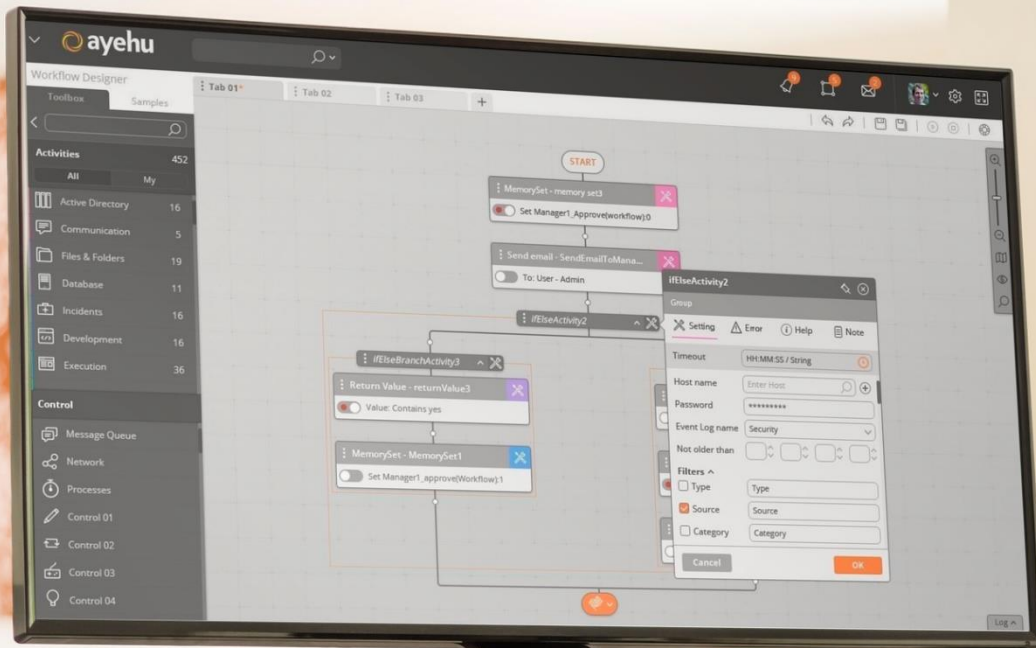




## Ayehu Activity Designer

12 August 2019



## Table of Contents

<b>1. WHAT IS AN ACTIVITY? .....</b>	<b>3</b>
<b>2. ACTIVITY DESINER OVERVIEW .....</b>	<b>3</b>
<b>3. FRONTEND CODE – JSON OVERVIEW .....</b>	<b>4</b>
<b>3.1. MANDATORY CONTROL PROPERTIES .....</b>	<b>4</b>
<b>3.2. OPTIONAL CONTROL PROPERTIES.....</b>	<b>7</b>
<b>3.3. SPECIAL PROPERTIES FOR CONTROL TYPES .....</b>	<b>8</b>
<b>3.4. OPTIONAL PROPERTIES FOR BASE TYPES .....</b>	<b>12</b>
<b>3.5. CONTROL TYPES EXAMPLES .....</b>	<b>13</b>
<b>3.6. FRONTEND CODE FULL EXAMPLES .....</b>	<b>14</b>
<b>4. BACKEND CODE - OVERVIEW .....</b>	<b>17</b>
<b>4.1. VB MANDATORY OBJECTS.....</b>	<b>17</b>
<b>4.2. C# MANDATORY OBJECTS.....</b>	<b>18</b>
<b>4.3. CLASS PROPERTIES .....</b>	<b>19</b>
<b>4.4. BACKEND CODE FULL EXAMPLES.....</b>	<b>20</b>
<b>5. ACTIVTY DESIGNER FULL EXAMPLES.....</b>	<b>22</b>

## 1. WHAT IS AN ACTIVITY?

An activity is an operative or logical action that makes up the workflow process. Activities may be simple or complex, and they can vary greatly in nature and purpose. Examples of activities are copying a file, sending an email, creating a new user, or retrieving current CPU information.

## 2. ACTIVITY DESIGNER OVERVIEW

Introduced first in Ayehu NG v1.4, the Activity Designer is a built-in coding tool that enables you to easily design and create a variety of custom activities.

The custom activities created are available for use in the Toolbox of the Workflow Designer, enriching the pre-built activities list.

To help you get started and create activities quickly, the Activity Designer consists of two code components (Frontend and Backend):

1. Frontend code – JSON is used for creating the UI box of each custom activity.
2. Backend code – Visual Basic or C# can be used for writing the custom activity's backend code.

**Important Disclaimer!!!** Please note that the usage of custom activities is at your own risk. The Activity Designer is provided on an "AS IS" basis, and any workflows including custom activities will not be officially supported by Ayehu Support.

Furthermore, it is highly recommended that you validate your code before implementing an activity in Ayehu NG. Ignoring errors and exceptions might lead to a memory leak, which subsequently can cause the platform to run slowly or even crash. To avoid any potential issues, ensure that you check for errors while using efficient data structures and algorithms.

### 3. FRONTEND CODE – JSON OVERVIEW

```

1  {
2    "data": {
3      "name": "Add Spaces",
4      "description": "Add a space between chars",
5      "Timeout": "00:01:00",
6      "class": [],
7      "rootSettings": {
8        "isCollapse": false,
9        "activitySettings": [

```

The JSON tab is where you can write the custom activity's frontend code, which is how the UI box of each custom activity is created.

The JSON of the Activity Designer consists of the following:

1. General properties (begins with the "data" property):
  - a) "name" – activity's name in Ayehu NG ("name": "Add Spaces").
  - b) "description" – activity's description in Ayehu NG ("description": "Add a space between chars").
  - c) "Timeout" – activity's timeout in Ayehu NG ("Timeout": "00:01:00").
  - d) "activitySettings" – an array of control objects that represent the activity's settings ("activitySettings": []).
2. Control properties (begins with the "activitySettings" property):
 

While most of the controls have the same properties, some of them have unique properties that are relevant only to them. The following sections describe the mandatory, optional, and special properties of the different controls.

### 3.1. MANDATORY CONTROL PROPERTIES

```
10      {
11          "value": "",
12          "key": "TheValue",
13          "label": "String",
14          "labelKey": "ADD_SPACES_STRING",
15          "baseType": "control",
16          "controlType": "textbox"
17      }
```

The JSON code requires several mandatory properties to make it work. The following table summarizes the different mandatory properties.

Property Name	Possible Values	Description	Examples
value	string	The initial value of the control. Oftentimes, it is set to an empty string ("").	<pre>"value": "" "value": "User" "value": "389"</pre>
key	string	The key name of the control. Must be equal to the property name in the activity class and "execute" method.	<pre>"key": "DateFirst" "key": "AccountType" "key": "ADUserName"</pre>
label	string	The label of the control. Can be set to an empty string as well.	<pre>"label": "First Date" "label": ""</pre>
labelKey	string	Used for translation of the label. "labelKey" is built as following: "name" + "label", while each space is replaced with underscore ("_") and all letters are upper case.  *If label above is empty, "labelKey" will be empty as well.	<pre>"name": "Date Difference" "label": "First Date"  "labelKey": "DATE_DIFFERENCE_FIRST_DATE"</pre>
baseType	<pre>"control" "group" "hostGroup"</pre>	<p>The base type of the control:</p> <p><b>"control"</b> – used for any of the controls listed below.</p> <p><b>"group"</b> – used whenever you want to group one control or more under one section.</p> <p><b>"hostGroup"</b> - many activities have the following controls together: "HostName", "UserName", "Password".</p> <p>Using <b>"hostGroup"</b> will add them all together, including their necessary properties.</p>	<pre>"baseType": "control" "baseType": "group" "baseType": "hostGroup"</pre> <p>Please see <b>section 3.6</b> for full examples of "hostGroup" and "group".</p>

controlType	“textbox” “dropdown” “autocomplete” “textarea” “richtext” “password” “datePicker” “autocompleteWithPlus” “checkbox” “radiobutton” “hidden”	<p>The type of the control. These are the most common control types.</p> <p>“hidden” – should be used if you want the control value to be sent to the server, but you don’t want it to be visible to the user.</p> <p>Usually a static value.</p>	“controlType” : “textbox” “controlType” : “dropdown” “controlType” : “autocomplete” “controlType” : “hidden” <p>Please see <b>section 3.5</b> for more information.</p>
-------------	--	---	--

### 3.2. OPTIONAL CONTROL PROPERTIES

The JSON code offers a few optional properties in addition to the mandatory ones. The following table summarizes the different optional properties.

Property Name	Possible Values	Description	Examples
required	true false	<p>Determines whether a field in the original activity is required.</p> <p>A required field will have a red * character next to the label, requiring the user to fill this field before saving. The user won’t be able to save the activity if the field is empty.</p> <p>Default value: <b>false</b></p> <p><b>*note:</b> if control is disabled, it is no longer required.</p>	“required”: true “required”: false
styleClass	string	Adds an existed css class to the control.	“styleClass”: “red”

disabled	true false	Disables/enables the control (visually). Default value: <b>false</b>	"disabled": true
----------	---------------	--	------------------

### 3.3. SPECIAL PROPERTIES FOR CONTROL TYPES

In addition to the properties above, some of the control types ("controlType" property) have their own properties:

**For controls "dropdown", "autocomplete", and "autocompleteWithPlus":**

Property Name	Possible Values	Description	Example
controlOptions	array of key-value pair	An array of objects that represent the source of the dropdown.  *Cannot be used together with autoCompleteParams	[ { "key": "1", "value": "value1" }, { "key": "2", "value": "value2" } ]
idFieldName	string	By default, a dropdown control sends the server only the text that appears in it (the "value" property). If you want to send the "key" (which is usually an id) property you should use "idFieldName".	"idFieldName" : "HostId"
autocompleteParams	Object	Sets dropdown source from an external API. The "entityType" number represents the API number.  *Cannot be used together with "controlOptions".  *See " <b>entityType</b> " property section below for more details.	"autocompleteParams": { "entityType" : 11 }
valueChangesActions	Object	Used to make changes after changing an input. *See " <b>valueChangesActions</b> " property section below for more details.	Please see <b>section 3.6</b> for full example.



**For control “textarea”:**

Property Name	Possible Values	Description	Example
styleClass	“xl-textarea” “xm-textarea” “xs-textarea”	xl - large size xm - medium size xs - small size	“styleClass” : “xl-textarea”

**For control “password”:**

Property Name	Possible Values	Description	Example
encrypt	true false	Encrypts password. Default value: <b>false</b>	“encrypt”: true

**For control “checkbox”:**

Property Name	Possible Values	Description	Example
convertBoolTo	“number”	Oftentimes the server accepts Boolean values as numbers:  0 – false 1 – true  Default value is <b>null</b> , which means – send the Boolean value as it is, without converting.	“convertBoolTo” : “number”
value	true false	The value in the checkbox must be Boolean (unlike the string used in all other control types).	“value”: false
valueChangesActions	object	Used to make changes after changing an input. *See “ <b>valueChangesActions</b> ” <b>property</b> section below for more details.	Please see <b>section 3.6</b> for full example.

**For control “radiobutton”:**

Property Name	Possible Values	Description	Example
controlOptions	array of key-value pair	An array of objects that represents the options of the radiobutton group.	[ { "label": "Value 1", "value": "1" }, { "label": "Value 2", "value": "2" } ]
styleClass	“one-line”	Aligns radio buttons horizontally rather than vertically. Default order align is vertically.	“styleClass”: “one-line”
valueChangesActions	object	Used to make changes after changing an input. *See “ <b>valueChangesActions</b> ” <b>property</b> section below for more details.	Please see <b>section 3.6</b> for full example.

**"entityType" property (for property "autocompleteParams"):**

```

None = 0,
Workflow = 1,
Device = 2,
User = 3,
Group = 4,
Duty = 5,
Templates = 6,
Global_Variables = 7,
Services = 8,
Classifications = 9,
Commands = 10,
Sites = 11,
OperationSystem = 12,
OperatorTypes = 13,
LogsCategories = 14,
Subject = 15,
TimeFrames = 16,
Conditions = 17,
Modules = 18,
ErrorHandling = 19,
Clusters = 20,
Activities = 21,
ActivityLogStatus = 22,
ConditionType = 23,
ModuleTypes = 24,
Destination = 26,
Incident = 27,
MessageBody = 28,
Source = 29,
Platform = 30,
IncidentDevices = 31,
IncidentServices = 32,
IncidentAnyClassificationDevices = 33,
IncidentAnyClassificationServices = 34,
MessageTemplates = 35

```

**"valueChangesActions" property (for controls "dropdown", "checkbox", and "radiobutton"):**

As you can see in the Date Difference example (#3 under section 3.6), the **"checkbox"** control has a **"valueChangesActions"** property in it.

If its value is set to **"false"** then the controls **"FirstDateFormat"** and **"FirstDate"** will be enabled, and if its value is set to **"true"** then two actions will be fired:

1. **"disable"** - will disable the controls **"FirstDateFormat"** and **"FirstDate"** (disable and enable actions receive an array of strings – control keys).
2. **"setValue"** - will set their values to be empty strings ("").

**"valueChangesActions"** supports 3 types of actions:

1. Enable
2. Disable
3. Set Value

Usually, this property is used along with the control types: **"checkbox"**, **"radiobutton"**, and **"dropdown"**.

### 3.4. OPTIONAL PROPERTIES FOR BASE TYPES

Some of the base types ("baseType" property) have optional properties:









#### For "hostGroup":

Property Name	Possible Values	Description	Example
customKeys	object	<p>When using the "baseType": "hostGroup", it produces static key names for the controls: HostName, UserName and Password.</p> <p>The "customKeys" property should be used in order to give these controls different key names.</p>	<pre>"customKeys": {   "hostName":"DstHostName",   "userName":"DstUserName",   "password":"DstPassword"   "hostId":"DstHostId" }</pre> <p>Please see <b>section 3.6</b> for full example.</p>

#### For "group":

Property Name	Possible Values	Description	Example
isVisible	true false	Sets the visibility of the group when opening an activity.	"isVisible" : true

### 3.5. CONTROL TYPES EXAMPLES

Control Type in JSON	Control Type in Ayehu NG
"controlType": "textbox"	<b>User Name</b> <input type="text"/>
"controlType": "autocomplete"	<b>Port</b> <input type="text" value="389"/> ▼
"controlType": "dropdown"	<b>Module Name*</b> <input type="text" value="AWS"/> ▼
"controlType": "autocompleteWithPlus"	<b>Host Name*</b> <input type="text" value="localhost"/> ▼  
"controlType": "password"	<b>Password*</b> <input type="password" value="....."/>
"controlType": "checkbox"	<input type="checkbox"/> Automatically Change the Severity
"controlType": "radiobutton"	<b>Message</b> <input type="radio"/> Original <input checked="" type="radio"/> Custom <input type="radio"/> Template
"controlType": "textarea"	<b>Query*</b> <div><div></div></div>
"controlType": "richtext"	<b>Body</b> <div> <div> <b>B</b> <i>I</i> <u>U</u> <u>A</u> <del>A</del> </div> <div>     </div> </div> <div></div>
"baseType": "hostGroup"	<b>Host Name*</b> <input type="text" value="localhost"/> ▼   <b>User Name</b> <input type="text"/> <b>Password</b> <input type="password"/>

### 3.6. FRONTEND CODE FULL EXAMPLES

We have summarized a couple of full frontend code examples used to create custom activities via the Activity Designer:

#### **1. Active Directory Add to Group (example of "hostGroup"):**

ADAddtoGroup- adAddtoGr...

Adds a user or a computer account to a group

Settings Error Help Notes

Account Type\* ☒ User ☐ Computer

Name\*

Host Name\*  ☒

User Name

Password

Group Name\*

Port

Cancel Save

#### **Link to JSON code in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Active%20Directory/ADAddtoGroup.json>

## **2. File Copy (example of “group”, “hostGroup”, and “customKeys”):**

FileCopy- fileCopy1

Copies a specified file from a source location to a...

Settings Error Help Notes

Source

Path\*

Host Name\*

User Name

Password

Destination

Path\*

Host Name\*

Cancel Save


Source Group

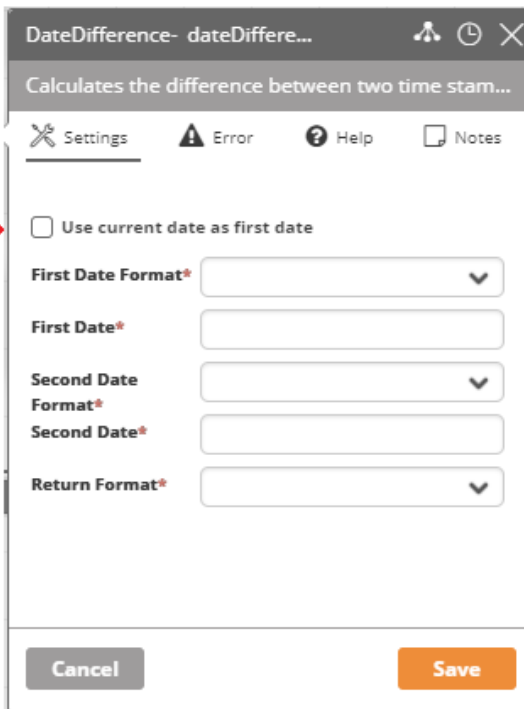
Destination Group

### **Link to JSON code in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Files%20and%20Folders/FileCopy.json>

### **3. Date Difference (example of “valueChangesActions”):**

valueChangesActions 



#### **Link to JSON code in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Time%20Controls/DateDifference.json>



## 4. BACKEND CODE - OVERVIEW

The Code tab is where you can write the custom activity's backend code, which can be written in Visual Basic or C#.

To make it work, the backend code requires a package named 'Ayehu.Sdk.ActivityCreation', which can be downloaded from the [NuGet website](#).

As mentioned previously in the Activity Designer Overview section, it is highly recommended that you validate your code before implementing an activity in Ayehu NG. Ignoring errors and exceptions might lead to a memory leak, which subsequently can cause the platform to run slowly or even crash. To avoid any potential issues, ensure that you check for errors while using efficient data structures and algorithms.

### 4.1. VB MANDATORY OBJECTS

The Activity Designer's VB code requires a couple of mandatory objects to make it work. The following code sample summarizes the different mandatory objects:

```

1. Imports Ayehu.Sdk.ActivityCreation.Interfaces
2. Imports Ayehu.Sdk.ActivityCreation.Extension
3. Imports System.Text

4. Namespace Ayehu.Sdk.ActivityCreation

5.     Public Class ActivityClass
6.         Implements IActivity
7.         Public TheValue As String
8.         Public Function Execute() As ICustomActivityResult Implements IActivity.Execute
9.             Dim dt As DataTable = New DataTable("resultSet")
10.            dt.Columns.Add("Result", GetType(String))

11.            dt.Rows.Add(Math.Round(Convert.ToDecimal(TheValue)))
12.            Return Me.GenerateActivityResult(dt)
13.        End Function
14.    End Class
15. End Namespace

```

1. Must declare using Ayehu.Sdk.ActivityCreation.Interfaces library (see section 4).
2. Must declare using Ayehu.Sdk.ActivityCreation.Extension library (see section 4).
3. Declare any libraries you need to use in your code.
4. Any namespace can be used.
5. Class declaration – public class with any name implementing IActivity interface.
6. Class properties – must be public with the name corresponding to the “key” value of the control that was configured in the JSON (frontend code). The property will be populated on run-time

with the value from the control. If needed, you can use your own properties with different names\protection levels (see section 4.3 for more details about the different “key” values).

7. Public Method that implements the IActivity interface (void method returning ICustomActivityResult).
8. Return ICustomActivityResult using SDK extension "GenerateActivityResult" – accepting DataTable (dt) or String (result).

## 4.2. C# MANDATORY OBJECTS

The Activity Designer’s C# code requires a couple of mandatory objects to make it work. The following code sample summarizes the different mandatory objects:

```

1. using Ayehu.Sdk.ActivityCreation.Interfaces;
2. using Ayehu.Sdk.ActivityCreation.Extension;
3. using System.Text;

4. namespace Ayehu.Sdk.ActivityCreation
   {
5.     public class DisplayValue: IActivity
6.     {
7.         public string ValueToDisplay;
8.         public ICustomActivityResult Execute()
           {
               var result = "Hello " + ValueToDisplay;
               return this.GenerateActivityResult (result);
           }
       }
   }

```

1. Must declare using Ayehu.Sdk.ActivityCreation.Interfaces library (see section 4).
2. Must declare using Ayehu.Sdk.ActivityCreation.Extension library (see section 4).
3. Declare any libraries you need to use in your code.
4. Any namespace can be used.
5. Class declaration – public class with any name implementing IActivity interface.
6. Class properties – must be public with the name corresponding to the value of the control "key" that was configured in the JSON. The property will be populated on run-time with the value from the control. If needed, you can use your own properties with different names\protection level (see section 4.3 for more details about the different “key” values).
7. Public Method that implements the IActivity interface (void method returning ICustomActivityResult).
8. Return ICustomActivityResult using SDK extension "GenerateActivityResult" – accepting DataTable (dt) or String (result).

### 4.3. CLASS PROPERTIES

The class property name must correspond to the value of the control "key" that was configured in the frontend code – JSON:

**1.** If the JSON uses "**baseType**": "**control**", then the class properties can be any of the "key" values that were configured in the JSON (see section 3.1).

For example:

"ADUserName"

"ADGroupName"

"SecurePort"

**2.** If the JSON uses "**baseType**": "**group**", then the class properties can be any of the "key" values that were configured in the JSON (see section 3.1).

For example:

"Source"

"Destination"

**3.** If the JSON uses "**baseType**": "**hostGroup**", then the class properties must use the following values (see section 3.1):

"HostName", "UserName", "Password".

#### 4.4. BACKEND CODE FULL EXAMPLES

We have summarized a couple of full backend code examples used to create custom activities via the Activity Designer:

##### **1. Active Directory Add to Group (example of "hostGroup"):**

ADAddtoGroup- adAddtoGr...

Adds a user or a computer account to a group

Settings Error Help Notes

Account Type\* ☒ User ☐ Computer

Name\*

Host Name\*  ☒

User Name

Password

Group Name\*

Port

Cancel Save

##### **Link to Visual Basic code in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Active%20Directory/ADAddtoGroup.vb>

## 2. DB2 Statement:

The screenshot shows a configuration window titled "DB2Statement- db2Statement1". Below the title bar, a subtitle reads "Executes a statement on a DB2 database". The window contains a toolbar with icons for Settings, Error, Help, and Notes. The main area has two required text input fields: "Statement\*" and "Connection String\*". Below these is a checkbox for "User Authentication". If checked, it reveals two more input fields: "User Name" and "Password". At the bottom, there are "Cancel" and "Save" buttons.

DB2Statement- db2Statement1

Executes a statement on a DB2 database

Settings Error Help Notes

Statement\*

Connection String\*

☐ User Authentication

User Name

Password

Cancel Save

### Link to C# code in Ayehu GitHub:

[https://github.com/Ayehu/activities/blob/master/Database/DB2Statement.c  
s](https://github.com/Ayehu/activities/blob/master/Database/DB2Statement.cs)

## 5. ACTIVITY DESIGNER FULL EXAMPLES

We have summarized a couple of full examples used to create custom activities via the Activity Designer:

### **1. Max Function:**

The screenshot shows a window titled 'Max- max1' with standard OS window controls. Below the title bar is a header bar with the text 'Returns the higher of two numbers'. Underneath is a toolbar with four icons: a wrench for 'Settings', a warning triangle for 'Error', a question mark for 'Help', and a notepad for 'Notes'. The main area contains two input fields. The first is labeled 'First Value\*' and the second is labeled 'Second Value\*'. At the bottom of the window are two buttons: a grey 'Cancel' button on the left and an orange 'Save' button on the right.

**Link to the JSON code (frontend code) in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Math/Max.json>

**Link to the VB code (backend code) in Ayehu GitHub:**

<https://github.com/Ayehu/activities/blob/master/Math/Max.vb>

#### **About Ayehu**

Ayehu's AI-powered automation and orchestration platform is a force multiplier for IT and security operations, helping enterprises save time on manual and repetitive tasks, accelerate mean time to resolution, and maintain greater control over IT infrastructure. Trusted by hundreds of major enterprises and leading technology solution and service partners, Ayehu supports thousands of automated processes across the globe. For more information, contact [info@ayehu.com](mailto:info@ayehu.com)