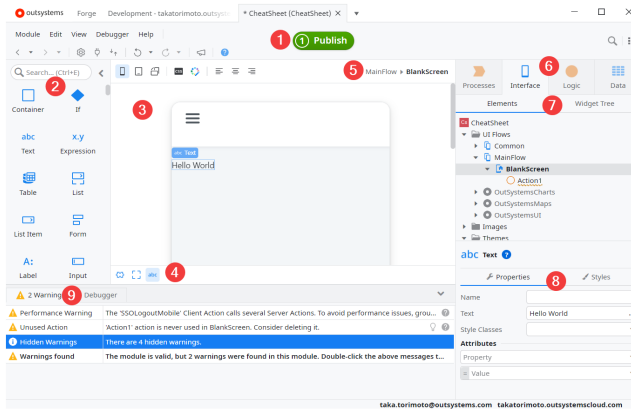


OutSystems Developer Cheat Sheet

Service Studio IDE



- (1) 1-Click Publish Button – Compile and Run
- (2) Toolbox – Elements to drag onto Canvas
- (3) Canvas to design Processes/UI/Logic/Data Model
- (4) Widget Breadcrumb – Navigate through UI hierarchy
- (5) Element Breadcrumb – Navigate through Elements
- (6) Layers Tab – Processes, Interface, Logic, Data
- (7) Elements for selected Layer – Some elements can be dragged onto Canvas; Widget Tree Tab (Interface Layer only) to navigate hierarchy of UI Elements
- (8) Properties for selected Element(s); Styles Tab (Interface Layer only) for selected UI Elements
- (9) True Change and Debugger Tabs

Define custom data structures

1. In **[Data] Layer**, 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 (8).

Create a Screen

- **Method 1:** In **[Interface] Layer**, right-click the **[MainFlow] UI Flow**, and “Add Screen”
- **Method 2:** In **[Interface] Layer**, double-click the **[MainFlow] UI Flow**, and drag a “Screen” widget from the Toolbox (2)

Define a reusable Method (OutSystems term: “Action”)

1. With **Module scope**:

- o In **[Logic] Layer**, right-click either the **[Client Actions]** or **[Server Actions] folder**, and “Add Client/Server Action”
1. OR with **Screen scope**:
 - o In **[Interface] Layer**, right-click the **Screen**, “Add Client Action”
 2. An orange ball for the Action will be created: Hollow orange for client-side (compiles to Javascript) and solid orange for server-side (compiles to .NET). NOTE: This default icon can be changed from the properties panel (8).
 3. Right-click on the newly created Client Action or Server Action to add any **Input/Output Parameters** or **Local Variables**
 4. Double-Click on the newly created Client Action or Server Action to define the logic, by dragging/dropping Logic widgets from the Toolbox (2) into the flow and setting their properties (8)
 5. NOTE: To make the Action also accessible in an **expression** as a Function, set the Action's “Function” property to “Yes”

Define Variables

- With **Method scope**:
 - o Right-click on the **Action**, “Add Local Variable”, and set its Data Type in the properties panel (8)
- With **Screen scope**:
 - o In **[Interface] Layer**, right-click the **Screen**, “Add Local Variable”, and set its Data Type in the Properties panel (8)
- With **Session scope** (Cleared on End User log-off or timeout)
 - o In **[Data] Layer**, right-click the **[Client Variables] folder**, “Add Client Variable”, and set its Data Type in the properties panel (8)
- With **Module scope**:
 - o In **[Data] Layer**, right-click the **[Site Properties] folder**, “Add Site Property”, and set its Data Type in the properties panel (8)

Define Variables beyond Basic Data Types

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

- An **Array/List**:
 - o Under **Other**, select **List...** then select the List type
- A temporary **Structure**:
 - o 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 (8)

- A **Primary Key** of an **Entity**:
 - o Under **Entity Identifiers**, select the Entity Identifier
- A **Structure** the same as the **Attributes (row)** of an **Entity**:
 - o Under **Entities**, select the Entity
- An **existing Structure**, defined under the **[Data] Layer**:
 - o Under **Structures**, select the Structure

Define calculations with Operators/Operands/Functions in an Action

- In a conditional: Using **If** logic widget
 - o In the properties panel (8) 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
 - o In the properties panel (8) 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 (8) 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 (8) 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.



- Complete the loop by dragging from the last logic element back to the **If** widget.
- 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"
- NOTE: You can create logic to exit the loop early by placing an **If** widget within the flow for the exit condition.

Call a Method/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] Layer**  Elements Tab (7)
 - From under the **[Client Actions]** or **[Server Actions]** folder in the **[Logic] Layer** 
 - "Run Client/Server Action" from the Logic widgets (2)


Define Database Tables and Relationships

- In **[Data] Layer** , under the **[Entities] folder**, right-click **[Database]** for **Server-Side** or **[Local Storage]** for **Mobile**, select "**Add Entity**" to define the Database Table
- Right-click the newly created **Entity** , and select "**Add Entity Attribute**" to define each Entity Attribute (column) in the properties panel (8)
- 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

- In an **Action** , drag the **Aggregate**  widget into the logic flow, then double-click on it to bring up the Aggregate editor
- Drag any **Entities** you need for this query from the **[Data] Layer**  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**".
- Add any **Filters** (WHERE) under the **Filters** Tab (May say "No Filters") in the Aggregate editor.
- Add any **Sorts** (SORT BY) under the **Sorting** Tab (May say "No Sorts") in the Aggregate editor.


Consume a REST API


- In **[Logic] Layer** , 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)

- Select the **Method** (GET, POST, PUT, DELETE, PATCH), and paste the **URL**, replacing any **parameters** in **braces** (curly brackets)
- 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
- Under the **Headers and Authentication** Tab, you can define any Request/Response headers, as well as any REST API Authentication requirements
- Click **Finish**, and you should see a Server **Action**  for the REST API you consumed that you can call from other Actions
 - 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

- In **[Logic] Layer** , 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.
- 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.


- 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] Layer**  Elements Tab (7), 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)

- In **[Interface] Layer**  Elements Tab (7), right-click the **Screen** , "**Fetch Data from Other Sources**"
- 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] Layer**  Elements Tab (7), 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 (8)
 - NOTE: It may require Data Conversion (see above)

OutSystems Main Concepts

The OutSystems Platform provides developers with a domain-specific language for developing and changing web and mobile business applications. This developer cheat sheet describes the OutSystems language main concepts and their mappings to other development languages, including .NET.

Processes Layer

Business Processes and Batch Processing

OutSystems	Description	Traditional/.NET/Javascript
Process	Integrate business processes into an application by defining activities within process flows	
Timer	Actions that execute asynchronously on a defined schedule.	Batch job

Process Actions

Launch<Process>	Explicitly launch an instance of a process	
-----------------	--	--

Timer Actions

Wake<Timer>	Explicitly wake a Timer, ignoring its schedule	
-------------	--	--

Interface Layer

User Interface

OutSystems	Description	Traditional/.NET/Javascript
UI Flow	User interaction diagram of screens	N/A

UI Flow Tools

Screen	HTML pages	
Block	Reusable group of widgets for other blocks/screens	
Email	Send an email asynchronously via a sending queue.	
External Site	Connects to a site external to your module through a well-known URL you specify	

Screen and Block Tools

Widget	UI Controls	UI Control Classes
Input Parameter	Send data to a screen	Query string variable of an HTTP request, or an HttpContext Item in Server request
Local Variable	Variables with Screen scope	Instance variable of the web page
Fetch Data from Database	Asynchronous Action that retrieves data to the Client using a Server-side Aggregate	
Fetch Data from Other Sources	Asynchronous Action that retrieves data to the Client using a Server Action flow	
Client Action	Actions only available in Screen; compiles to Javascript on screen	

Screen and Block Widgets

Container	A "box" to contain other widgets and containers	Panel
If	Control displayed content based on condition(s) with "then/else"	if/else
Expression	Render value computed at run-time	Label
(See Documentation for Others)	Table, List, List Item, Form, Label, Input, Text Area, Switch, Checkbox, Radio Group, Dropdown, Upload, Button, Button Group, Link, Popover Menu, Image, Icon, Popup, Block, HTML Element, etc	

Logic Layer

Business Logic

OutSystems	Description	Traditional/.NET/Javascript
Client/Server Action	Business Logic; Server Actions compile to .NET and run on the Server; Client Actions compile to Javascript and run on the Client.	Method

Input Parameter	Send data into Action	Method parameter
Output Parameter	Return data from Action	Method out parameter
Local Variable	Variables with Action scope	Method Local Variable
User-Defined Function	Actions that can be invoked in Expressions	Static Method

Client Action Tools (Compiles to Javascript on the Client)

Start	Start of Action flow	
Message	Pop-up an Alert message	
Run Client Action	Call a Client-Side Method	
Run Server Action	Call a Server-Side Method	
Aggregate (Mobile)	SELECT Query, using visual editor; for JOIN, WHERE, SORT BY, GROUP BY, etc; DB agnostic (DB type is abstracted away)	
Refresh Data	Refreshes a Screen Aggregate to update the resultset for the Client *Only available on Screen Actions	
Comment	Add a comment	
If	Conditional for branching Action flow into 2 paths	If statement
Switch	Conditionals for branching Action flow into multiple paths	Switch/case/break statement
For Each	Repeats Action path execution	Foreach statement
Assign	Assigns 1 or more expressions to 1 or more variables	Series of = statements
JSON Serialize	Convert a Record or a List of Records into JSON data-interchange format text.	

JSON Deserializ e	Convert a JSON string into a Structure or List	
Exception Handler	Handle exceptions by catching them and allowing you to define a parallel error handling flow	Catch statement with catch block
Raise Exception	Explicitly launch a user exception	Throw statement
Javascript	Add in in-line Javascript code to execute	
End	Ends the Action flow. More than one End element in the same Action flow is allowed for organization.	
Destination	Jumps to a specific screen *Only available on Screen Actions	
Download	Download facility *Only available on Screen Actions *Not on Mobile	Method that writes the request response to a file, by setting the request content type, its header, and writing the file content to its output stream, all processed on the Client side.
Server Action Tools (Compiles to .NET on the Server)		
Start	Start of Action flow	
Run Server Action	Call a Server-Side Method	
Aggregate	SELECT Query, using visual editor; for JOIN, WHERE, SORT BY, GROUP BY, etc; DB agnostic (DB type is abstracted away)	Method with SQL query returning a dataset
SQL	SELECT Query, using DB-specific text-based SQL	Method with SQL query returning a dataset
If	Conditional for branching Action flow into 2 paths	If statement
Switch	Conditionals for branching Action flow into multiple paths	Switch/case/break statement
For Each	Repeats Action path execution	Foreach statement

Assign	Assigns 1 or more expressions to 1 or more variables	Series of = statements
Record List To Excel	Exports contents of a record list to an Excel file	Method that writes a list of objects into an Excel file using a library to write Excel files
Excel To Record List	Imports content of an Excel file into a record list	Method that reads a list of objects from an Excel file using a library to read Excel files
JSON Serialize	Convert a Record or a List of Records into JSON data-interchange format text.	
JSON Deserializ e	Convert a JSON string into a Structure or List	
Exception Handler	Handle exceptions by catching them and allowing you to define a parallel error handling flow	Catch statement with catch block
Raise Exception	Explicitly launch a user exception	Throw statement
Comment	Add a comment	
Send Email	* Does not exist in Data Action	
End	Ends the Action flow. More than one End element in the same Action flow is allowed for organization.	

Integrations

OutSystems	Description	Traditional/.NET/Javascript
SOAP	Expose/Consume SOAP API	
REST	Expose/Consume REST API	
SAP	Integrate with SAP BAPI	
Extension (.xif)	Wrappers to existing C# libraries, database tables, etc, visually exposed inside the IDE as reusable Actions, Entities, and Structures. Extensions are created/modified with Integration Studio.	Class Library project compiled into a native assembly DLL

Security

OutSystems	Description	Traditional/.NET/Javascript
Role	Implement security representing a group of elements that share the same grant policy	Security setting granted to users and roles
Role Actions		
Check<Role>	Checks whether a specific End User has been granted access to a specific permission area	N/A
Grant<Role>	Provides access for a specific End User to a specific permission area.	N/A
Revoke<Role>	Denies access for a specific End User to a specific permission area.	N/A

Data Layer

Database (Server-Side) and Local Storage (Client-Side)

OutSystems	Description	Traditional/.NET/Javascript
Entity	Persistent data storage representing database model	Database Table, similar to DataSource or an Object-Relational (OR) Mapping Class
Entity Attribute	Entity storage	Database Table Field, similar to DataSource field or a field in an Object-Relational (OR) Mapping Class
Entity Actions	CRUD (Create, Read, Update, Delete) Actions automatically created and managed by the OutSystems platform for each Entity	OR Mapping CRUD Methods or Methods that execute the CREATE, GET, UPDATE, and DELETE SQL statements
Static Entity (Server Only)	Entity with static data managed during design time with strong typing. Used for constants and enumerations.	Entity with static and fixed data (enums) defined during design-time.