Data Sync Exercise

Table of Contents

Outline.......... **2**

Resources 2

Scenario 2

How-To.......... **4**

Getting Started 4

Read-Only Pattern 6

Read-Write Pattern 8

Configure Sync 10

Test 11

Edit Customers 11

Outline

In this exercise, we will focus on creating a synchronization procedure to enable you to sync data from the server to the local storage.

Two synchronization patterns will be implemented, using Service Studio accelerators:

- Read-Only
- Read-Write

This will enable us to have a copy of the data from the server for read-only purposes, while another Entity will be allow editing data.

Resources

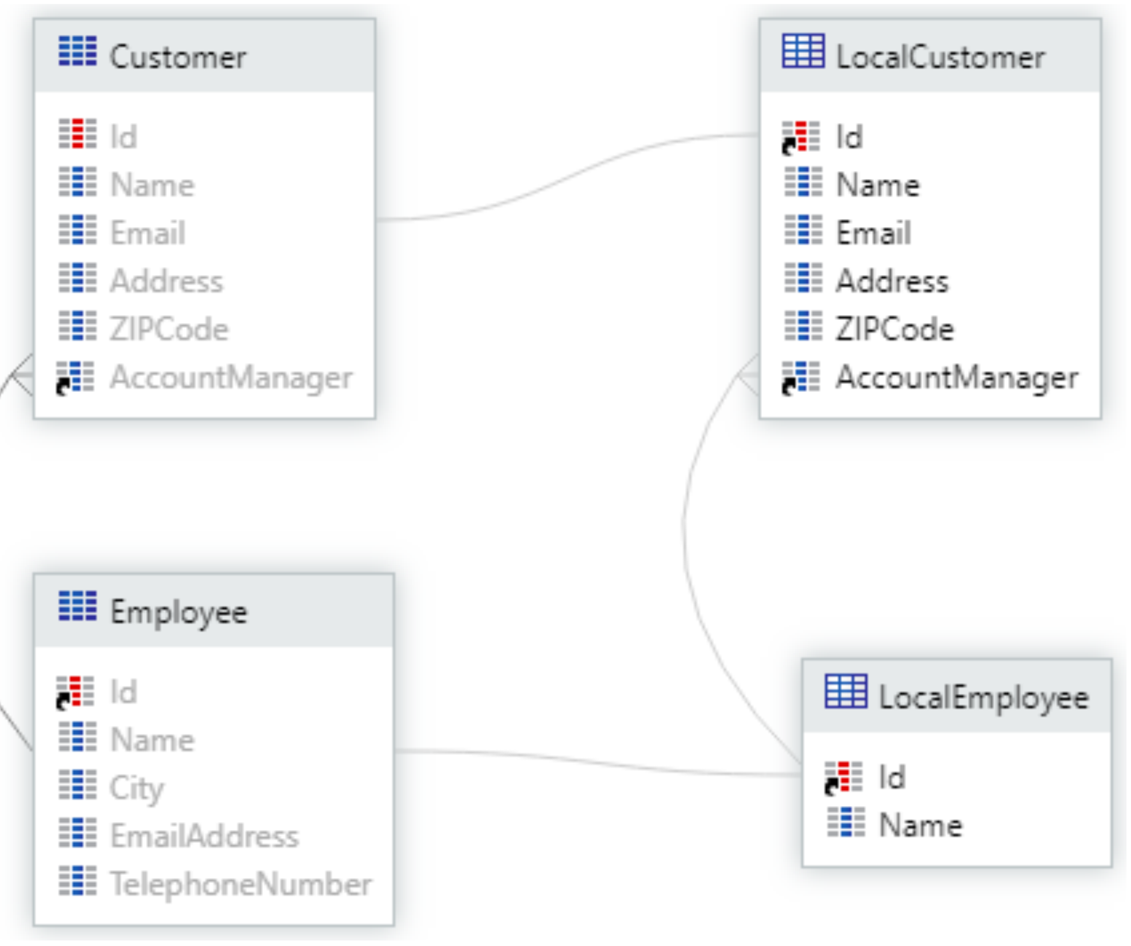
This exercise has a Quickstart application already created. This application has everything needed to start the exercise. This Quickstart application can be found in the Resources folder of this exercise, with the name **Data Sync Exercise.oap**.

Scenario

In this exercise, we will start from an existing application with two Modules. One of the modules (named *DataSynchronizationExerciseCore*) only contains the Entities defined in the Server Database, as well as the logic to bootstrap the data into those Entities. Throughout the exercise, we won't need to modify this Module.

In this exercise, all actions will be performed in the *DataSynchronizationExercise* Module. This Module already contains the user interface of our Mobile App and the Local Storage Entities.

The data model is quite simple, as we can see below.



The two Entities to the left are server Entities, while the two on the right are defined in local storage. We have a set of customers, and each customer has an Account Manager (Employee). This relationship is defined by the *AccountManager* attribute in the *Customer* Entity (and *LocalCustomer*) that references the *Employee* Entity (and *LocalEmployee*).

Regarding the user interface, only two Screens exist: Customers and CustomerDetail. The first displays the list of customers from the Local Storage Entity, while the second allows us to edit the customer information.

Currently, the synchronization is not defined. That is exactly what will be implemented in this exercise. The LocalEmployee Entity will be synced using a Read-Only pattern, while the LocalCustomer will use the Read-Write pattern.

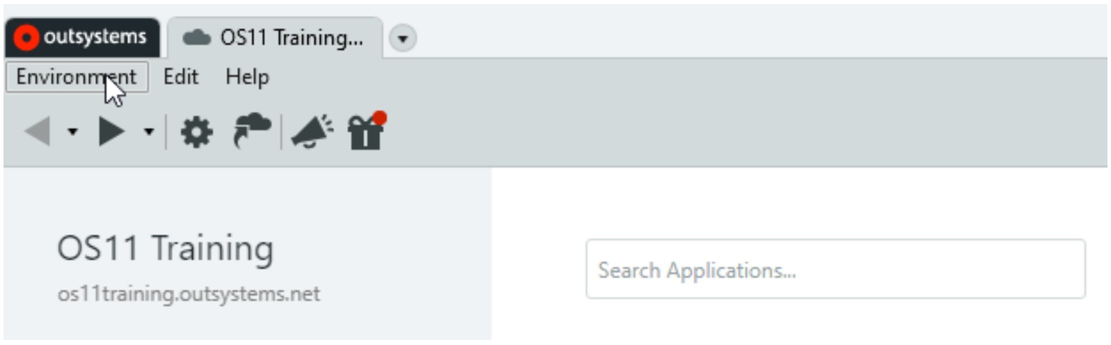
How-To

In this section, we will show you how to do this exercise, with a thorough, step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again**. If you didn't finish the exercise, that's fine! We are here to help you.

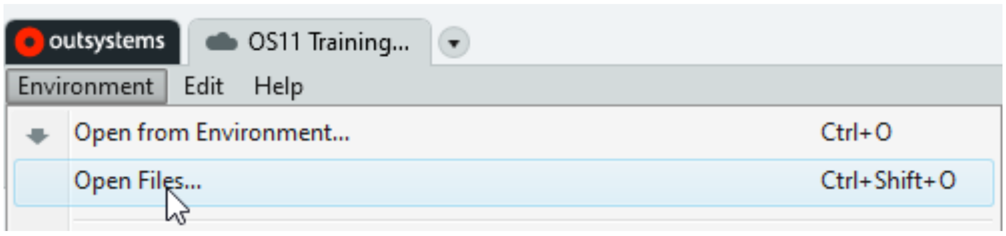
Getting Started

The first step is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

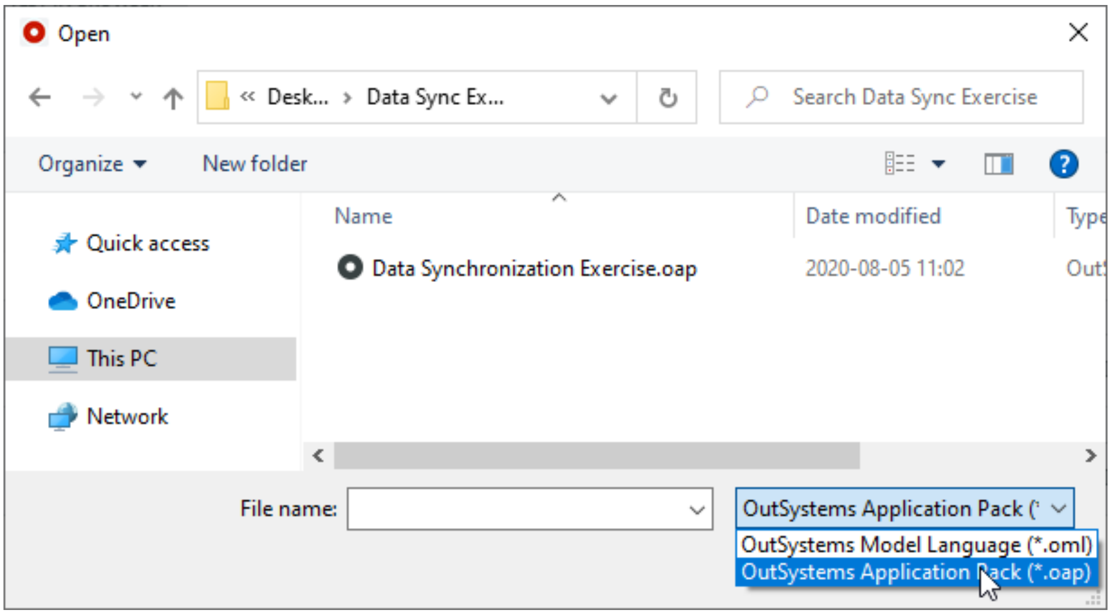
1) In the main window of Service Studio, select the Environment menu on the top left.



2) Select the Open Files...



3) In the new dialog that appears, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart, and open the file named **Data Sync Exercise.oap**.



Open

« Desk... › Data Sync Ex...

Search Data Sync Exercise

Organize ▾ New folder

Quick access
OneDrive
This PC
Network

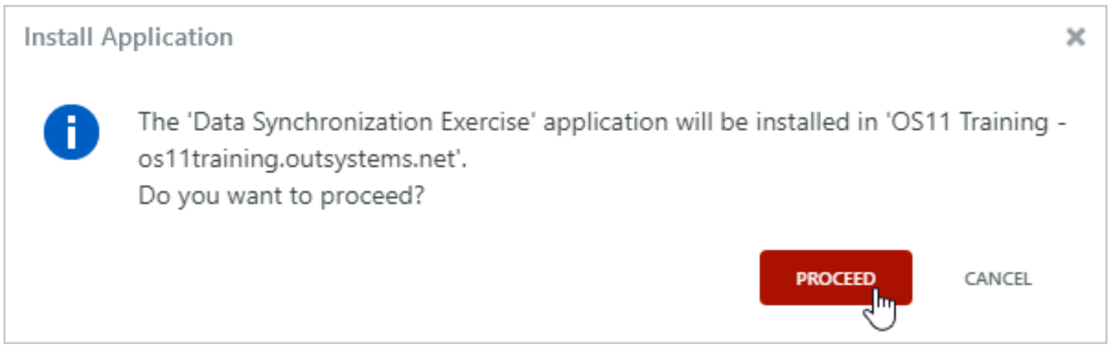
Name	Date modified	Type
Data Synchronization Exercise.oap	2020-08-05 11:02	Out

File name:

OutSystems Application Pack (

OutSystems Model Language (*.oml)
OutSystems Application Pack (*.oap)

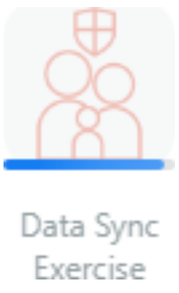
4) In the new confirmation dialog, select Proceed.



Install Application

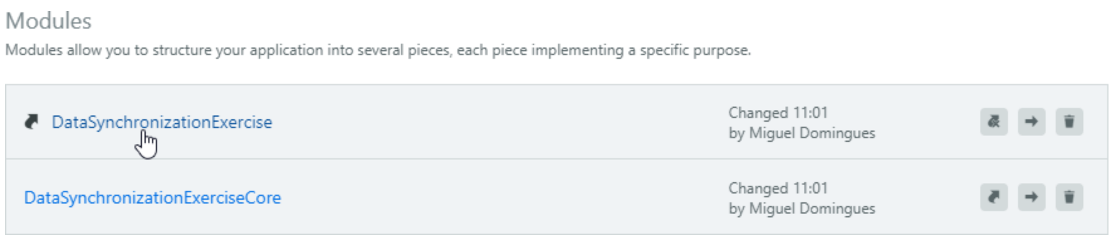
The 'Data Synchronization Exercise' application will be installed in 'OS11 Training - os11training.outsystems.net'.
Do you want to proceed?

5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.

7) The application has two Modules. Open the *DataSynchronizationExercise* Module.



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

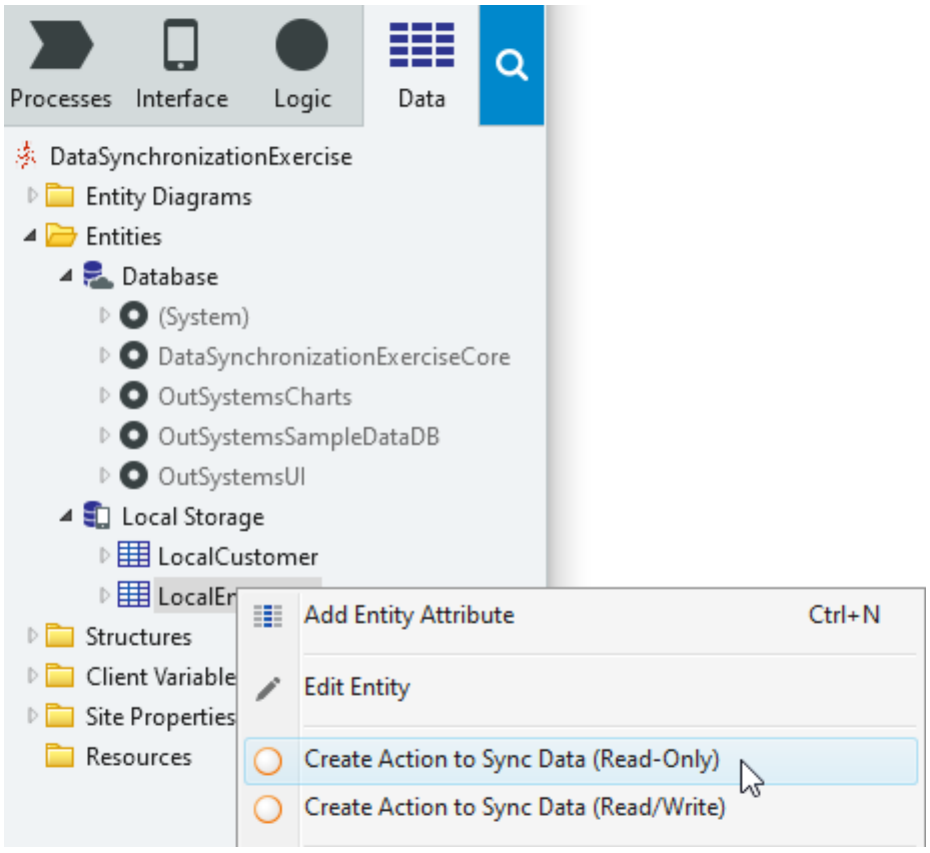
DataSynchronizationExercise	Changed 11:01 by Miguel Domingues
DataSynchronizationExerciseCore	Changed 11:01 by Miguel Domingues

Read-Only Pattern

In this section, we will implement the read-only pattern that will enable syncing employees' data from the server to the Local Storage Entity.

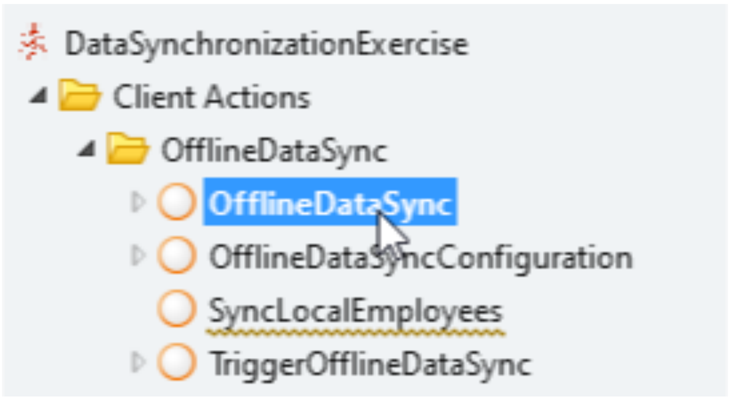
1) Switch to the **Data** Tab.

2) Right-click the LocalEmployee Entity, then select **Create Action to Sync Data (Read-Only)**.

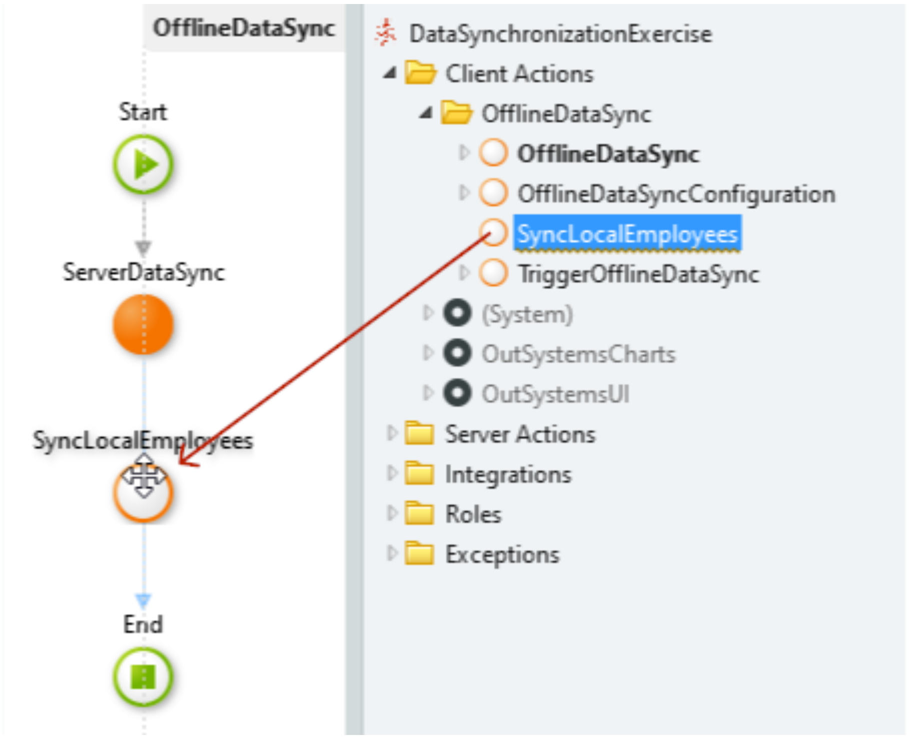


You'll see a new Client Action named *SyncLocalEmployees*, and a Server Action named *SyncEmployees*. These two actions have the read-only pattern implemented for the Employee and LocalEmployee Entities. However, they are not (yet) part of the synchronization process.

3) Open the **OfflineDataSync** Client Action.



4) Drag the **SyncLocalEmployees** Client Action and drop it in the flow after the ServerDataSync.



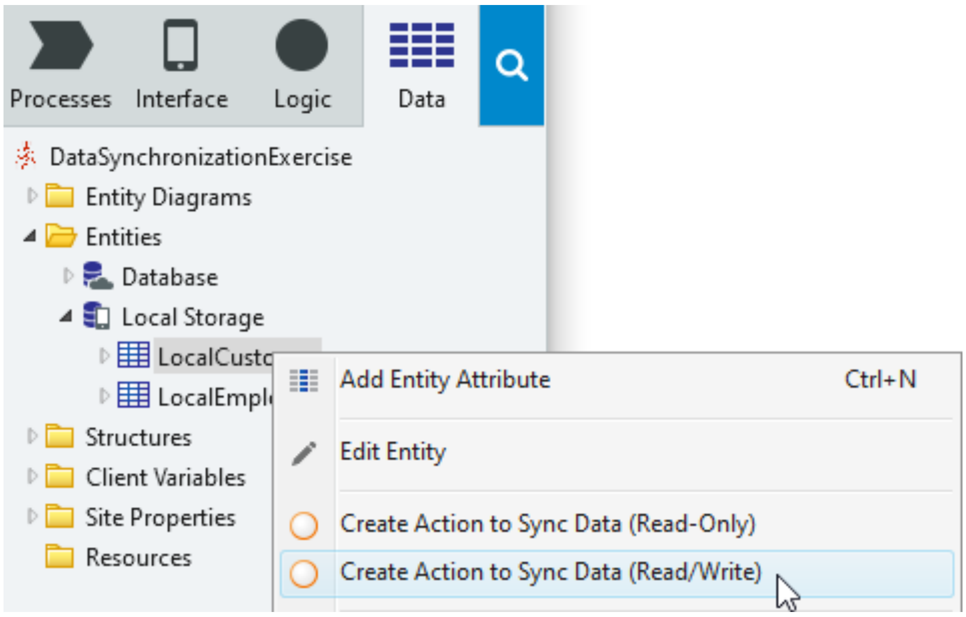
5) **Publish** the Module to save its state.

At this point, it is not yet possible to see this working. Later on, the synchronization of customers will be added, and the app will be tested.

Read-Write Pattern

In this section, the synchronization of customers will be added. This time, the read-write pattern will be used, which will allow us to modify data on the device local storage and then sync it to the server.

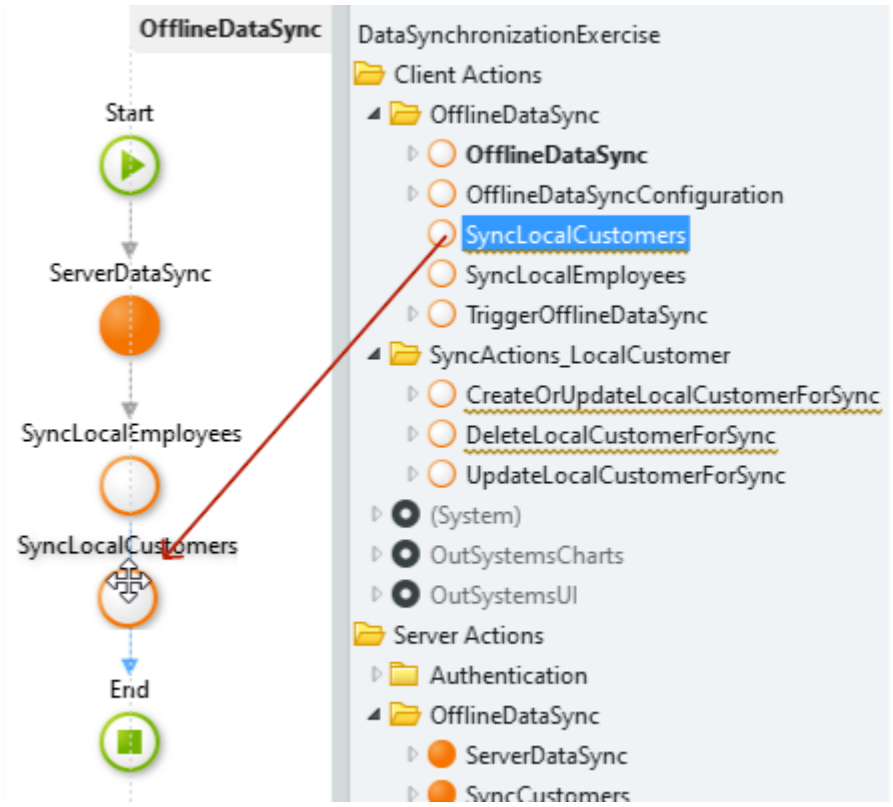
1) On the Data tab, right-click the **LocalCustomer** Entity, then select **Create Action to Sync Data (Read/Write)**.



When compared to the Read-Only pattern accelerator, this accelerator will create a few more Actions. We have the *SyncLocalCustomers* Client Action and *SyncCustomers* Server Action as before, although their logic is a bit different. Three wrappers were also created: *CreateOrUpdateLocalCustomerForSync*, *DeleteLocalCustomerForSync* and *UpdateLocalCustomerForSync*. These client action wrappers should be used when manipulating data on the Local Storage Entity, instead of the built-in Entity Actions. The reason has to do with the fact that the wrappers are aware of the synchronization metadata in the Entity (the new attributes in the *LocalCustomer*: *IsFromServer*, *IsModified* and *IsActive*).

2) Open the **OfflineDataSync** Client Action.

3) Drag the **SyncLocalCustomers** Client Action and drop it between the *SyncLocalEmployees* and the *End*.

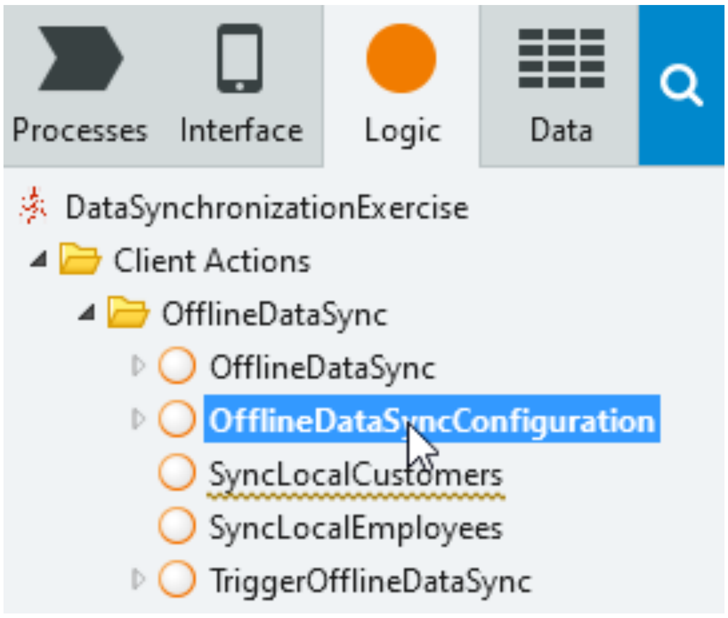


The order of the sync Actions is important. In our case, the Customer Entity has a reference attribute (*AccountManager*) to the Employee Entity; therefore, the Employees Entity should be synced first.

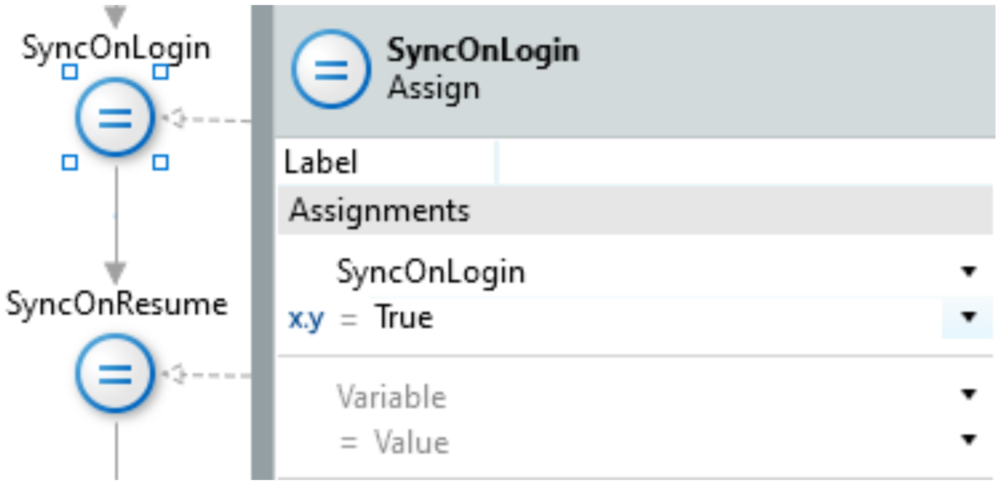
Configure Sync

So far, you have created the synchronization using the read-only and read-write patterns accelerators. In this section, the configuration of the sync will be set.

1) Open the **OfflineDataSyncConfiguration** Client Action.



2) Select the SyncOnLogin Assign (the second in the flow) and change the assignment to 'True'.



SyncOnLogin
x.y = True

3) **Publish** the Module.

Test

1) Click the Open In Browser button.

2) Click on the employee 'Edward Taylor' to log in.

3) On the Customers Screen, you should see the list of existing customers. Clicking any of them will show the details.

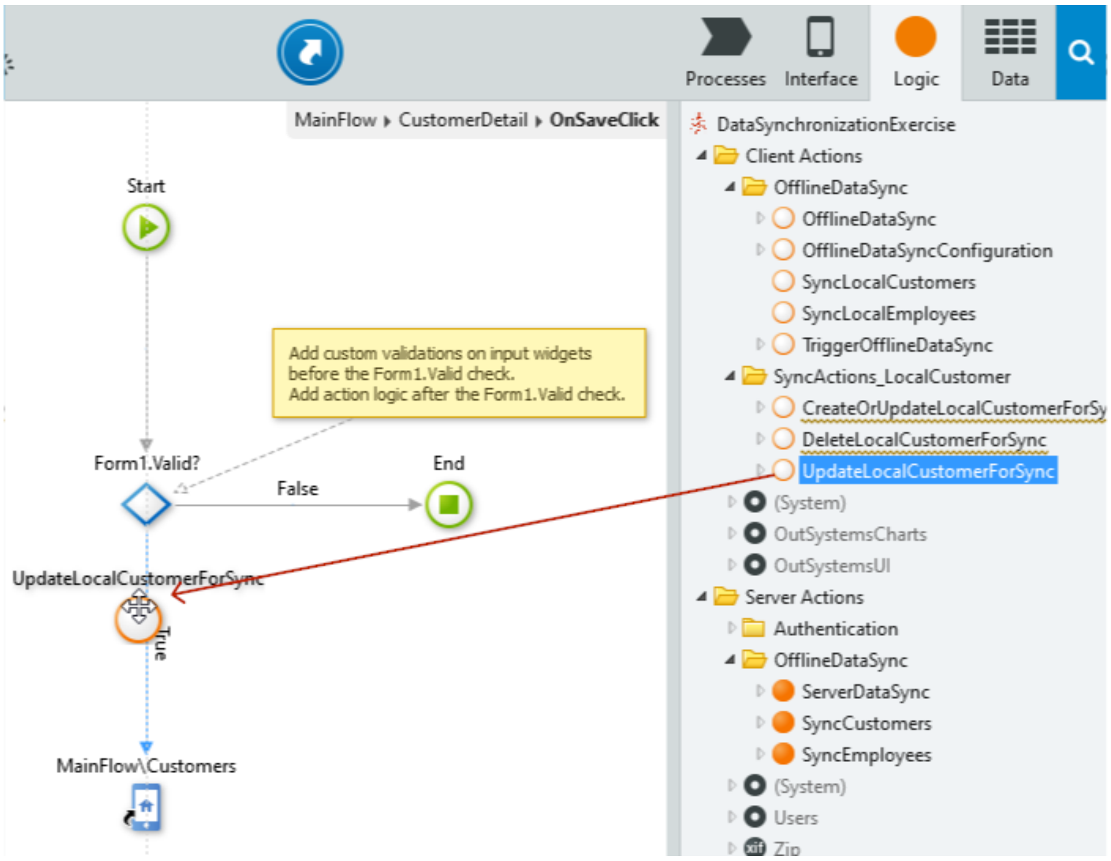
   At this point, the Save functionality is not yet complete. It will be implemented on the next section.

Edit Customers

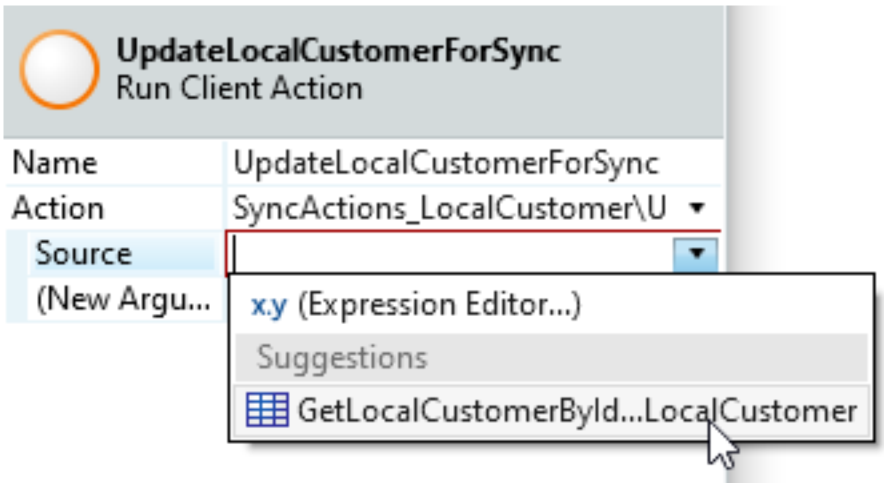
In this section, the logic associated to the Save button of the CustomerDetail Screen will be completed. This will enable us to change the data that is stored on the Local Storage Entities and then sync it to the server.

1) Switch to the Interface tab, then open the **OnSaveClick** Client Action of the CustomerDetail Screen.

2) From the Logic tab, drag the **UpdateLocalCustomerForSync** and drop it on the True branch of the If.

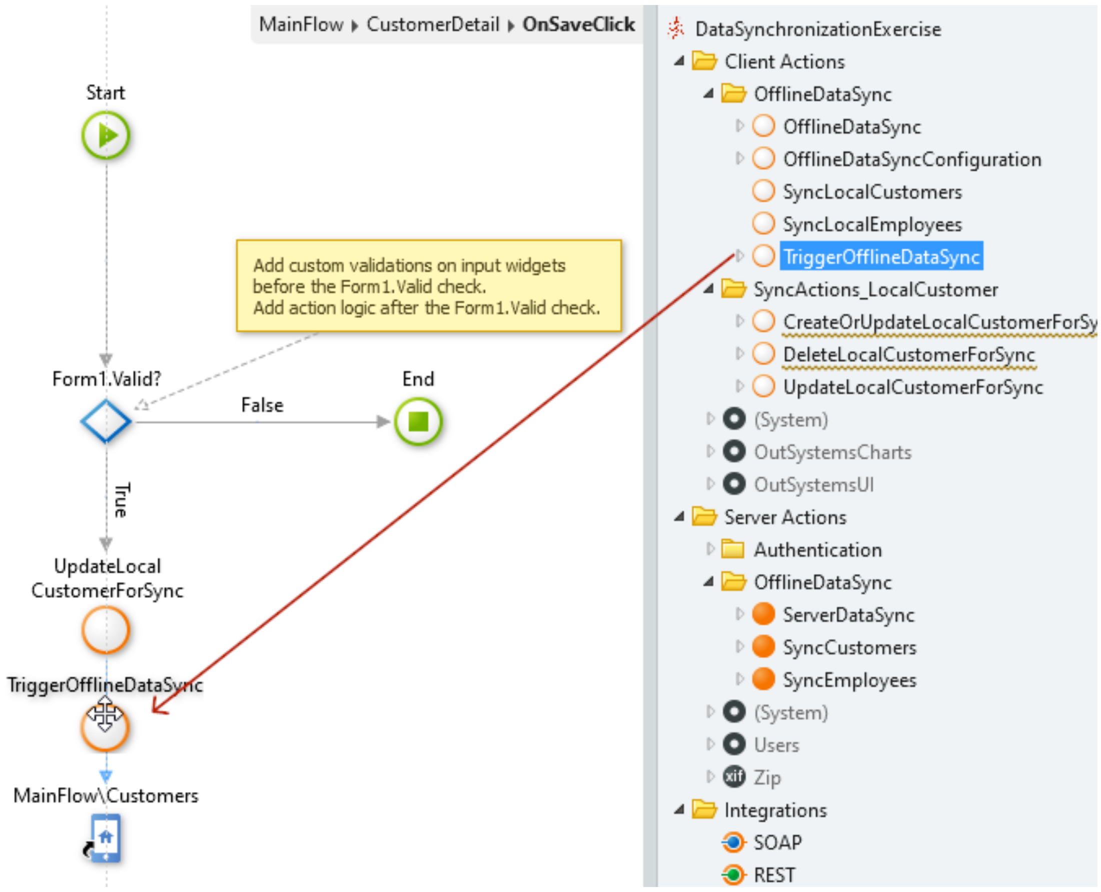

Processes
Interface
Logic
Data
MainFlow ▸ CustomerDetail ▸ OnSaveClick
Start
Add custom validations on input widgets before the Form1.Valid check.
Add action logic after the Form1.Valid check.
Form1.Valid?
False
End
UpdateLocalCustomerForSync
True
MainFlow\Customers
DataSynchronizationExercise
Client Actions
OfflineDataSync
OfflineDataSync
OfflineDataSyncConfiguration
SyncLocalCustomers
SyncLocalEmployees
TriggerOfflineDataSync
SyncActions_LocalCustomer
CreateOrUpdateLocalCustomerForSy
DeleteLocalCustomerForSync
UpdateLocalCustomerForSync
(System)
OutSystemsCharts
OutSystemsUI
Server Actions
Authentication
OfflineDataSync
ServerDataSync
SyncCustomers
SyncEmployees
(System)
Users
Zip

3) In the properties, set the **Source** parameter to `GetLocalCustomerById.List.Current.LocalCustomer`.


UpdateLocalCustomerForSync
Run Client Action
x.y (Expression Editor...)
Suggestions
GetLocalCustomerById...LocalCustomer

Name	UpdateLocalCustomerForSync
Action	SyncActions_LocalCustomer\U
Source	
(New Argu...	

4) Drag the **TriggerOfflineDataSync** Client Action and drop it after the *UpdateLocalCustomerForSync*.


MainFlow ▸ CustomerDetail ▸ OnSaveClick
Start
Add custom validations on input widgets before the Form1.Valid check.
Add action logic after the Form1.Valid check.
Form1.Valid?
False
End
True
UpdateLocal CustomerForSync
TriggerOfflineDataSync
MainFlow\Customers
DataSynchronizationExercise
Client Actions
OfflineDataSync
OfflineDataSync
OfflineDataSyncConfiguration
SyncLocalCustomers
SyncLocalEmployees
TriggerOfflineDataSync
SyncActions_LocalCustomer
CreateOrUpdateLocalCustomerForSy
DeleteLocalCustomerForSync
UpdateLocalCustomerForSync
(System)
OutSystemsCharts
OutSystemsUI
Server Actions
Authentication
OfflineDataSync
ServerDataSync
SyncCustomers
SyncEmployees
(System)
Users
Zip
Integrations
SOAP
REST

It is important to be aware that triggering the sync process often may lead to performance issues, depending on the actual sync process. In this case, the amount of data is relatively small. Optimizations can be done to ensure that only changed data is exchanged between the device and the server. This leads to a more complex data sync procedure, but yields better performance results.

5) **Publish** the Module, then open it in the browser.

6) Navigate to the details of a customer and change the data in the form, then save it.

7) Using the menu, you can log out and log in as a different user and check that the data fetched from the server contains the update.

You may also test it using two different browsers (or the incognito mode) to log in with two different employees and change data on both.