieve the contact information in the Preparation of the ContactDetail screen, and then on the Save action update that contact.

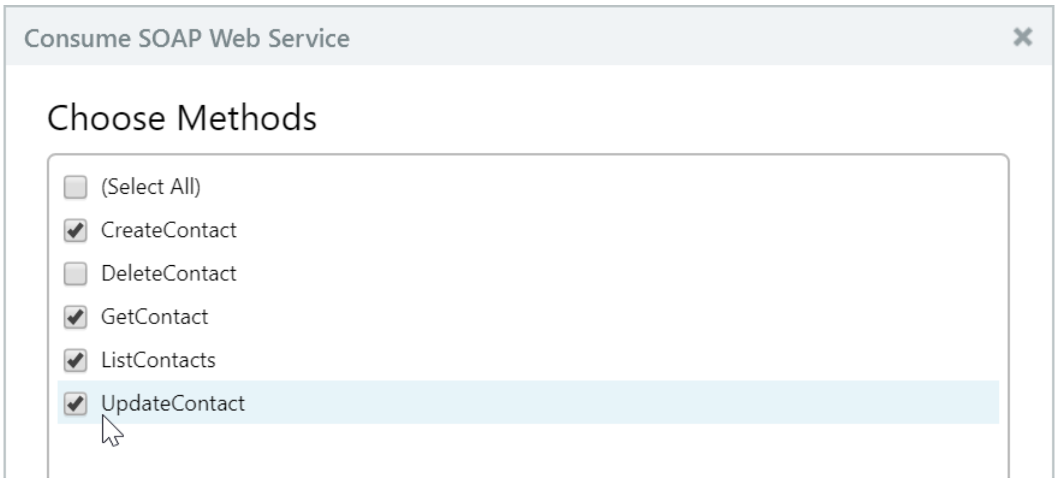
1) Consume the method in the external SOAP Web Service that, given an Email, will update the changed attributes of the contact.

   a) Switch to the Logic tab, right-click the **ContactsAPI** and and select *Refresh SOAP Web Service...*



   b) If a dialog appears asking whether to proceed, click *Yes*.

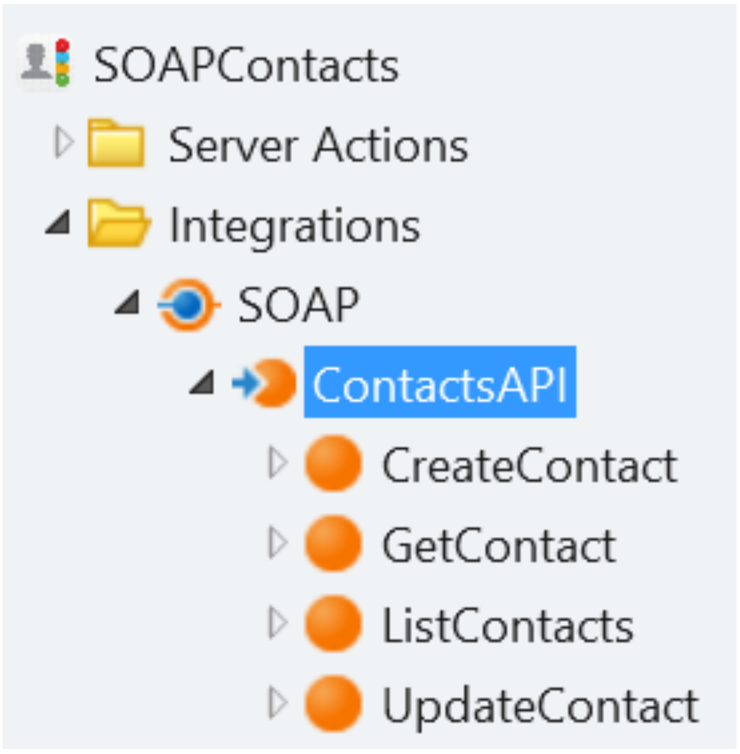
c) In the next dialog, *ListContacts* and *CreateContact* should be already selected. Select also **GetContact** and **UpdateContact** and press *Finish*.


Consume SOAP Web Service

Choose Methods

- [ ] (Select All)
- [x] CreateContact
- [ ] DeleteContact
- [x] GetContact
- [x] ListContacts
- [x] UpdateContact

d) Change the name of the SOAP Integration from *v1* to **ContactsAPI**.



NOTE: This step is not mandatory, but it allows developers to have custom or more familiar names in their integrations.

Update Contact User Interface

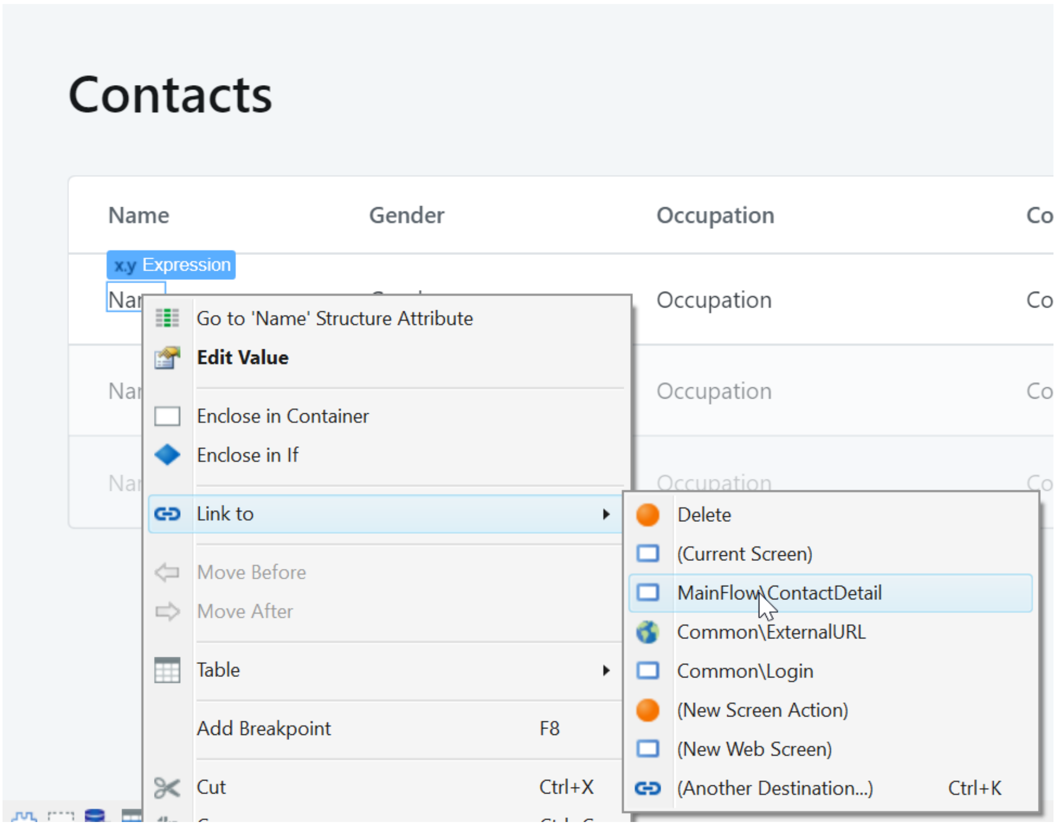
Now that we have all the required integration methods completed, we will change our user interface and logic to allow end users to update contacts.

First we will link the Contacts and ContactDetail screens through another link to update a given record. Then, on the ContactDetail screen the information of the contact being edited will be fetched from the external system, and upon saving the information, the new data will be sent and stored in the external system.

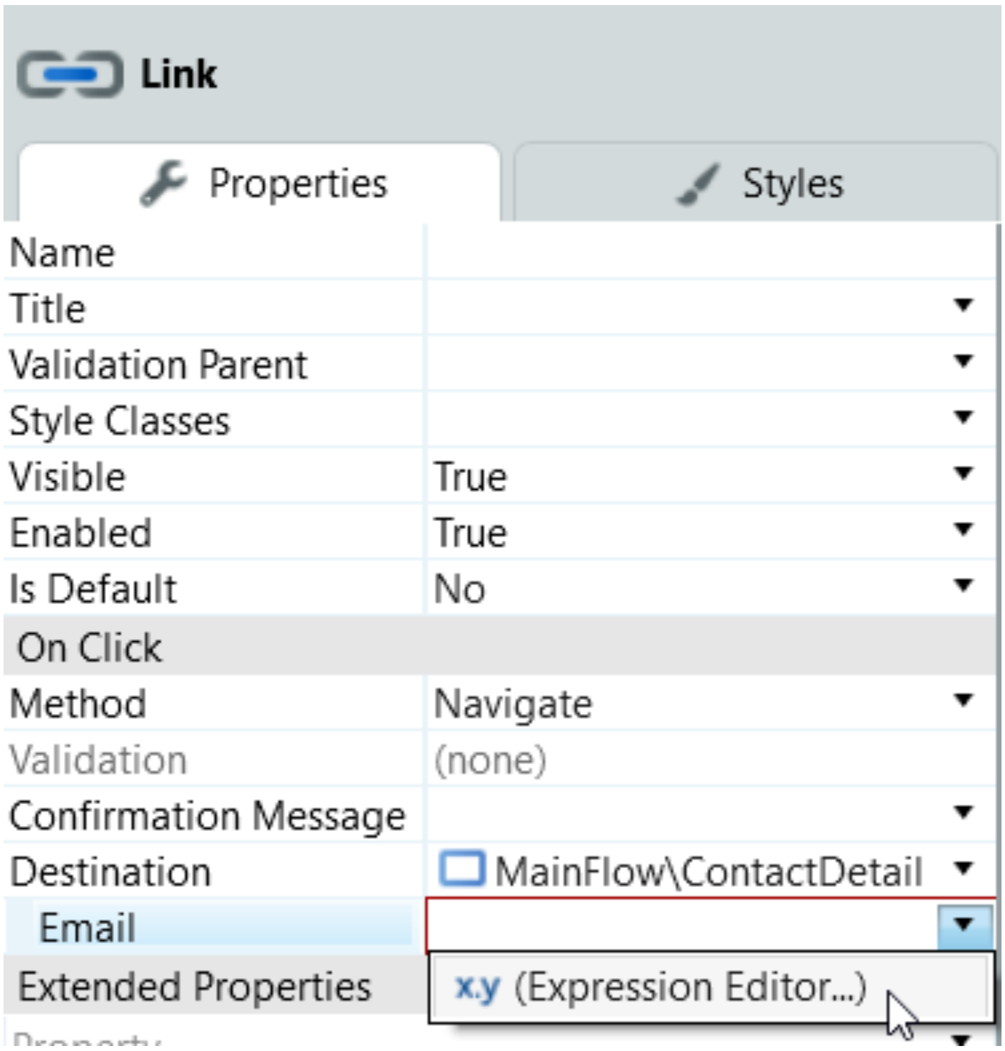
1) Create a link from the Contacts screen to the ContactDetail screen for each row (record) in the Table Records.

   a) Open the **Contacts** screen from the Interface tab.

   b) Right-click the **Name** expression, and select *Link to* > *MainFlow\ContactDetail*.



c) In the Properties Pane, go to the **Email** property and select *(Expression Editor...)* in the drop down.

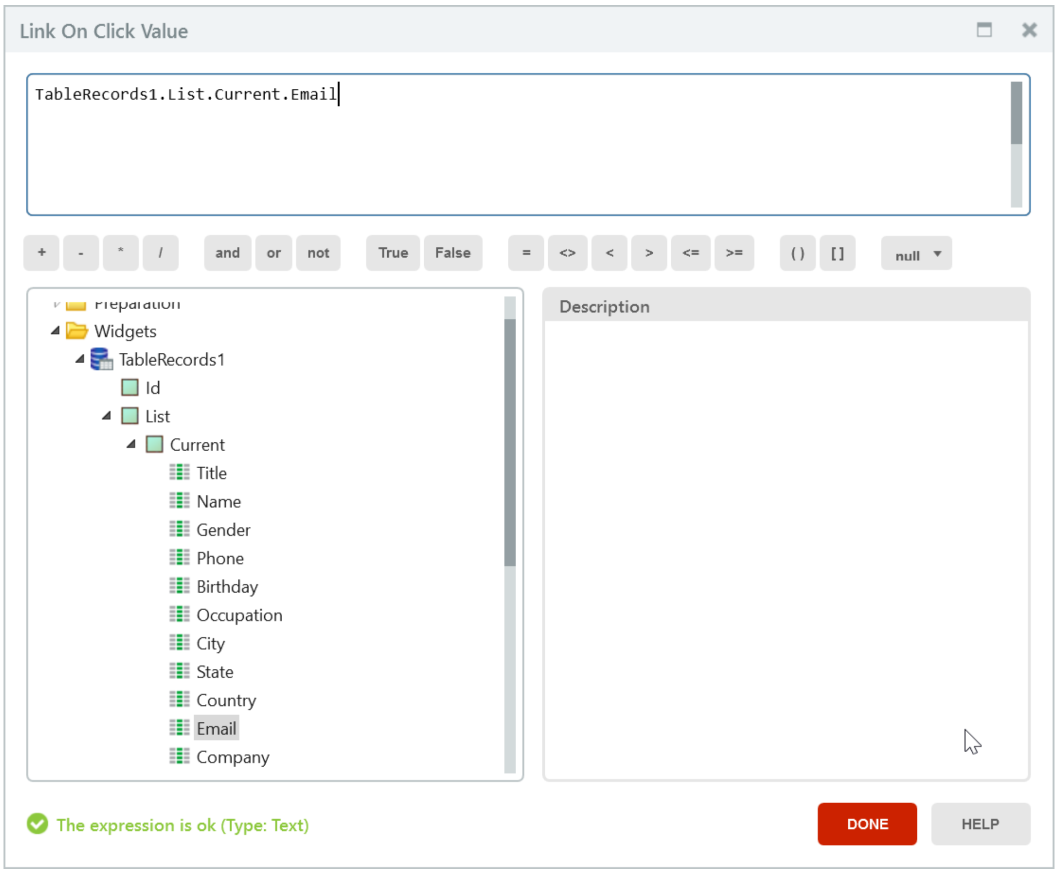


Link

Name	
Title	
Validation Parent	
Style Classes	
Visible	True
Enabled	True
Is Default	No
On Click	
Method	Navigate
Validation	(none)
Confirmation Message	
Destination	MainFlow\ContactDetail
Email	
Extended Properties	

d) Write the following expression:

```
TableRecords1.List.Current.Email
```



```
TableRecords1.List.Current.Email
```

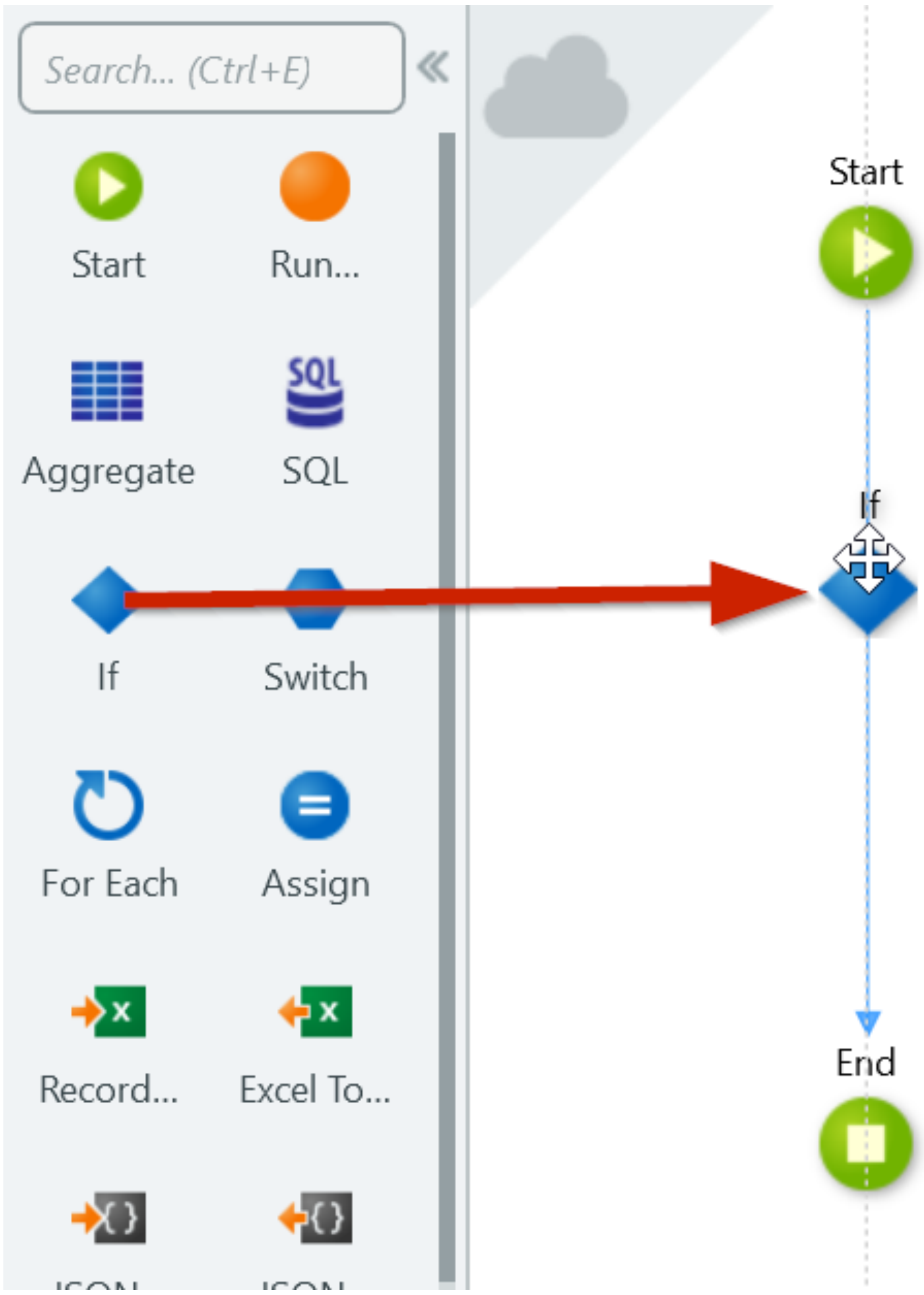
The expression is ok (Type: Text)

e) Click **Done** to close the Expression Editor.

2) Change the Preparation of the ContactDetail screen to fetch a single Contact from the external system if the Email input parameter is not empty.

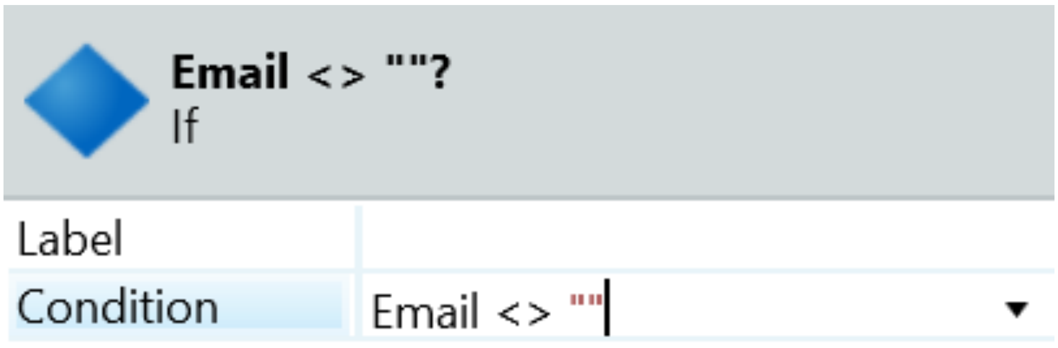
a) In the Interface tab, open the Preparation of the **ContactDetail** screen.

b) Drag an **If** and drop it between the Start and End.



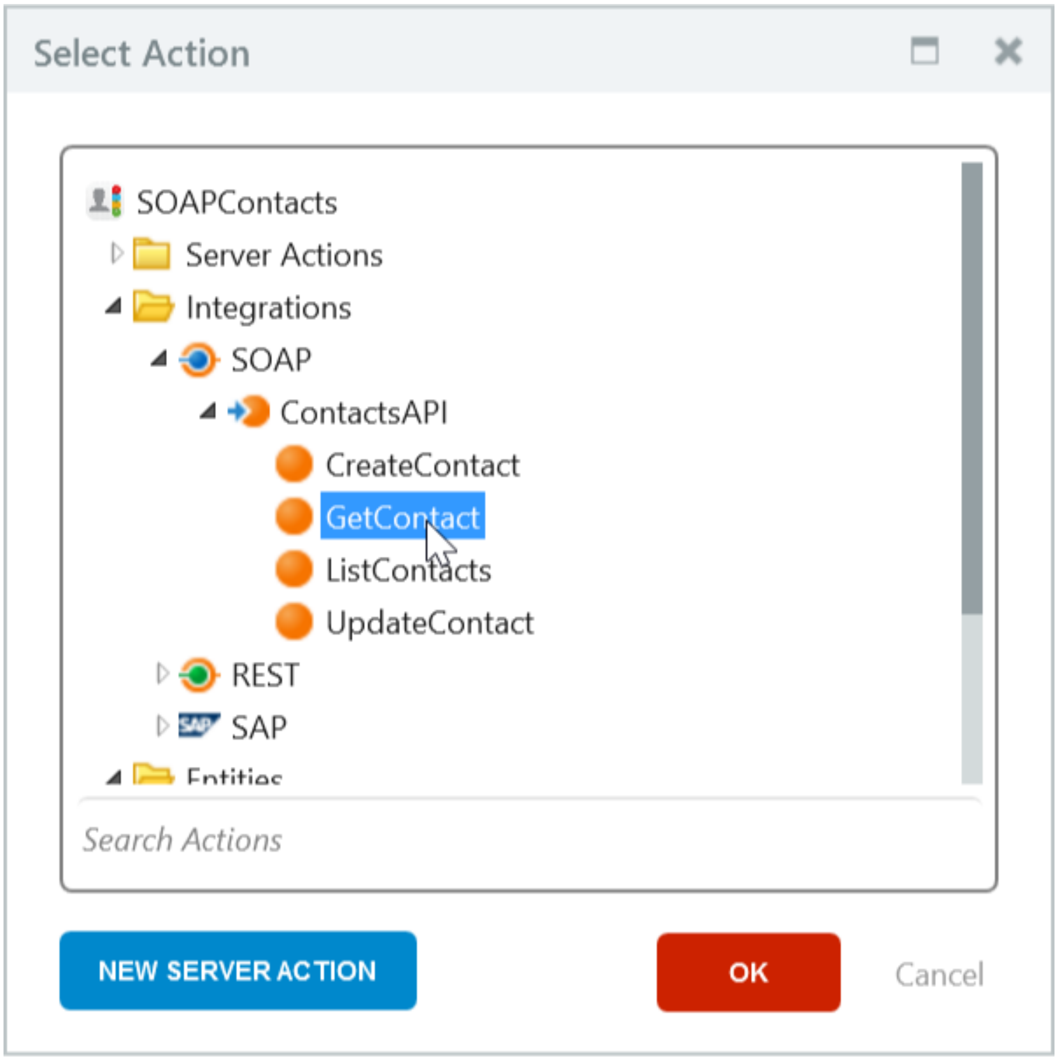
c) Set the Condition property of the If to

```
Email <> ""
```

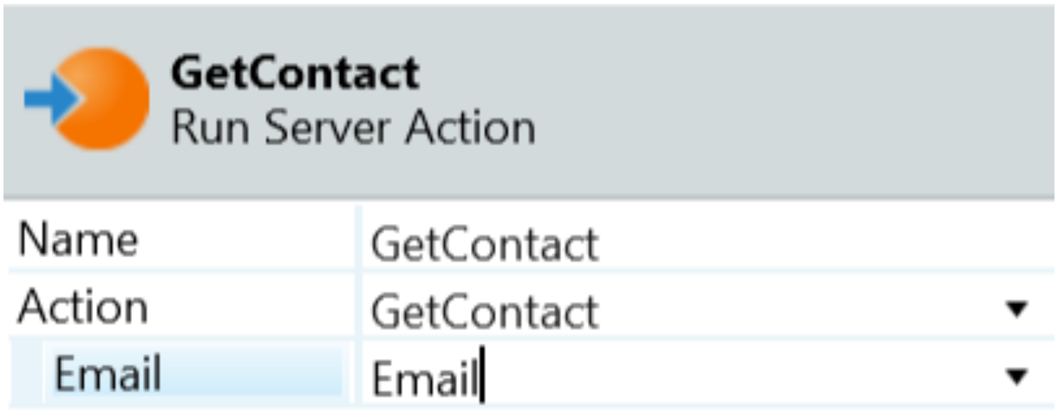

Email <> ""?
If

Label
Condition Email <> ""

d) Drag a **Run Server Action** and drop it to the right of the If, then in the Select Action dialog, choose the GetContact from the ContactsAPI SOAP Integration.



e) Set the Email parameter with Email, the input parameter of the Screen.



GetContact
Run Server Action

Name	GetContact
Action	GetContact
Email	Email

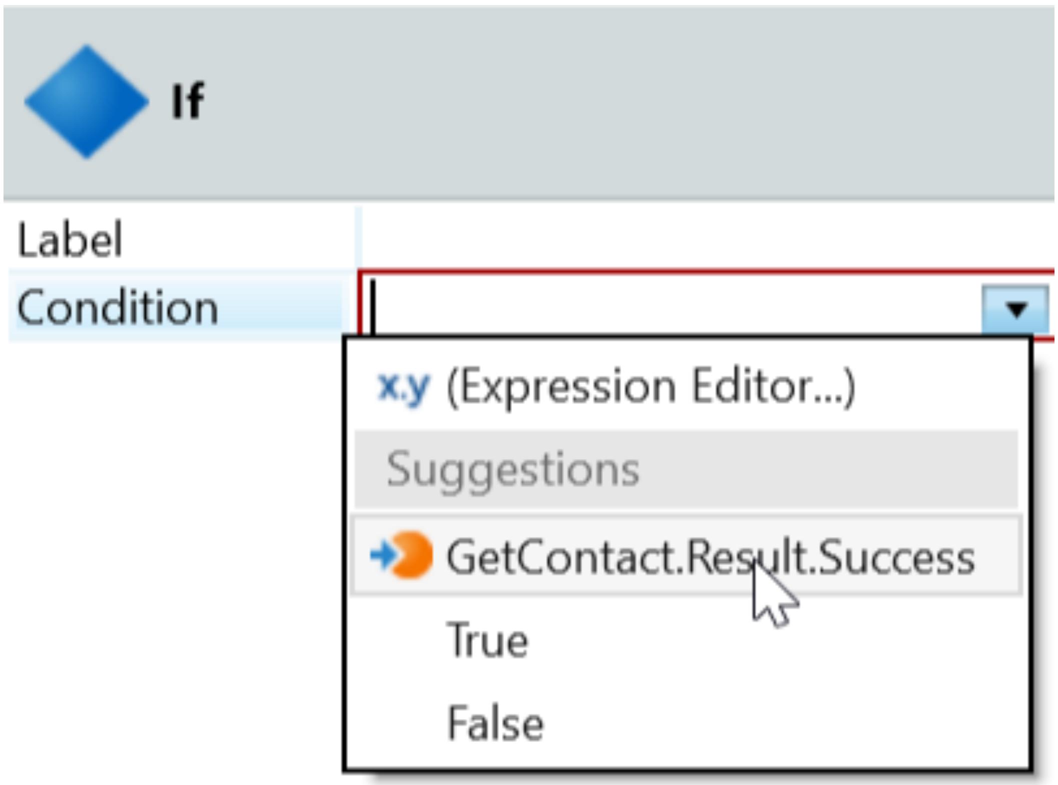
f) Create the **True** branch connector from the If to the *GetContact* action created above.

3) Validate if the operation to get the contact information from the external system was successful.

a) Drag another **If** and drop it to the right of the GetContact, then create a connector from the *GetContact* to the new *If*.

b) Set the **Condition** of the new *If* to

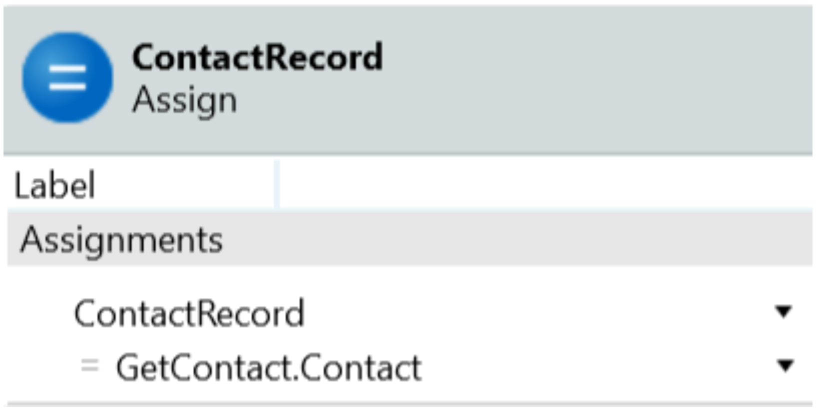
```
GetContact.Result.Success
```



c) Drag an Assign and drop it to the right of the If and connect the If to it to create the *True* branch.

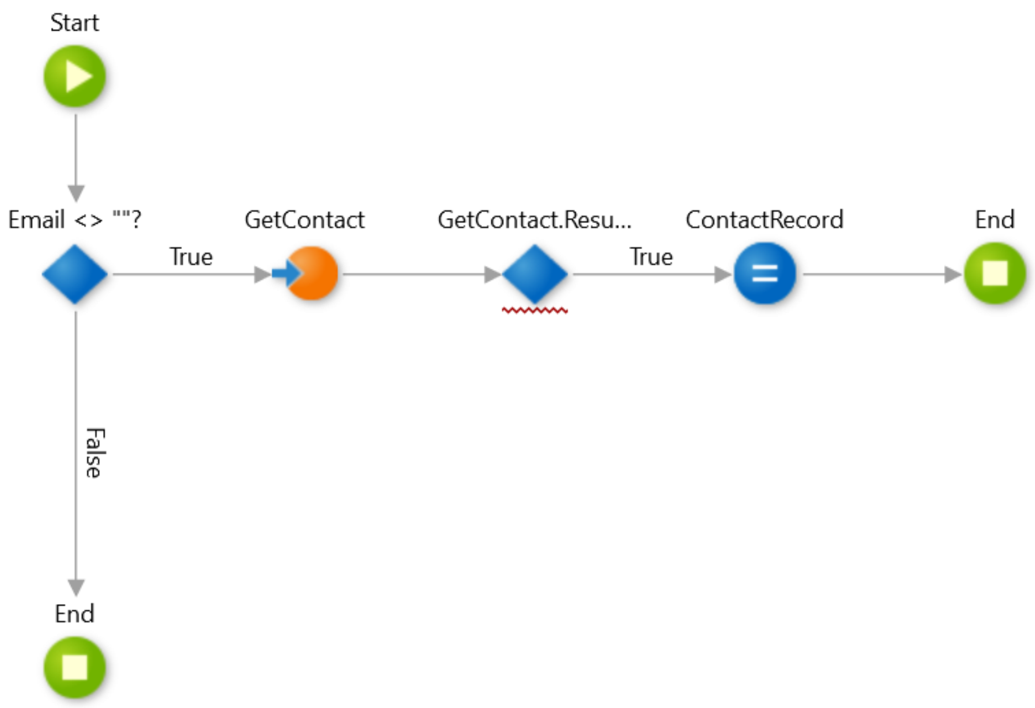
d) Define the following assignment

```
ContactRecord = GetContact.Contact
```



e) Drag an **End** and drop it to the right of the assign and then connect both.

f) The Preparation flow should look like this:

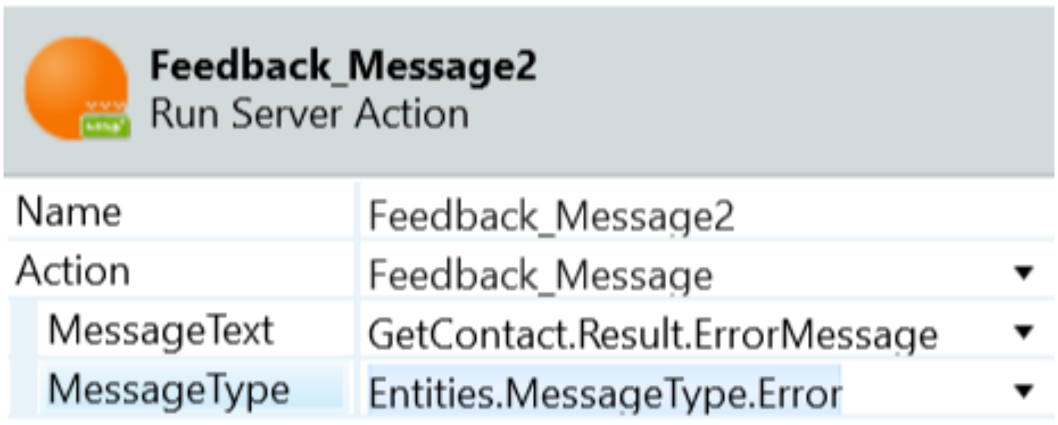


g) Drag a **Run Server Action** and drop it below the If to validate the success of the call to *GetContact*. And, in the **Select Action** dialog, choose *Feedback_Message*.

h) With the *Feedback_Message2* action created, connect the If to it to create the *False* branch.

i) Set the parameters of the *Feedback_Message2* as follows:

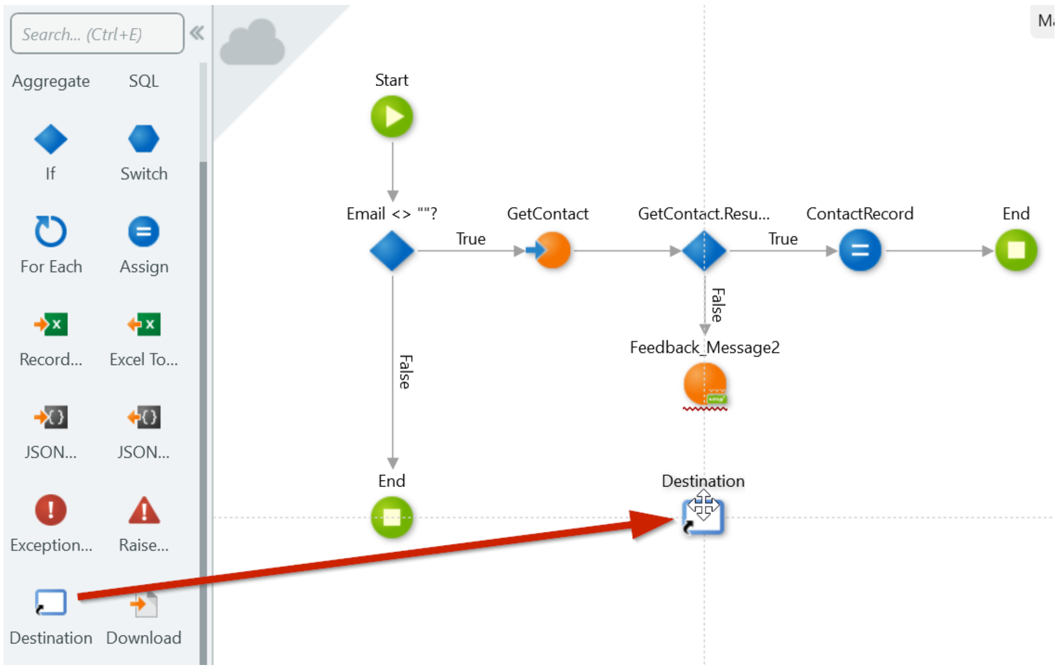
MessageText: GetContact.Result.ErrorMessage
MessageType: Entities.MessageType.Error



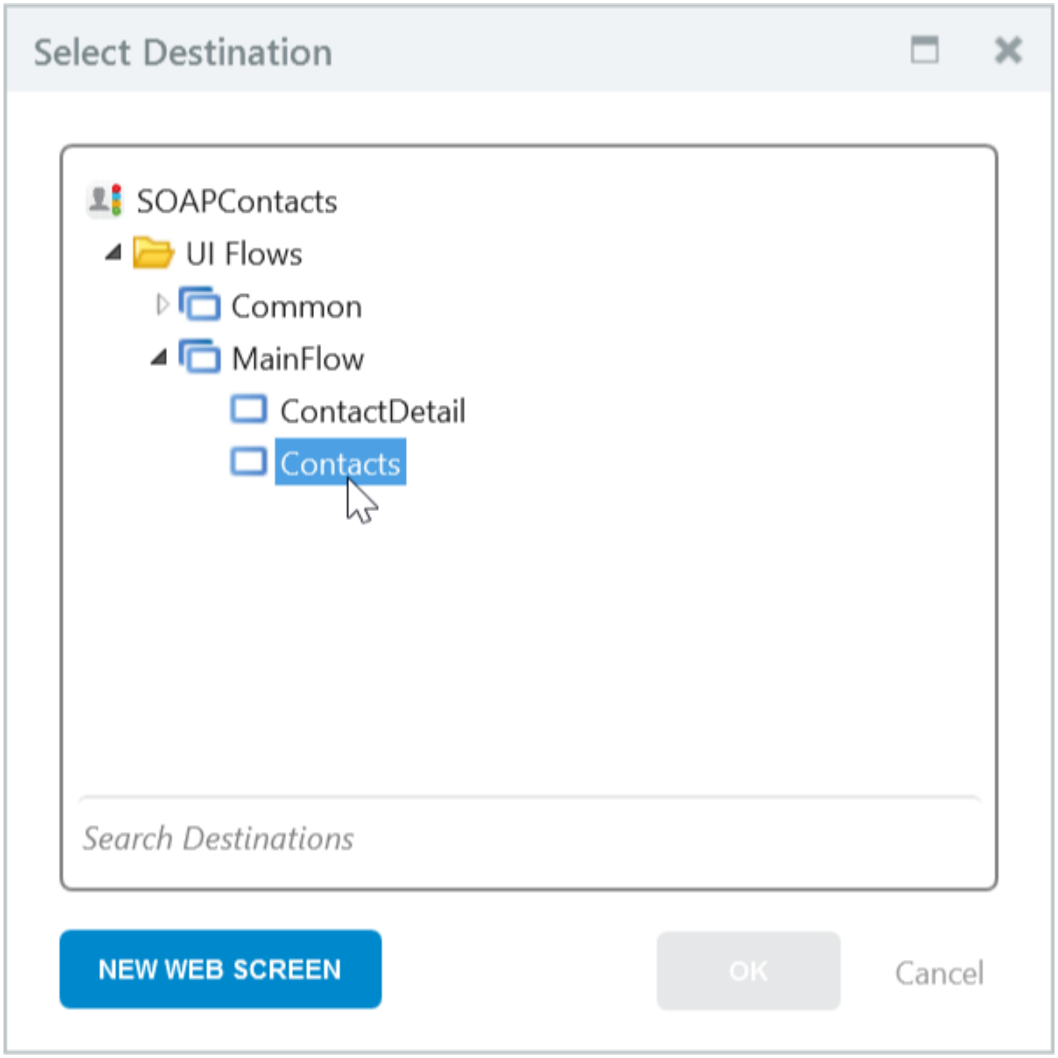
Feedback_Message2
Run Server Action

Name	Feedback_Message2
Action	Feedback_Message
MessageText	GetContact.Result.ErrorMessage
MessageType	Entities.MessageType.Error

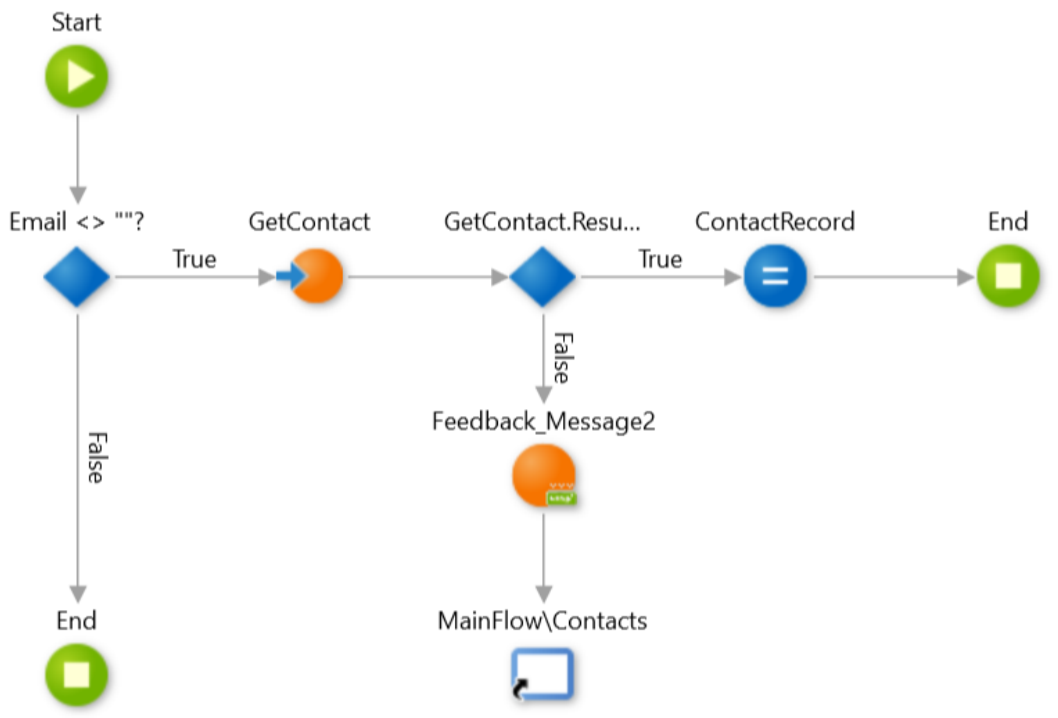
j) Drag a **Destination** and drop it below the *Feedback_Message2* action.



k) In the Select Destination dialog choose the *Contacts* screen and click OK.



l) Create the connector from the *Feedback_Message2* action to the Destination node. The flow should now look like this:

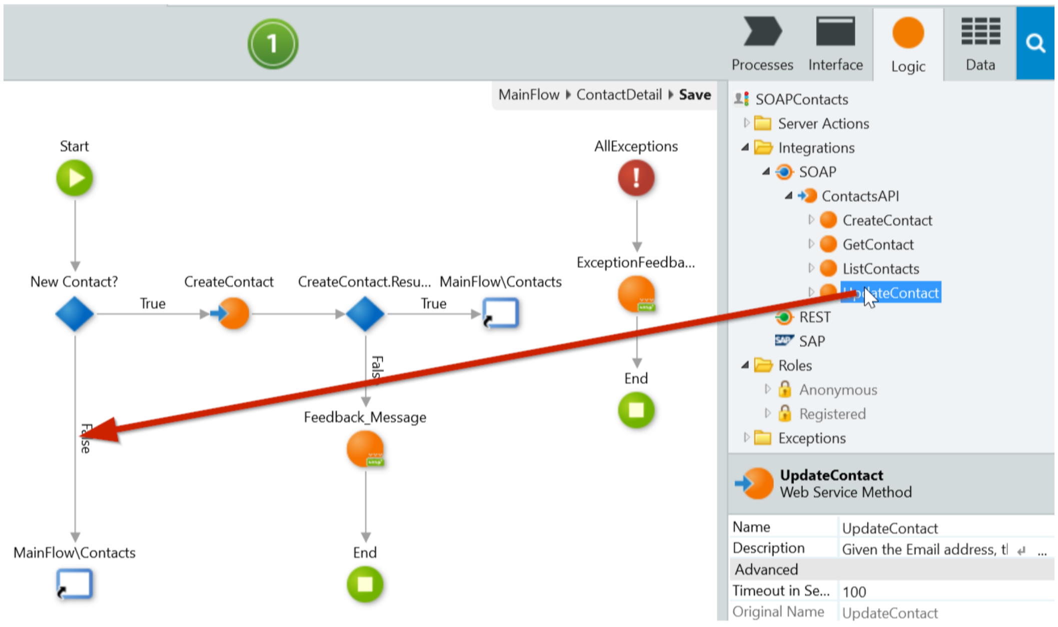


NOTE: The logic defined in the previous steps will ensure that when we navigate from the Contacts screen to the ContactDetail screen (i.e. the Email input parameter has a value), we will retrieve the Contact information from the external system, and (on success) store it in a local variable. This local variable is then used to populate the form input values on the ContactDetail screen. If the operation to retrieve the contact information fails, we show a feedback message, and redirect the user to the Contacts screen.

4) Change the Save screen action to invoke the UpdateContact method of the ContactsAPI, to update a contact when the Email input parameter is not empty.

a) In the Interface tab, open the **Save** screen action of the *ContactDetail* screen.

b) In the Logic tab, drag the *UpdateContact* method and drop it on the *False* branch

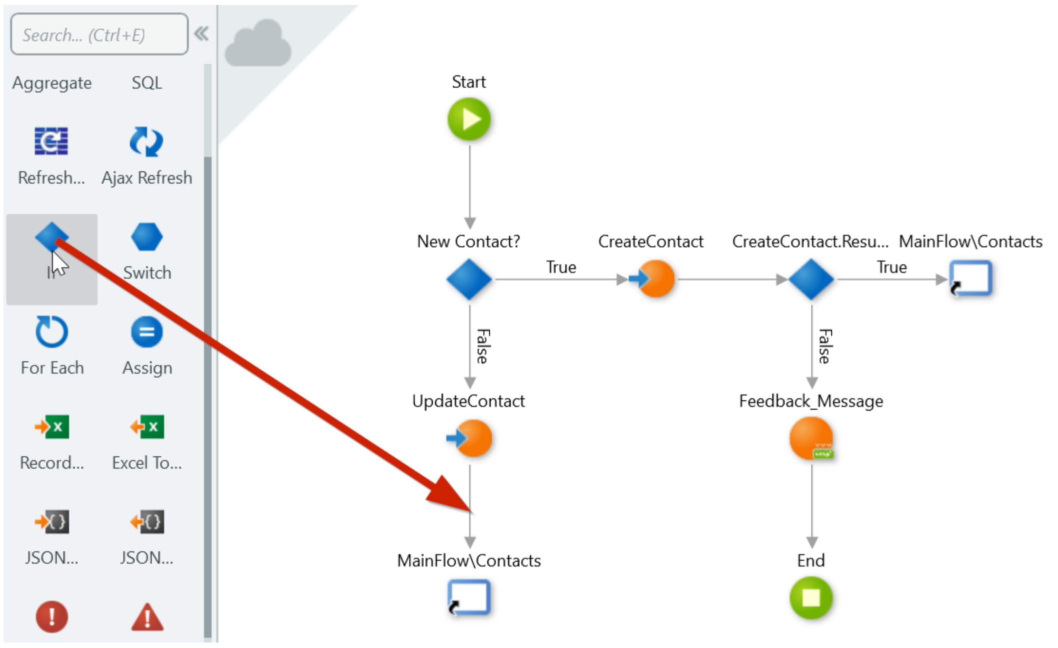


Name	UpdateContact
Description	Given the Email address, t
Advanced	
Timeout in Se...	100
Original Name	UpdateContact

c) Set the Contact parameter to:

```
ContactRecordForm.Record
```

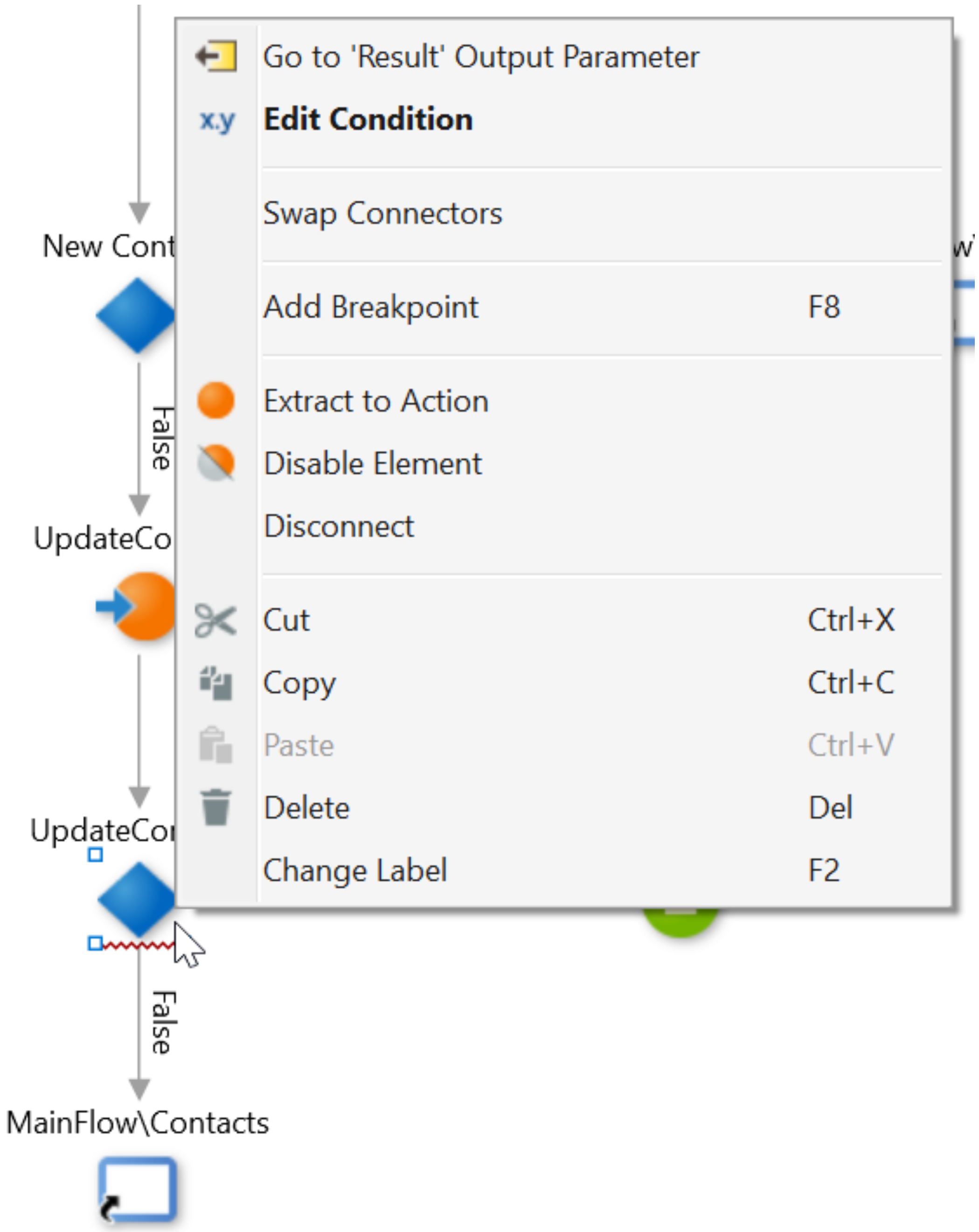
d) Drag an **If** and drop it between the *UpdateContact* and *MainFlow\Contacts*.



e) Set the Condition of the If to:

```
UpdateContact.Result.Success
```

f) Right-click the **If** and select *Swap Connectors*.

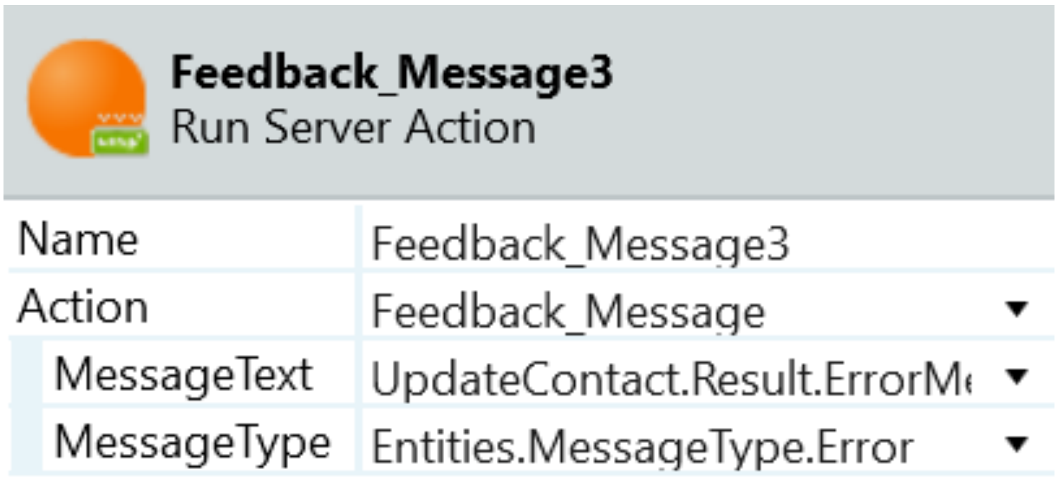


g) Drag a **Run Server Action** and drop it to the right of the If, then in the Select Action dialog choose *Feedback_Message*.

h) Create the False branch connector from the If to the new Feedback_Message, i.e., *Feedback_Message3*.

i) Set the parameters of the *Feedback_Message3* to:

MessageText: UpdateContact.Result.ErrorMessage
MessageType: Entities.MessageType.Error

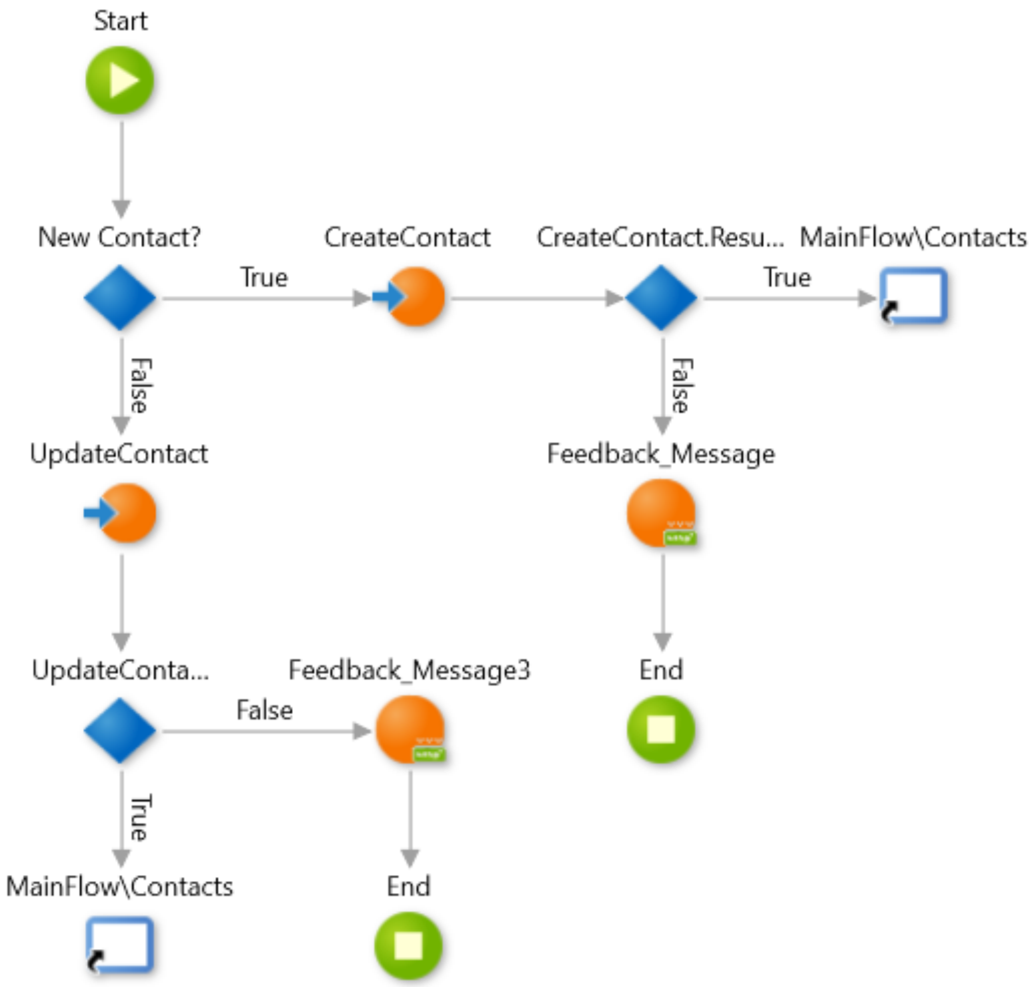


Feedback_Message3
Run Server Action

Name	Feedback_Message3
Action	Feedback_Message
MessageText	UpdateContact.Result.ErrorM
MessageType	Entities.MessageType.Error

j) Drag an **End** and drop it below the Feedback_Message3 created above, then create the connector between both.

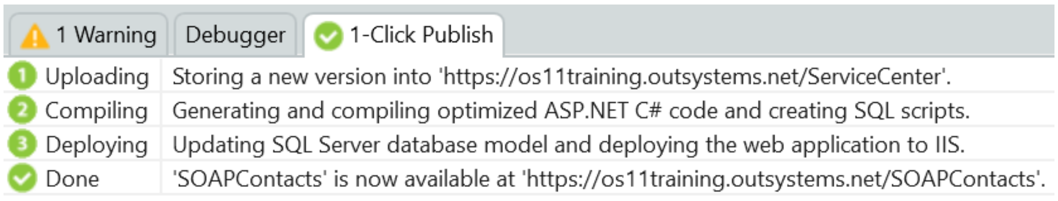
k) The Save action flow should look like this:



5) Publish the application using 1-Click Publish button and verify that the publish completed successfully in the 1-Click Publish Tab.

   a) Click on the **1-Click Publish** button to publish the module to the server.

   b) Verify in the 1-Click Publish tab that the publishing process was successful.



Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.
Compiling	Generating and compiling optimized ASP.NET C# code and creating SQL scripts.
Deploying	Updating SQL Server database model and deploying the web application to IIS.
Done	'SOAPContacts' is now available at 'https://os11training.outsystems.net/SOAPContacts'.

   c) Preview the app in browser by clicking on the **Open in Browser** button.

   d) In the list, click on the **name of the contact you created**.

   e) Change some of the contact data in the form then click **Save**.

   f) After saving you should be redirected to the Contacts screen, and see the changes on the contact.

End Lab

In this lab, we integrated with a SOAP Web Service, namely methods to create and update Contacts on the external system.

To accomplish that, we have used the OutSystems visual interface to add a few extra SOAP methods that enable the creation of a new contact, and updating a new contact.

Once the new SOAP methods were added to the integration, the user interface and underlying logic was changed to allow creating new contacts and changing existing contacts.

At the end of this exercise you should be able to integrate with a simple SOAP Web Service to write and update data in an OutSystems application.