**3-4– App Errors & Conditions** 

Macro and apps usually need user input to process the workflow so it is not uncommon that errors might be created when a wrong field is mapped or a datatype has been changed. Aside from errors, some processes can get complex and might require you to add conditions when taking in input values. Adding Error Message Tools and Condition Tools to your apps and macros allows for a flexible workflow. For this lesson, we are going to discuss how these tools work as well as their use cases.

**Error Message Tool is used to display an error message for an app or macro based on criteria in an expression. If an error message displays, downstream processing stops.**

Let’s create a simple app that takes in both username and password and checks if it matches the stored constant strings. In a new workflow, drag a text box tool from the interface palette and place it into the canvas. Configure the text tool to display the text “Username:”. Next, drag another text tool into the canvas. Configure it to display the text “Password:”. Once the labels are done, tick the box for “Mask Text'' to hide the password characters as an asterisk. Now that we have the interface text in place, let's check how it looks like by using the wand icon beside the run button to run the workflow as an analytical app. There should be 2 textboxes for the Questions tab, 1 for the Username, and another for the Password. Users can freely type in any strings on both textboxes, but the password textbox is masked since we enabled the mask function. Since we simply added the front-end interface tools, clicking the finish button will not generate any message nor show any output. We will use the error message tool to generate messages if the username and password entered does not match our text and if the textboxes were left empty. But before we add additional tools, let’s look at how constants work.

**Constants are global variables for a workflow that are defined on the Workflow tab in the Workflow Configuration window. Constants make it possible to change a value in a single location and have that change apply to the rest of the workflow.**

On the alteryx designer, click on any blank part of the canvas and navigate to the workflow tab of the configuration window. The constants are listed on the lower part of the window. There are 4 columns on this list. “Type” for the type of constant, “Name”, the constants “Value” and lastly, the “Is numeric” checkbox. Constants can be used dynamically but they can only hold a single value which you can set under the Value column**. By default, constants are set as strings, but you can make them numeric by ticking the “Is numeric” checkbox. You can use constants in any expression, such as the expression boxes in formulas. Simply navigate to the variables tab to show the list of constants. In addition, you can also use constants using reference shortcuts. Add a percent symbol at the start and end of the complete constant name. E.g., “%User.Constant1%”.** The reference shortcut can then be used in an expression editor or in a text box, and the constant can be updated via a Question tool or modified on the Workflow tab in the Workflow Configuration window. You need to be careful when using the reference shortcuts as it might not work exactly as you think it will.

**There are 3 types of constants; Engine, Question and User. Engine constants are available in every workflow. They contain the workflow information, and additional engine constants are added depending on the type of file. For example, during the iterative macro’s lesson, we’ve used the “Iteration Number” which is an engine constant that contains the number of times the macro ran. Question constants are automatically added if you insert an interface tool in an app or macro. The name of the constant will correspond to the name of the tool in the Annotation tab of the Tool Properties. In this app we have 2 text boxes as question constants since we added the tools earlier. User Constants can be added by using the + button on the constant’s configuration window. You can add any name and value as long as they are not conflicting with any of the Engine and Question constants.**

Let’s go back to our alteryx app and add 2 user constants. Click the + button on the constant’s configuration window, then type the Name “UN” with value “Admin”. Once done, click the + button again and type the name “PW” with value “1234@alteryx”. We will leave the IS numeric checkbox unchecked since these constants will both be strings. These constants will now stick to the workflow even if you package it, send it, schedule it or upload to a Gallery.

From the interface tool, add an error message tool and connect its Q anchor to the text box 1 which has the username input. There are 2 settings you need to configure for this tool; the expression and the error message. If the expression is true, the display message will be shown to the end-user upon clicking run or finish. For this first error message, we wanted to check if the user has entered the correct username. In order to do this, we need to compare the text box 1 input, and the UN constant. Click the expression editor and insert a variable. Double click on the “Text Box username” under connections from Questions to add the username input on the expression. This will reflect “[#1]” on the expression box. Next, type “!=” then insert a constant. Open the constants tree to show the complete list. Double click on “User.UN”. Now that we have the expression, type in the error message that will pop up if the expression was true. Under the error message box, type “Username is incorrect” This completes our first error message. Duplicate this tool and connect its copy to the text box for password. In the 2nd error message, change the expression to “[#1]!=[User.PW]”. Once you’ve updated the expression, change the error message after. In the message box, type “Password is incorrect”.

Next, we will add error messages if one or more textbox is null or empty. Add another error message tool and connect it to the Q anchor of the text box 1 for username. On its expression box, type “IsNull([#1]) or IsEmpty([#1])” make sure no errors were found on the expression, then type the error message “Please input your username”. After configuring the 3rd error message, create a copy of it and connect it to the textbox 2 for password. Once connected, edit the error message on the duplicate to “Please input your password”. This error message tool should capture if any of the textboxes were null or empty.

To test them, click the wand button and input “user” on the username and “123” on the password. Hit finish to run the app. The analytic app will take all error messages and show them on the pop up. Since 2 errors were found, 2 messages have been shown. Close the error message. Next, leave both text boxes blank and hit finish. As we can observe, all 4 errors were shown since the scenario hit all 4 conditions.

Now, we are going to continue working on the app and create a simple process which will be enabled only when the correct username and password was entered. In order to text both inputs value and pass data to another tool, we are going to employ the Condition tool.

The **Condition Tool** is used to test for entered values and return either true or false. The Condition tool is similar to a Filter tool except for the fact that it tests end user inputs instead of table data. If values meet the criteria of the expression, they come out of the True output. If values do not meet the criteria, they come out of the False output.

We are going to use the condition tool on this app to check if the username and password entered matched our constants. If so, it will replace the dummy data with a string and output it to a log file. First, navigate to the interface tool tab and drag a condition tool to the canvas. Connect the Q anchor of both text boxes to the condition tool. Once done, add the expression “[#1]=[User.UN] and [#2]=[User.PW]” If the UN and PW values match perfectly with the text box inputs, it will flow to the T anchor, if not, it will go to the F anchor. We already created the error messages so there is no need to create a separate process for the F anchor. Add a new text input tool below the condition tool. We are going to create a dummy or placeholder for our data. Create a new column on the text input and name it as “Message”. Set the value to “placeholder for message”. Once done, connect an output data tool to it and save the output as “log.yxdb” in your pc. To pass a new message value, add an action tool between the condition tool and the text input. Connect the diamond input anchor of the action tool to the T output anchor of the condition tool, then connect the lighting bold output of the action tool to the text input. To update the value, select the action type “Update Value with Formula”. Navigate the text input tree until you find Data > r > c-value = “placeholder for message”. Select this line as the value to update. On the bottom part of the configuration for the action tool, set the formula as “"Welcome " + [User.UN] + "! " + DateTimeNow()”.

Finally, click the wand button to run it as an analytic app. Input the username “Admin” and password “123@alteryx”. Then hit finish. Since we passed the condition, the new value should be written on the log file. Open it to check. The logged message should contain “Welcome Admin!” as well as the date and time of the app run.