




# Developer Cheat Sheet

## I want to...










### Define custom data structures

1. In **[Data] Tab** , right-click the **[Structures] folder**, and **"Add Structure"**
2. Right-click on the newly created Structure, select **"Add Attributes"**, and set its properties (i.e. Data Type) in the properties panel (bottom right).

### Create a Screen






- **Method 1:** In **[Interface] Tab**  (Elements sub-tab), right-click the **[MainFlow] UI Flow**, and **"Add Screen"**
- **Method 2:** In **[Interface] Tab**  (Elements sub-tab), double-click the **[MainFlow] UI Flow**, and drag a "Screen" widget from the left panel

### Define a reusable Method (OutSystems term: "Action")

1. With **Module scope**:
  - In **[Logic] Tab** , right-click either the **[Client Actions]** or **[Server Actions] folder**, and **"Add Client/Server Action"**
1. OR with **Screen scope**:
  - In **[Interface] Tab**  (Elements sub-tab), right-click the **Screen** , **"Add Client Action"**
2. **An orange ball** (default icon) for the Action will be created: Hollow orange  for client-side (compiles to Javascript) and solid orange  for server-side (compiles to .NET).  
NOTE: Icon can be changed from the properties panel (bottom right).
3. **Right-click** on the newly created Client  or Server  Action to add any **Input/Output Parameters** or **Local Variables**
4. **Double-Click** on the newly created Client  or Server  Action to define the logic, by dragging/dropping Logic widgets (left panel) into the flow and setting their properties (bottom right panel)


5. NOTE: To make the Action also accessible in an [expression](#) as a Function, set the Action's "Function" parameter to "Yes"

## Define Variables





- With **Method scope**:
  - Right-click on the **Action** , "Add Local Variable", and set its Data Type in the properties panel (bottom right)
- With **Screen scope**:
  - In **[Interface] Tab**  (Elements sub-tab), right-click the **Screen** , "Add Local Variable", and set its Data Type in the Properties panel (bottom right)
- With **Session scope** (Cleared on End User log-off or timeout)
  - In **[Data] Tab** , right-click the **[Client Variables] folder**, "Add Client Variable", and set its Data Type in the properties panel (bottom right)
- With **Module scope**:
  - In **[Data] Tab** , right-click the **[Site Properties] folder**, "Add Site Property", and set its Data Type in the properties panel (bottom right)

## Define Variables *beyond* Basic Data Types





In the **Data Type** property of the Variable, to define a Variable of:


- An **Array/List**:
  - Under **Other**, select **List...** then select the List type
- A temporary **Structure**:
  - Under **Other**, select **Record...** then right-click on the Variable and select "Add Attribute" to add other Attributes of the Record/Structure as necessary, defining each Attribute's Data Type in the properties panel
- A **Primary Key** of an Entity:
  - Under **Entity Identifiers**, select the Entity Identifier
- A **Structure** the same as the **Attributes (row) of an Entity**:
  - Under **Entities**, select the Entity
- An **existing Structure**, defined under the **[Data] Tab**  :
  - Under **Structures**, select the Structure

## Define calculations with Operators/Operands/Functions in an Action

- In a conditional: Using **If**   logic widget
  - In the properties panel (bottom right) after selecting the **If** widget, double-click on the **Condition** property row to bring up the [Expression Editor](#) and type your expression in the Expression field using operands and operators, as well as variables, functions, and scripts in scope as seen in the Scope pane (bottom left of Expression Editor)
- In an assignment: Using **Assign**   logic widget
  - In the properties panel (bottom right) after selecting the **Assign** widget, select a **Variable** to assign the expression to, then double-click on the **Value** property row to bring up the [Expression Editor](#) and type your expression in the Expression field using operands and operators, as well as variables, functions, and scripts in scope as seen in the Scope pane (bottom left of Expression Editor)




## Define For Each and While loops in an Action

- **For Each**  logic widget
  1. In the properties panel (bottom right) after selecting the **For Each** widget, fill in the **Record List** property, and optionally the Start Index and Maximum Iterations properties.
  2. Drag your mouse from the **For Each** widget (cursor will turn into crosshairs  ) to drag out the **Cycle** branch to define the 1 or more logic widgets that will execute in this loop.
  3. Complete the loop by dragging from the last logic element back to the **If** widget.
  4. NOTE: You can create logic to exit the loop early by placing an **If** widget within the flow for the exit condition.
- **While:** Using **If**   logic widget
  1. In the properties panel (bottom right) after selecting the **If** widget, fill in the **Condition** property.




2. Drag your mouse from the **If** widget (cursor will turn into crosshairs ) to drag out the **True** branch to define the 1 or more logic widgets that will execute in this loop as long as the If Condition property is true.
3. Complete the loop by dragging from the last logic element back to the **If** widget.
4. NOTE: You can swap the while loop to loop while the **If** condition is False by right-clicking on the If widget, and selecting "Swap connectors"
5. NOTE: You can create logic to exit the loop early by placing an **If** widget within the flow for the exit condition.

## Call an Action

*NOTE: If calling from a **UI element**, first double-click on the UI element to edit the **OnClick Action** for that element.*


- Drag any reusable Action into the Action logic flow
  - From under the current **Screen**  in the **[Interface] Tab**  (Elements sub-tab)
  - From under the **[Client Actions]** or **[Server Actions]** folder in the **[Logic] Tab** 
  - "Run Client/Server Action" from the Logic widgets (left panel)

## Define Database Tables and Relationships

1. In **[Data] Tab** , under the **[Entities] folder**, right-click **[Database]** for **Server-Side** or **[Local Storage]** for **Mobile**, select "Add Entity" to define the Database Table
2. Right-click the newly created **Entity**  , and select "Add Entity Attribute" to define each Entity Attribute (column) in the properties panel (lower right)
3. To define a **Foreign Key** column, set its **Data Type** in the properties panel to one of the "Entity Identifiers" OR name the attribute to **<EntityName>Id** where OutSystems will automatically assume it's the Foreign Key to that Entity.



## Define a Query (SELECT) from a Database visually using Aggregates

1. In an **Action**  , drag the **Aggregate**   widget into the logic flow, then double-click on it to bring up the Aggregate editor


2. Drag any **Entities** you need for this query from the **[Data] Tab**  onto the Aggregate editor. Each one will be added to the **Sources** list under the **Sources** Tab in the Aggregate editor, and any relationships will automatically appear under the **Joins** column, which can also be customized. If a join needs to be manually added, click on **"Add join"**.
3. Add any **Filters** (WHERE) under the **Filters** Tab (May say "No Filters") in the Aggregate editor.
4. Add any **Sorts** (SORT BY) under the **Sorting** Tab (May say "No Sorts") in the Aggregate editor.

## Integrations via REST API

- **Consume** a REST API

1. In **[Logic] Tab** , right-click REST under the **[Integrations] folder**, **"Consume REST API"**, then select **"Add single method"** (You can select "Add multiple methods" if you have access to a REST API with Swagger specification and want to consume multiple methods at once – see documentation)
2. Select the **Method** (GET, POST, PUT, DELETE, PATCH), and paste the **URL**, replacing any **parameters** in **braces** (curly brackets)
3. Depending on the Method selected, fill in the **Body** Tab with sample **Request** and/or **Response** JSON or text/plain example to allow the platform to generate the parameters and structures
  - NOTE: Under the **Test** Tab, you can execute a test of the API Method (providing any required parameters) and copy the request/response body to the Body Tab
4. Under the **Headers and Authentication** Tab, you can define any Request/Response headers, as well as any REST API Authentication requirements
5. Click **Finish**, and you should see a Server **Action**  for the REST API you consumed that you can call from other Actions
6. NOTE: For advanced pre-processing and post-processing, you can define the **OnBeforeRequest** and **OnAfterResponse** Actions that will execute for all REST APIs under the REST API properties.





- **Expose** a REST API

1. In **[Logic] Tab**  , right-click REST under the **[Integrations] folder**, “Expose REST API”, then right-click the newly created REST API and select “Add REST API Method” for each API Method you want to define.
2. Defining each API Method is the same as defining a reusable Method/Action as per above instructions; Double-click on the newly created Action to edit it, add input/output parameters, local variables, call other Actions, etc.
3. NOTE: For every exposed REST API, the OutSystems Platform will automatically generate Swagger documentation that you can share with other developers (including non-OutSystems) who need to integrate with your exposed REST API. You can access this documentation’s URL by right-clicking on the REST API and selecting “Open Documentation”



## Data Conversions

- In an Expression, you have access to the Functions listed under the **Data Conversion** folder under the **Built-in Functions** folder.

## Retrieve data for Screen

- **From Aggregate** (asynchronously executed SQL query on page load)
  - In **[Interface] Tab**  (Elements sub-tab), right-click the **Screen**  , “Fetch Data from Database” and define the Query in the Aggregate editor (see above)
- **From Action** (asynchronously executed Method on page load)
  1. In **[Interface] Tab**  (Elements sub-tab), right-click the **Screen**  , “Fetch Data from Other Sources”
  2. In the newly created Data Action, select the Output parameter and modify the properties (i.e. Data Type) as necessary, and define the Action logic to retrieve the data to assign to the Output parameter.

## Display retrieved data on Screen (Basic)

- As an Expression:
  - In **[Interface] Tab**  (Elements sub-tab), double-click the **Screen**  , and drag any Screen’s **Local Variable**, Data Action **Output Variable**, Screen **Entity**, or any of their **attributes** to the Screen.

- In a Form Element (i.e. Input, Text Area, Switch, Checkbox, etc):
  - Select the Form element widget, and configure the Variable property to bind it to in the properties panel (bottom right)
  - NOTE: It may require Data Conversion (see above)