Configuring Business Logic - Scripts and Error Messages Online Training

Abstract:

This training will take you into more practical uses for DELMIA Apriso. You will use what you have learned up to this point and expand upon it, creating a screen based on pre-determined requirements. You will also expand your use of common operation functions like the SQL query, User Formula, and Show Message function.

Detailed objectives. After the training you will know:

- Create a screen to meet specifications
- Expand knowledge of SQL query, User Formula, and Show Message functions

Target audience:

- People who need to configure business logic as part of their customer projects

Requirements:

- Working knowledge of Process Builder
- Skills to build Screens in Process Builder
- Skills to configure Standard Operations in Process Builder

Role and level:

- Working knowledge of Process Builder

Duration: 60 min

If you went through the Screen Flow Management training modules, you have used these functions already. Here, you will have an opportunity to expand and rehearse your knowledge and skills on them.

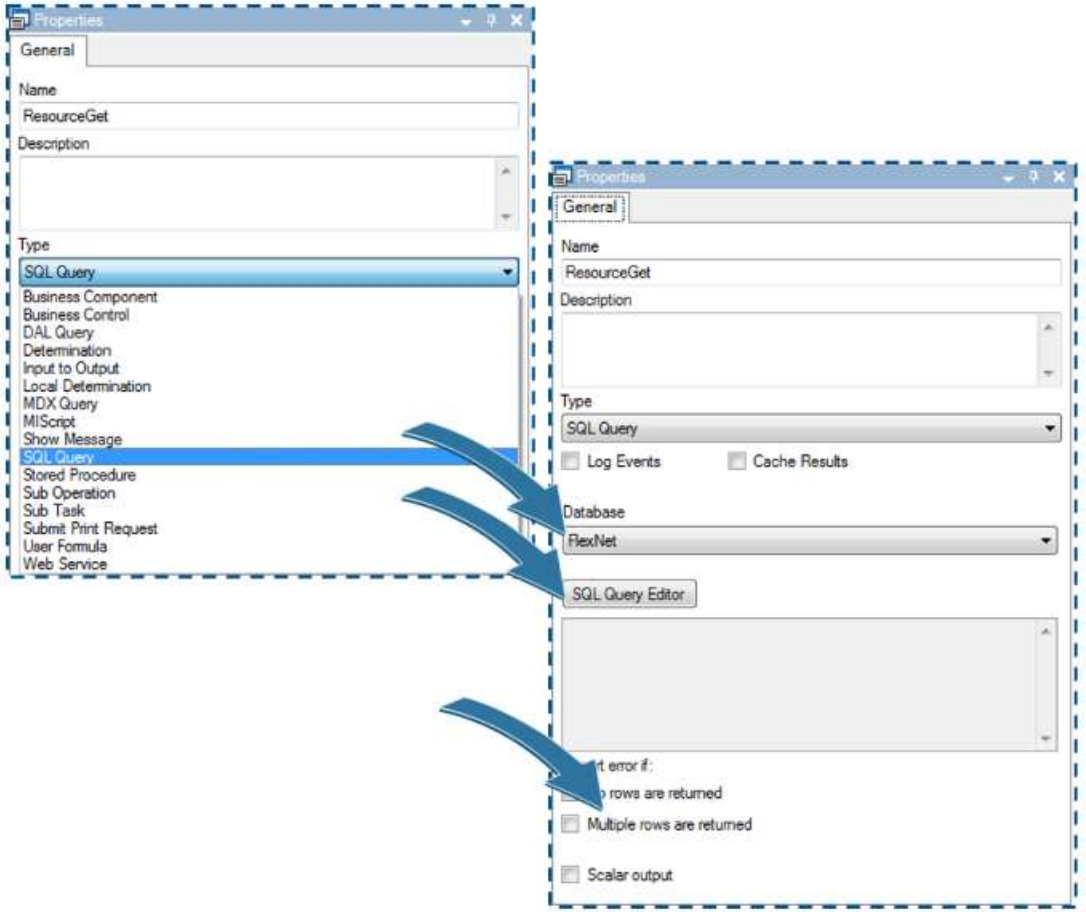
SQL Query Function

The SQL Query Function executes a user-defined query against the database and return results.

This function type is used only to extract data from the database (i.e. Select statements only). In order to add/update/delete data, Business Components and stored procedures are used.

The properties allow you to:

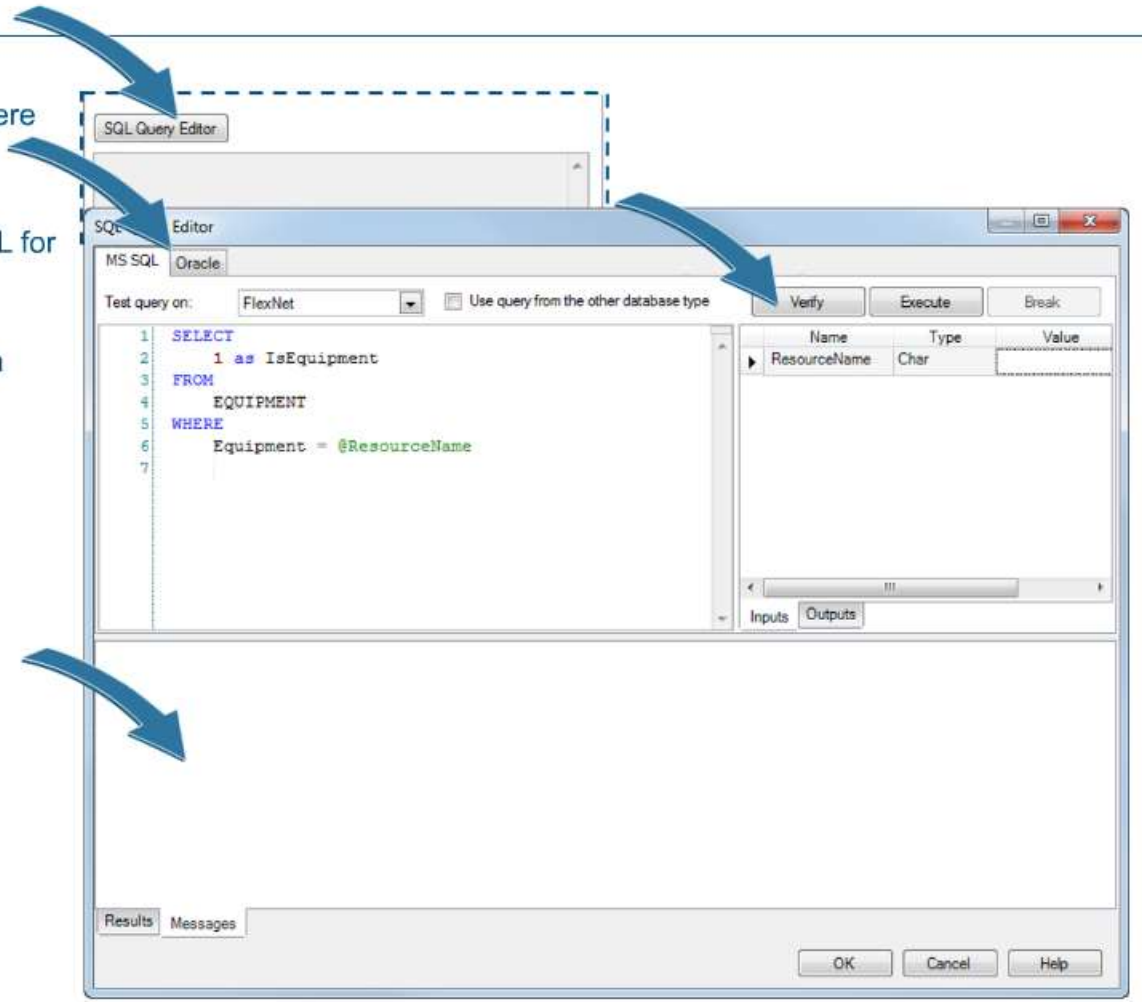
- Run the query against defined database
- Edit, validate, and test execute the query
- Change between array and scalar outputs and allow for error messages



SQL Query Editor

The SQL Query Editor button opens a popup, where you can work with your query.

- You have a choice between MS SQL and SQL for Oracle
- You can verify the syntax of your script - any messages appear at the bottom of the screen

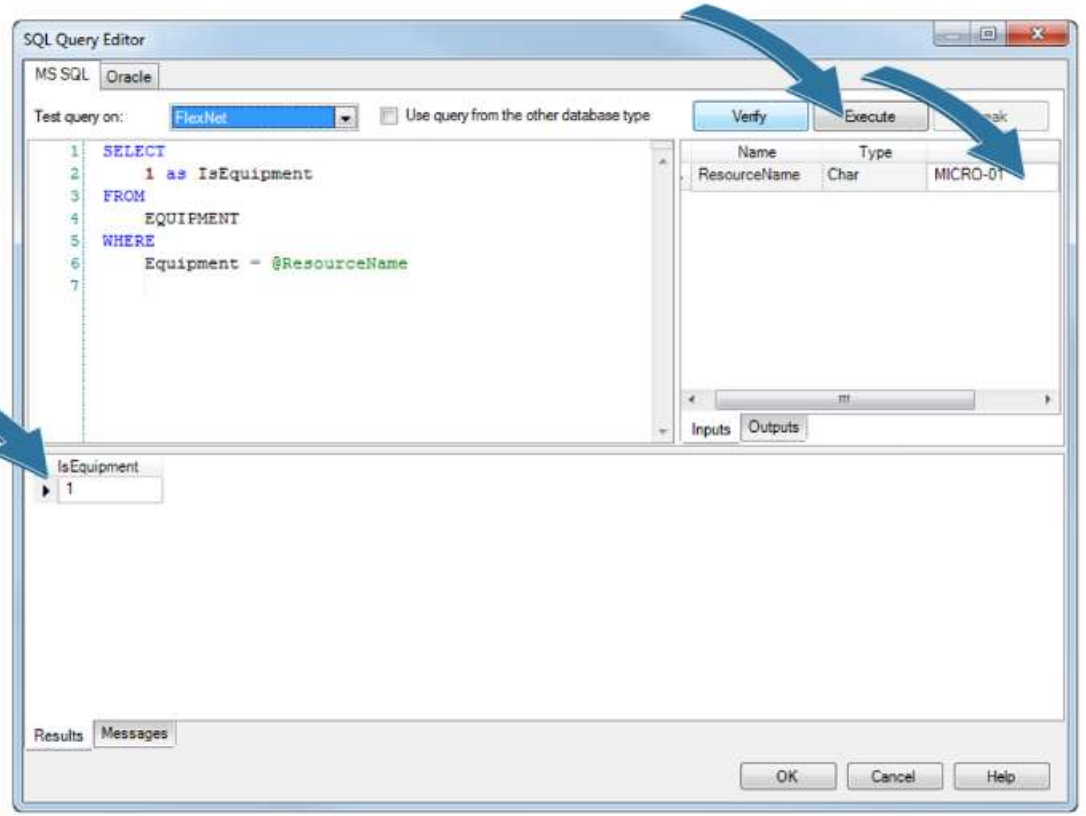


SQL Query Editor - Test Run

You can also test run your query.

For this purpose, you can enter values for required inputs, if needed:

- Click Execute
- Observe how the bottom tab changes to Results and shows what your query will return

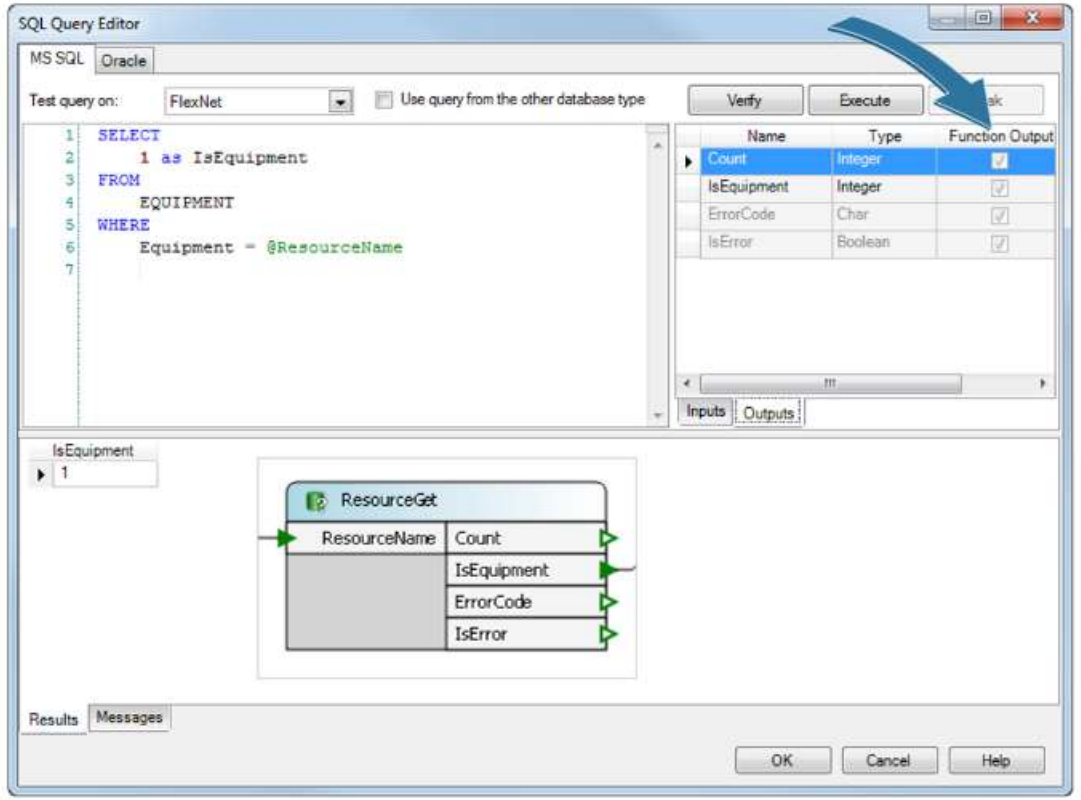


SQL Query Editor - Inputs and Outputs

When you add a SQL Query function, you don't need to create function inputs and outputs first.

When you paste or enter your SQL query, and close the SQL Query Editor, the system will automatically configure the query's inputs and outputs as the function inputs and outputs.

Also, when you reopen your SQL Query function, and modify the query, the system will allow you to decide which of new query outputs should be shown as function outputs. In such cases, you will see the check boxes in the Function Output column as active.



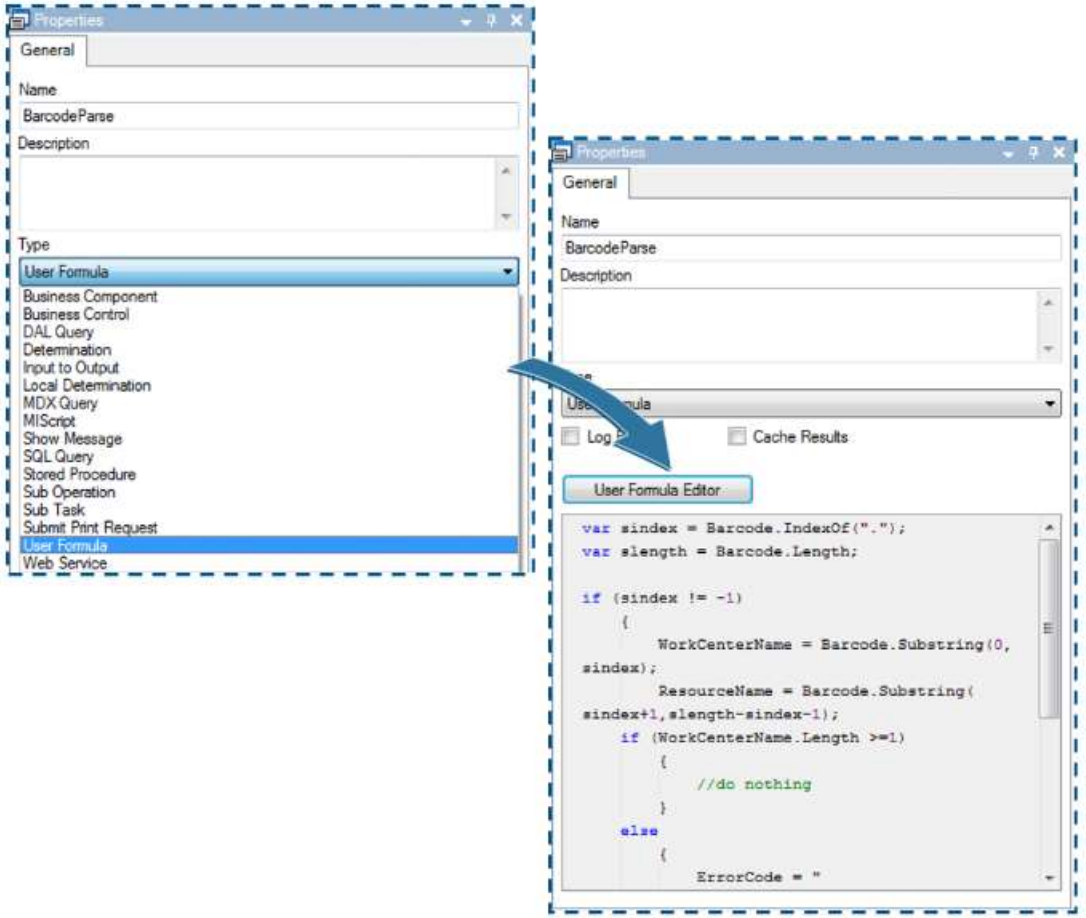
User Formula Function

A User Formula function is used to add programming language scripts to the business logic.

DELMIA Apriso supports scripts in

- C#
- JavaScript
- Visual Basic

The formula is added and edited in the User Formula Editor.



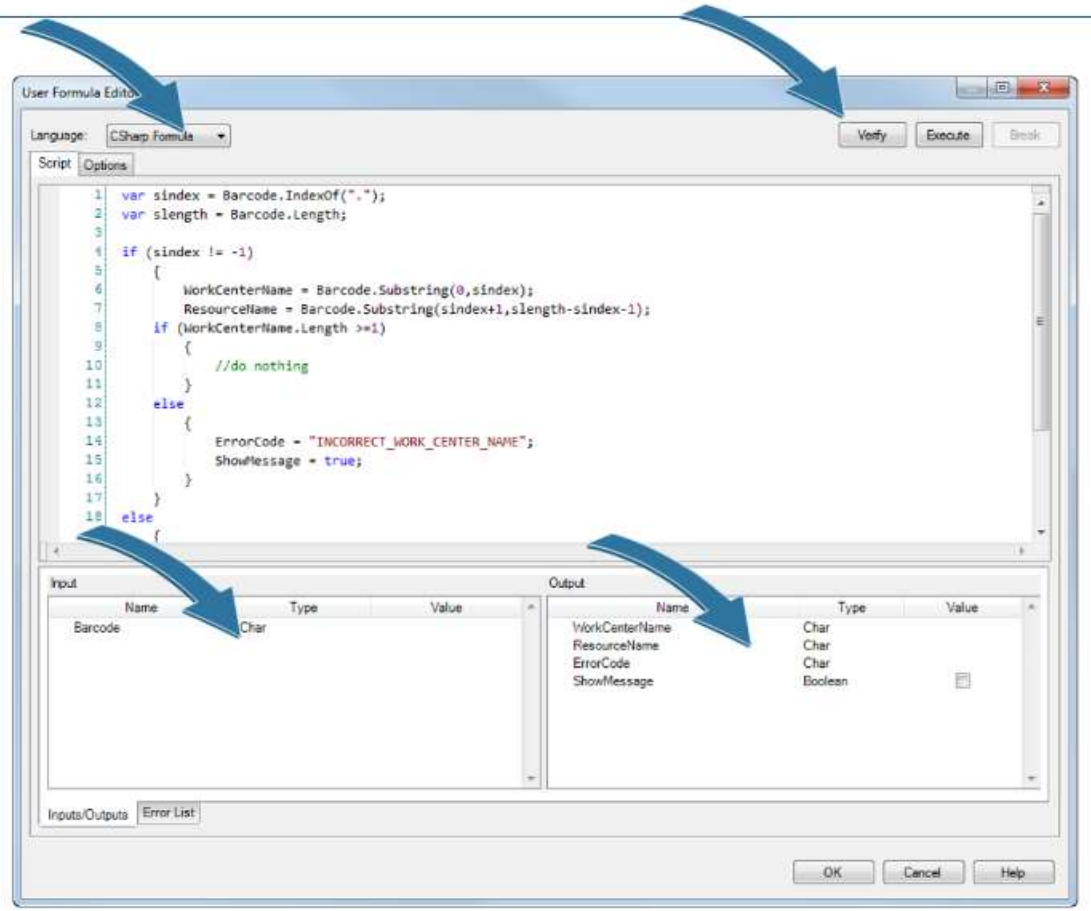
User Formula Editor

The look of the User Formula Editor is similar to the SQL Query Editor.

- There is a dropdown to determine the formula language (C# is the default for DELMIA Apriso 2017)
- You have Verify and Execute buttons which work similar to those in the SQL Query Function
- The test run inputs can be entered
- The test run results can be verified, too

For a User Formula function, it is advised to add function inputs and outputs before you paste the script. Otherwise you will get errors when verifying your script in the User Formula Editor. You can always leave the Editor, add the inputs/outputs, and re-verify.

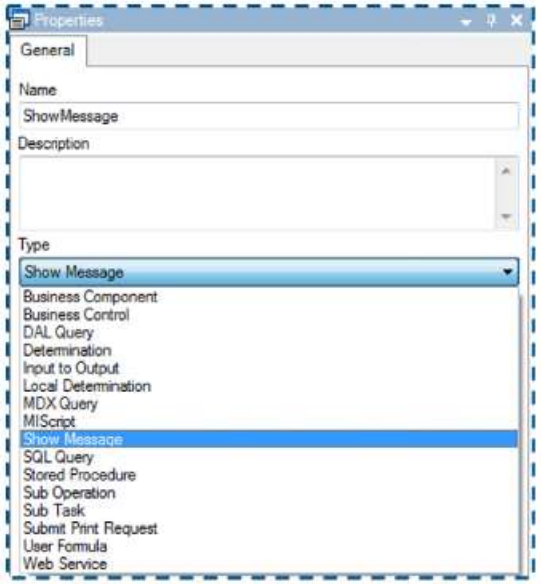
Always make sure the data types between inputs/outputs and the script variables match.

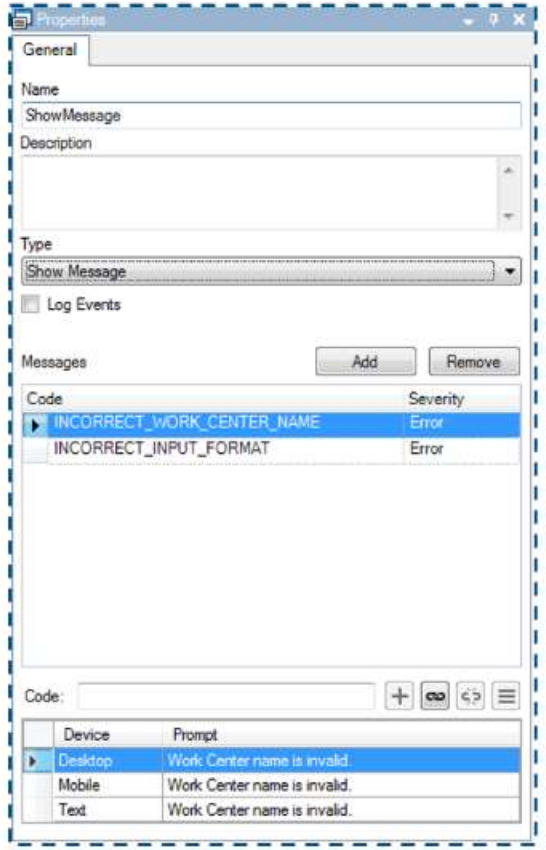


Show Message Function

The Show Message Function is used to define a set of messages which will be shown on the screen.

It is used for error messages, therefore when the function returns a message, the Standard Operation execution is stopped, and no data changes are committed to the database.





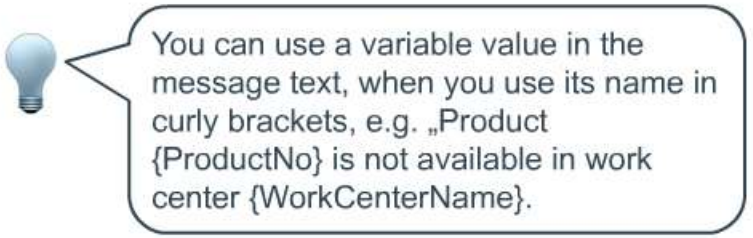
Show Message Function Properties

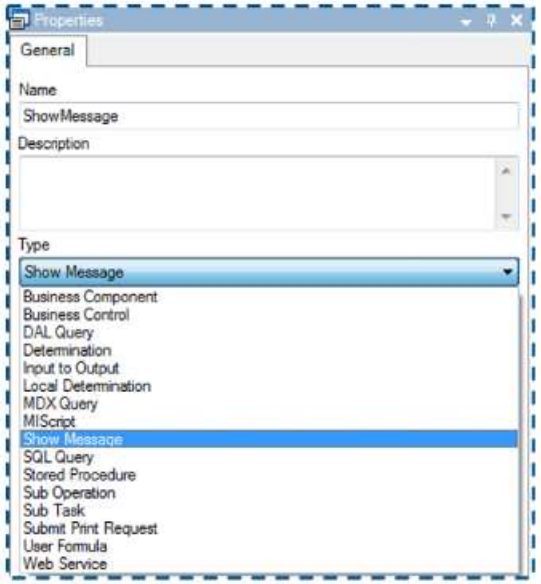
There are 2 main properties of the function:

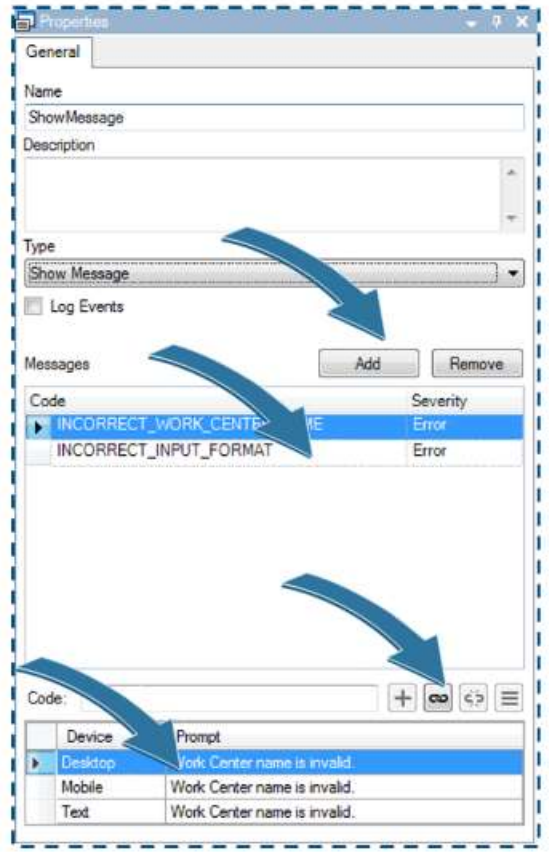
- Message Code - an error code, typically generated in a User Formula type function
- Prompt - which is the actual text users see on screen, when the error message is triggered

Here are the ways to configure messages:

- You can manage message codes using the Add and Remove buttons
- You can type or paste message codes in the function properties
- The corresponding message texts can be added from the Dictionary (if a message is a standard one, and is used in many places in the solution)
- The texts can be pasted individually, and adjusted for different type of devices



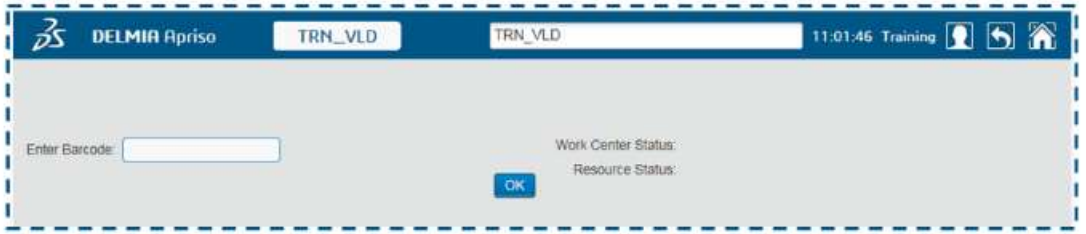




LAB 1: Create Screen to Validate Barcode Format

Task:

- Create Screen TRNXX_VLD
- Create and attach a View to capture the barcode
- Create and attach a View to display validation results



What you will learn:

- How to create the Screen for validating barcodes. You will configure all visual elements, and so you will set the stage for the next lab, in which you will configure the business logic with the user formulas and error messages

Training environment:

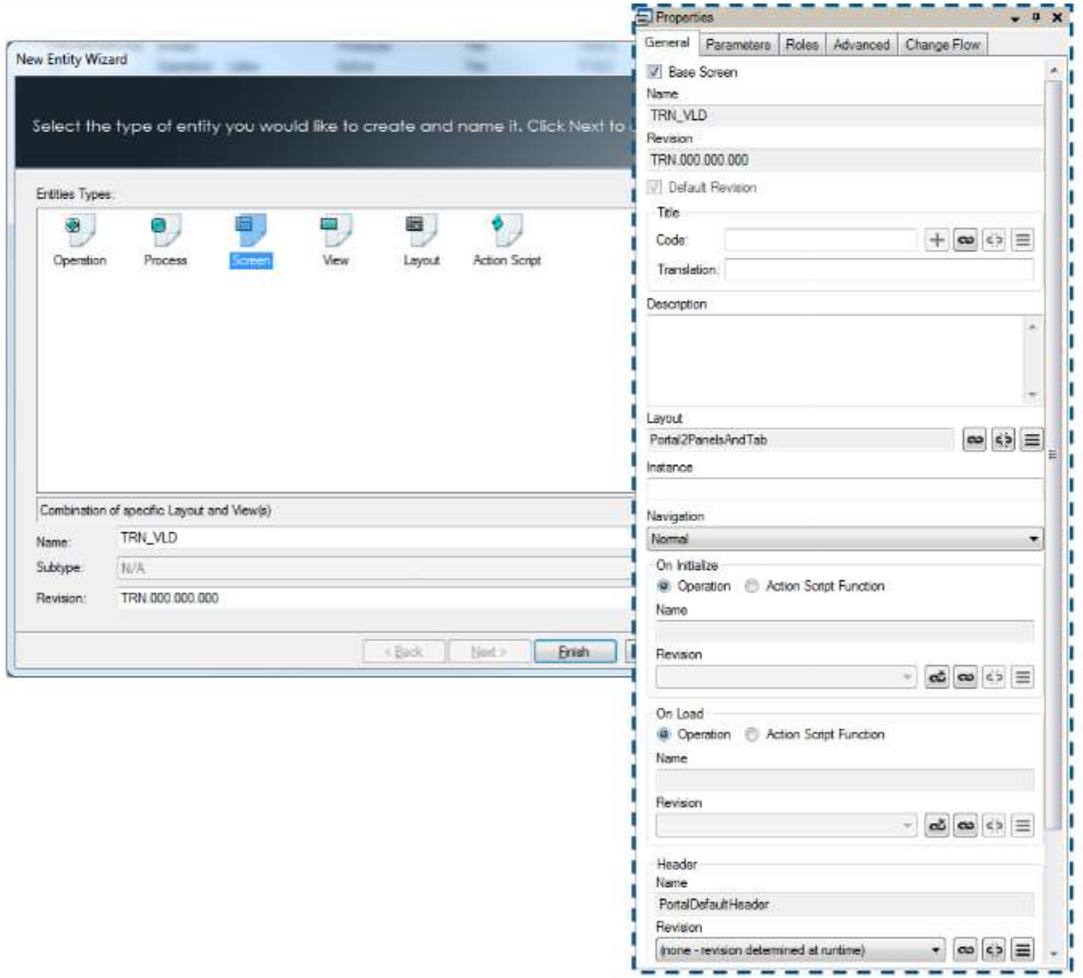
- In case of any technical problems, please contact DELMIA.Apriso.training@3ds.com

Remember to use the following to login and name Screens thorough this entire training:
- TRN<yourinitials> if your are an external self-paced learner
- TRN<yourtrigram> if you are a 3DS employee self-paced learner

20 min

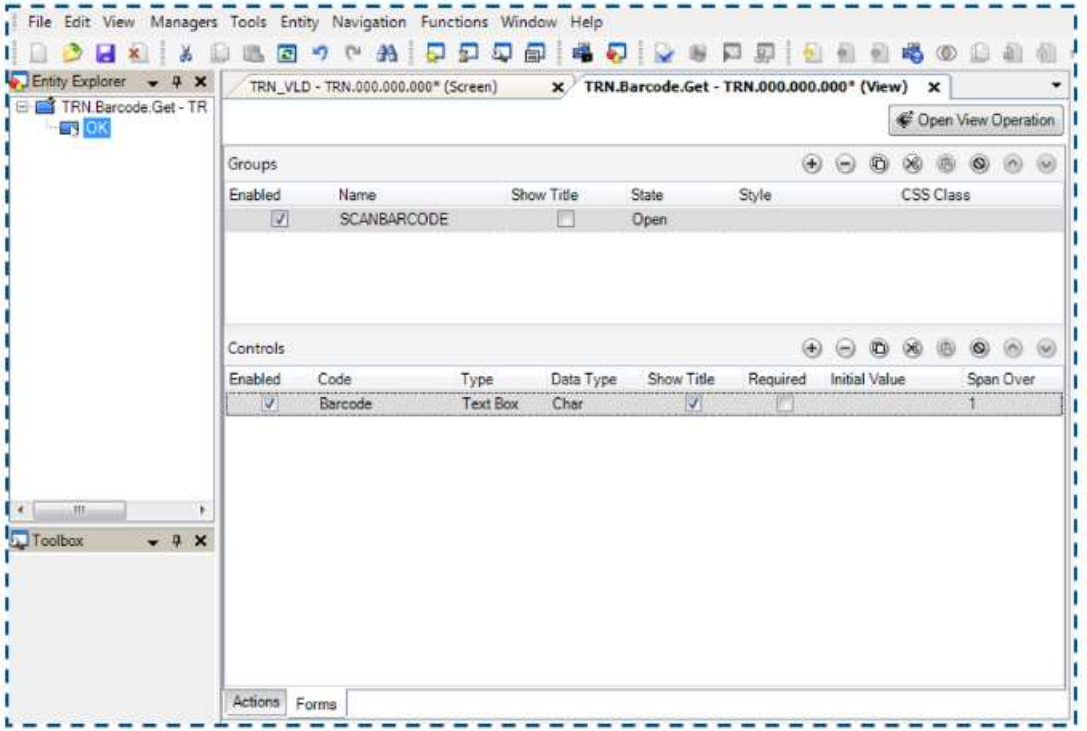
LAB 1: Create TRNXX_VLD Screen

- Create Screen TRNXX_VLD
  - Base Screen: checked
  - Layout: Portal2PanelsAndTab
  - Header: PortalDefaultHeader



LAB 1: TRNXX.Barcode.Get View Configuration

- In the context_left panel, copy and link the View PortalForm to TRNXX.Barcode.Get View
- Open the new View and delete GROUP2
- Rename GROUP1 to SCANBARCODE and uncheck the Show Title box
- Delete the check_box Control
- Change the text_ex Control to:
  - Code: Barcode
  - Show Title: checked
  - Title Translation: Enter Barcode
  - Type: Text Box
  - Data Type: Char
- In Actions tab, delete the BUTTON_2 Action
- Change the BUTTON_1 Action to:
  - Name: OK
  - Show Title: checked
  - Title Translation: OK
  - Type: Button (Primary)
- Save the View
- Change the View and the Screen to Prototype



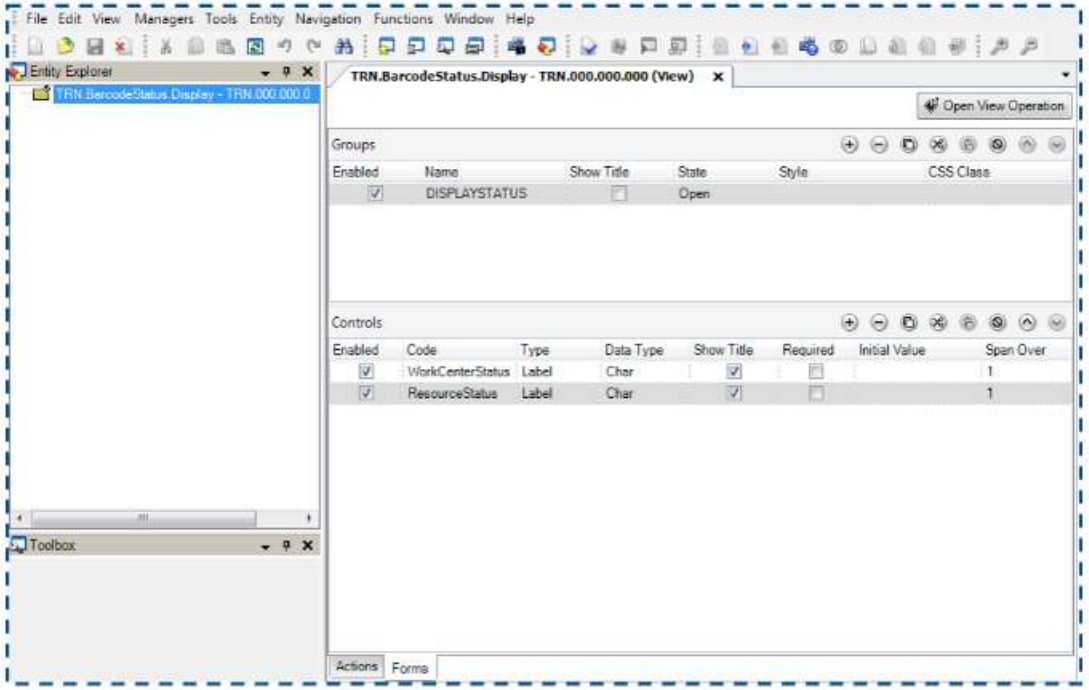
LAB 1: Test Run TRNXX_VLD Screen

- Test run the TRNXX_VLD Screen
- It should look like the screenshot to the right

In next steps, you will configure the right panel of the Screen.

LAB 1: TRNXX.BarcodeStatus.Display View Configuration

- In the context_right panel of Screen TRNXX_VLD, copy and link PortalForm View to TRNXX.BarcodeStatus.Display View
- Open the new View and delete GROUP2
- Rename GROUP1 to DISPLAYSTATUS and uncheck the Show Title box
- Change the text_ex Control to:
  - Code: WorkCenterStatus
  - Show Title: checked
  - Title Translation: Work Center Status
  - Type: Label
  - Data Type: Char
- Change the check_box Control to:
  - Code: ResourceStatus
  - Show Title: checked
  - Title Translation: Resource Status
  - Type: Label
  - Data Type: Char
  - Clean Initial Value and Default Value
- In Actions tab, delete both Actions
- Save the View and change to Prototype

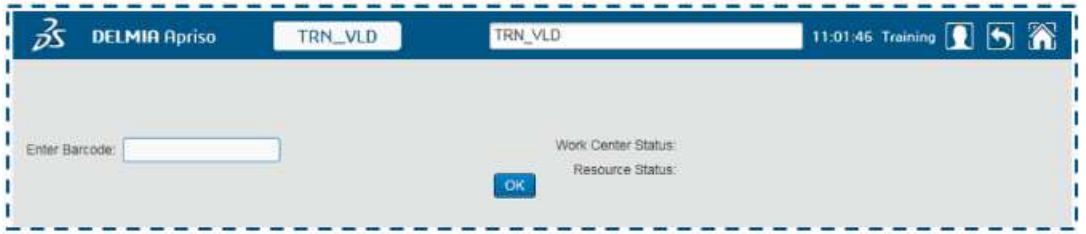


LAB 1: Test Run TRNXX_VLD Screen Again

- Test run the TRNXX_VLD Screen
- It should look like the screenshot to the right

- When you enter anything into the Enter Barcode field, and click OK, the Screen will reload, but nothing will happen

In the next lab, you will add business logic so that this screen becomes fully functional.



LAB 2: Create Business Logic to Validate Barcode

Task:

- Create a Standard Operation with the business logic to validate the barcode consisting of several functions

What you will learn:

- About function that uses C# formula to check the barcode input format for its correctness
- About function will use a different C# formula to inform the user on the Screen if the workcenter and resource (both included in the barcode) exist or not
- A Show Message function will be responsible for displaying error messages if needed

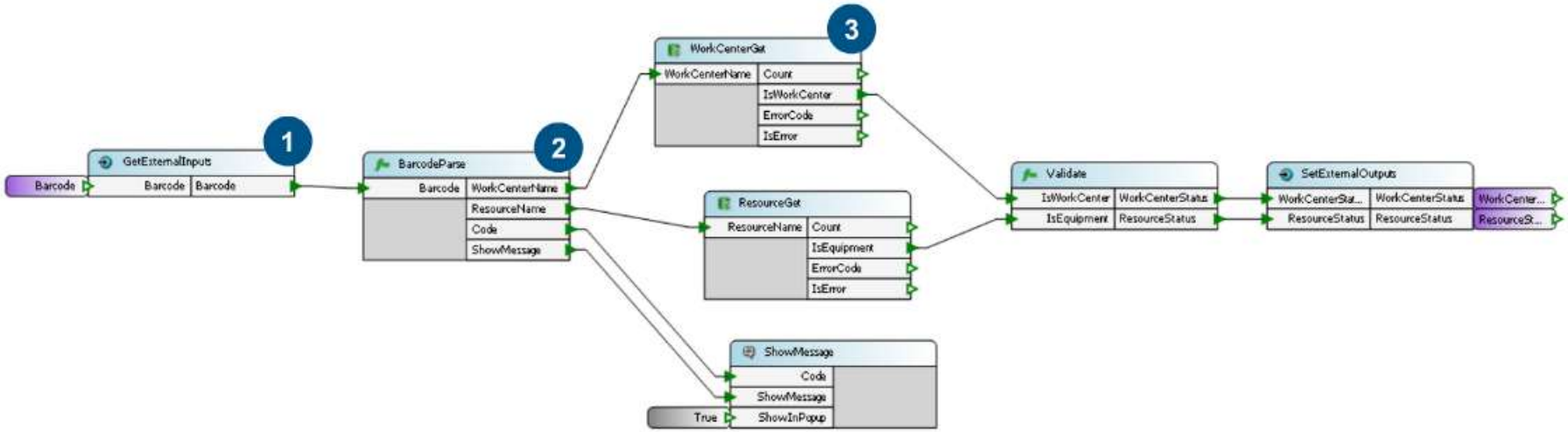
Training environment:

- In case of any technical problems, please contact DELMIA.Apriso.training@3ds.com

LAB 2: Functions Description, part 1

The Operation is quite complex, so let's take a look at the final version, along with why it is structured this way. The Operation has seven different functions:

1. GetExternalInputs - this function gathers the barcode information provided to the system
2. BarcodeParse - the purpose of this function is to take the barcode, check if exists (which separates work center and resource), extract the names of the work center and resource as separate variables, and return error messages if the barcode is formatted incorrectly
3. WorkCenterGet - this function verifies if the workcenter given in the barcode exists in the database. It will return a 1 if the workcenter can be found in the database. This 1 will later be translated into a meaningful message on the TRNXX_VLD Screen



LAB 2: Functions Description, part 2

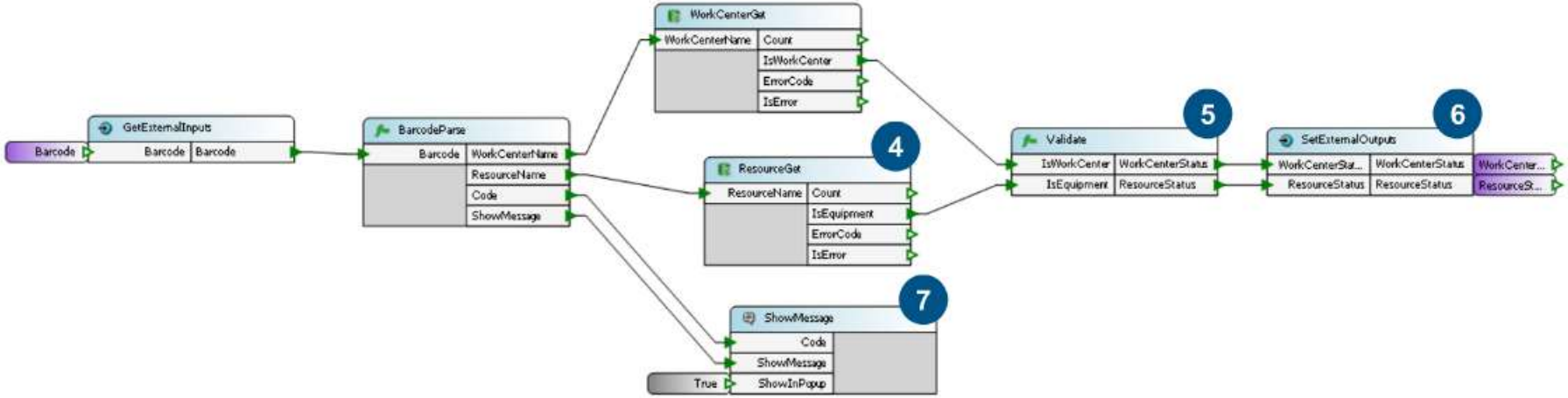
4 ResourceGet - the purpose of this function is to confirm if the resource (equipment) from the barcode can be found in the database

5 Validate - this function validates the information about the workcenter and resource are present and displays the end result (positive)

6 SetExternalOutputs - the purpose of this function is to return information when the workcenter and resource are correct

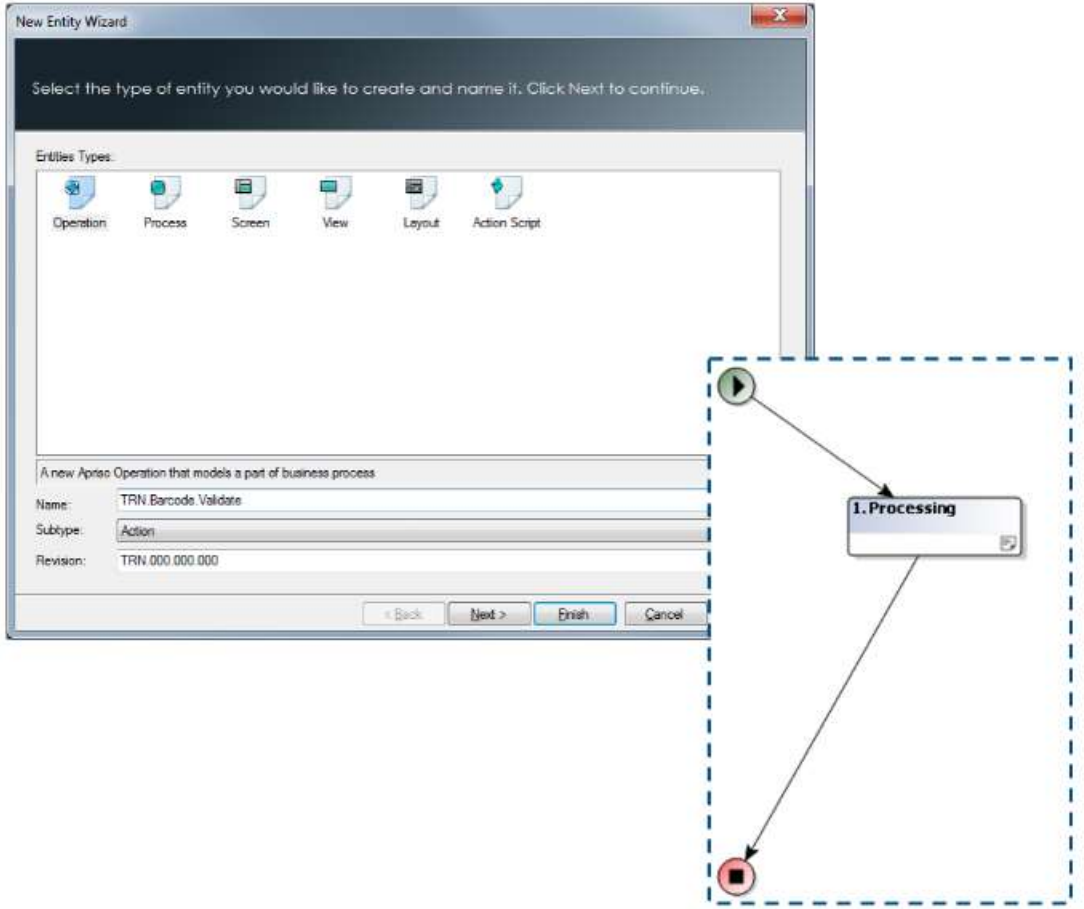
7 ShowMessage - this function will return an error messages when the barcode is structured incorrectly

Take a look at the final Operation structure and… let's get to work!



LAB 2: Create TRNXX.Barcode.Validate Operation

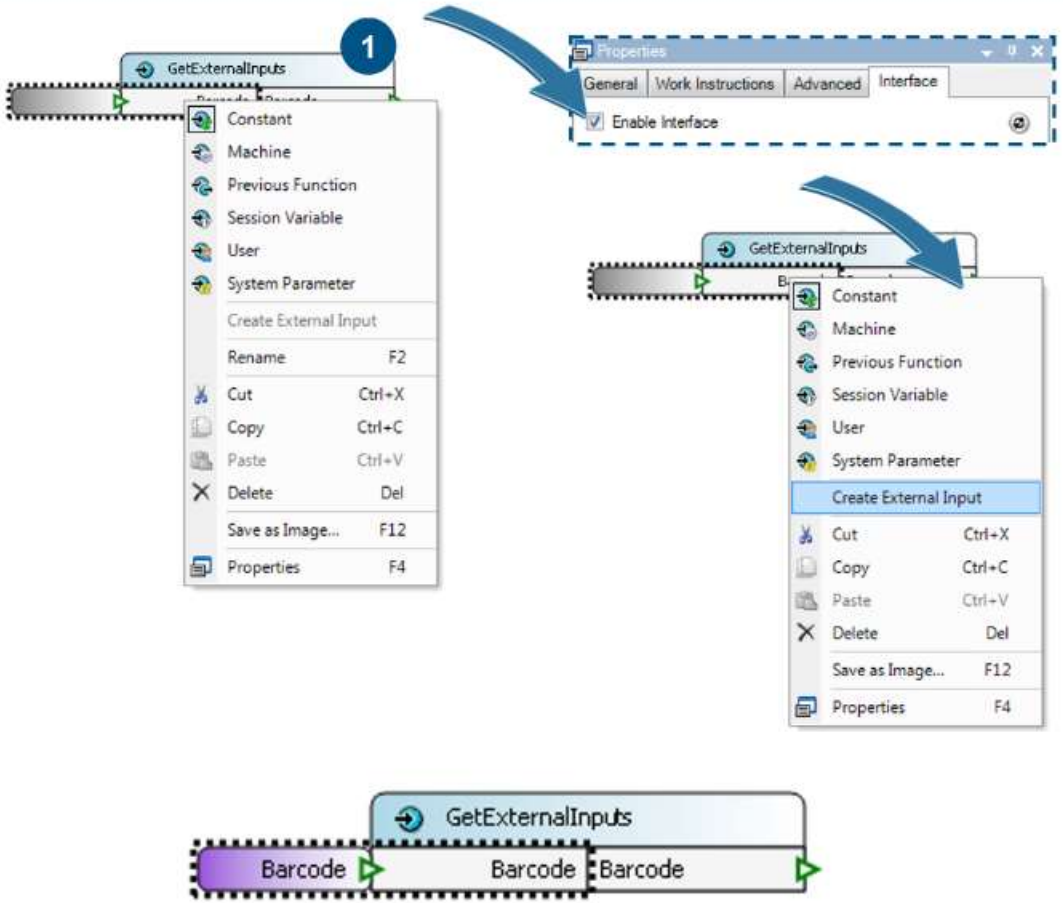
- Create a new Operation:
  - Name: TRNXX.Barcode.Validate
  - Subtype: Action
- It has only one Step, change its name to: Processing
- Make sure the Step navigation is complete



LAB 2: Add GetExternalInputs Function

- Go into the Step
- Add a new Function:
  - Name: GetExternalInputs
  - Type: Input to Output
- Add a Pair to the function:
  - Name: Barcode

  (It has to be identical to the Barcode Form Control on the TRNXX.Barcode.Get View.)
- Right click the function input and select Create External Input
- If allowing external input is greyed out, go to Operation Properties, Interface Tab, and check the Enable Interface box

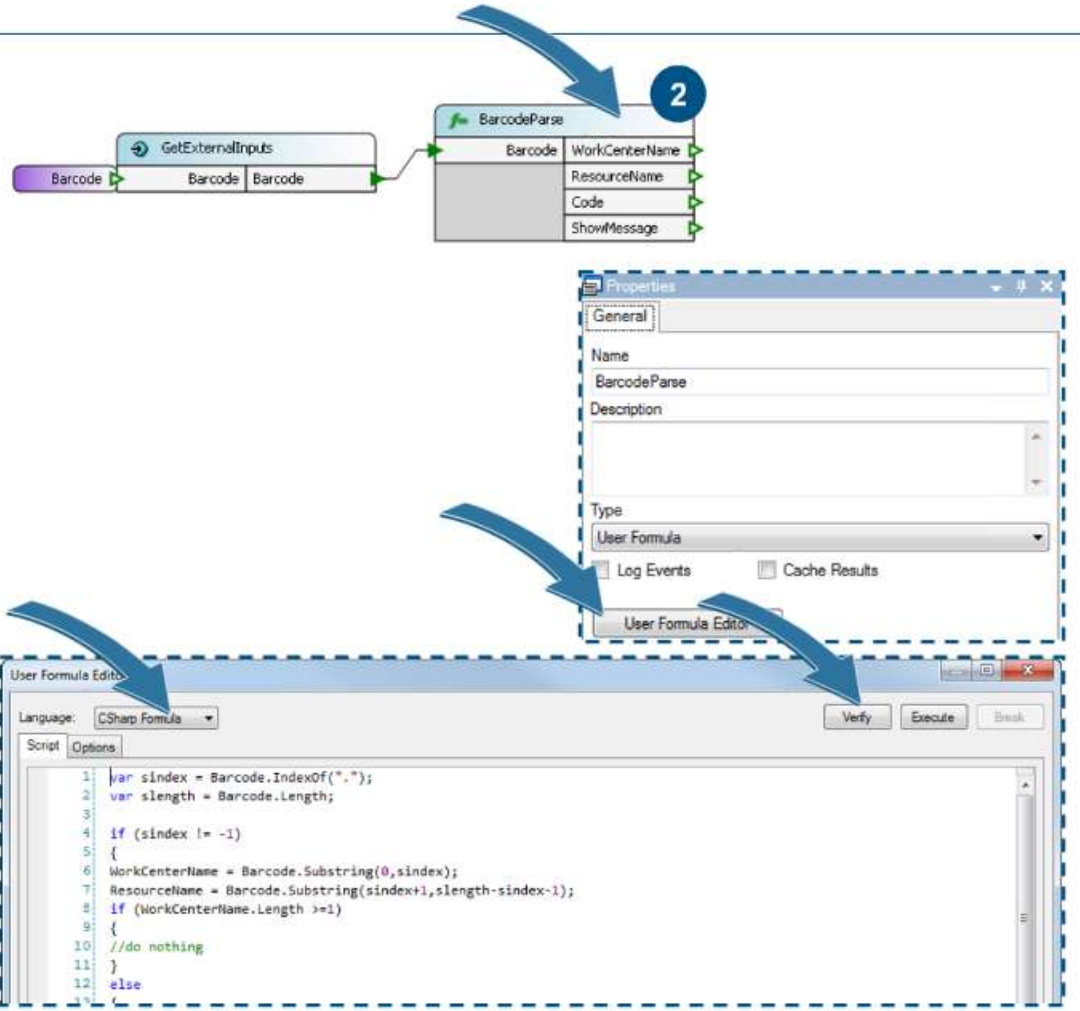


LAB 2: Add BarcodeParse Function

- Add a new Function:
  - Name: BarcodeParse
  - Type: User Formula
- Add 4 Outputs to the function:
  - Name: WorkCenterName, Type: Char
  - Name: ResourceName, Type: Char
  - Name: Code, Type: Char
  - Name: ShowMessage, Type: Boolean
- Drag and Drop the outputs from the GetExternalInputs and as inputs to the BarcodeParse function to match the screenshot
- In the BarcodeParse function's properties, click on the User Formula Editor
- Make sure the Language is CSharp Formula, paste the C# script from scripts:

SCRIPT FILE: Desktop/Training Materials/Level 1

- Click Verify to make sure the formula is valid
- Close User Formula Editor, and save the Operation



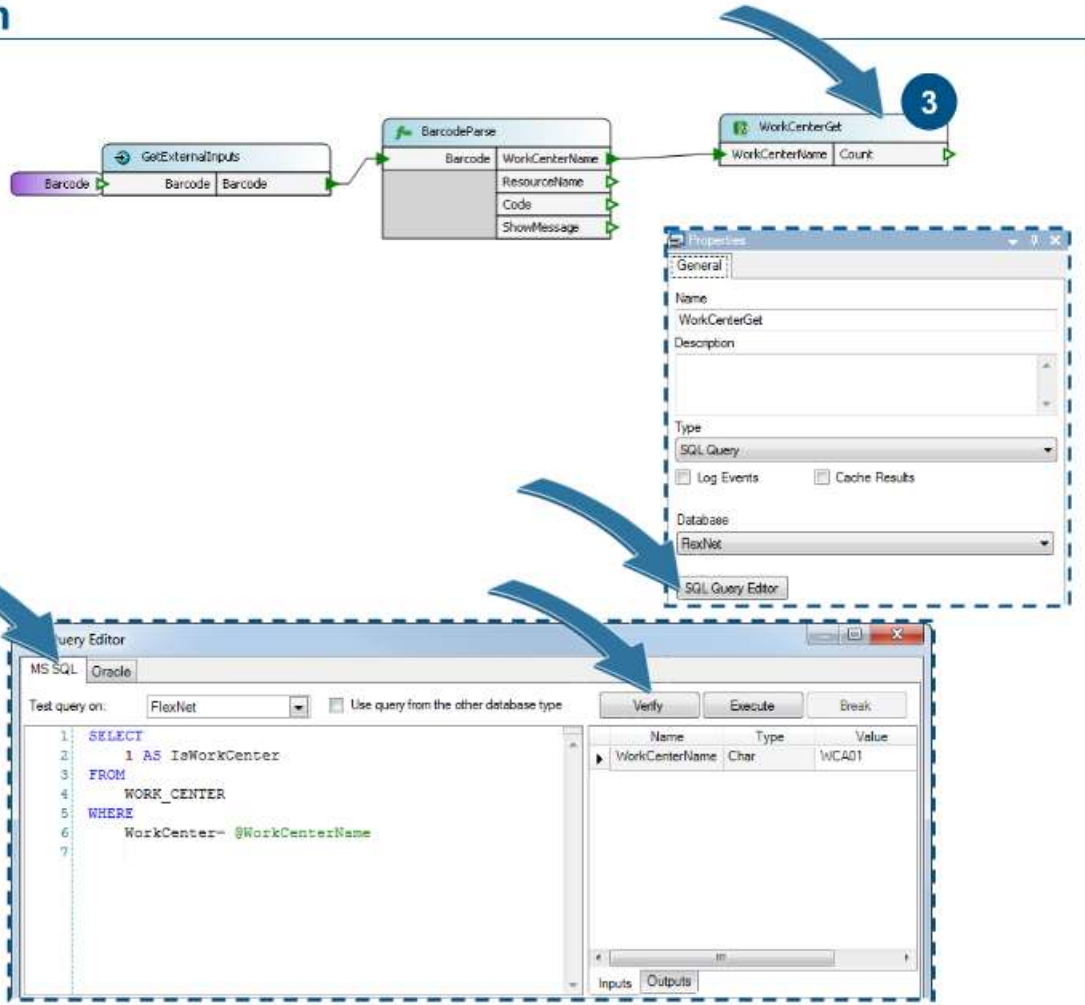
LAB 2: Add WorkCenterGet Function

Let's move to checking the barcode info against workcenter and resource data.

- Add a new Function:
  - Name: WorkCenterGet
  - Type: SQL Query
- Drag and Drop the WorkCenterName output from the BarcodeParse function as an input to the WorkCenterGet function
- In the WorkCenterGet function's properties, click on the SQL Query Editor
- Make sure the selected tab is MS SQL
- Paste the SQL script from the scripts:

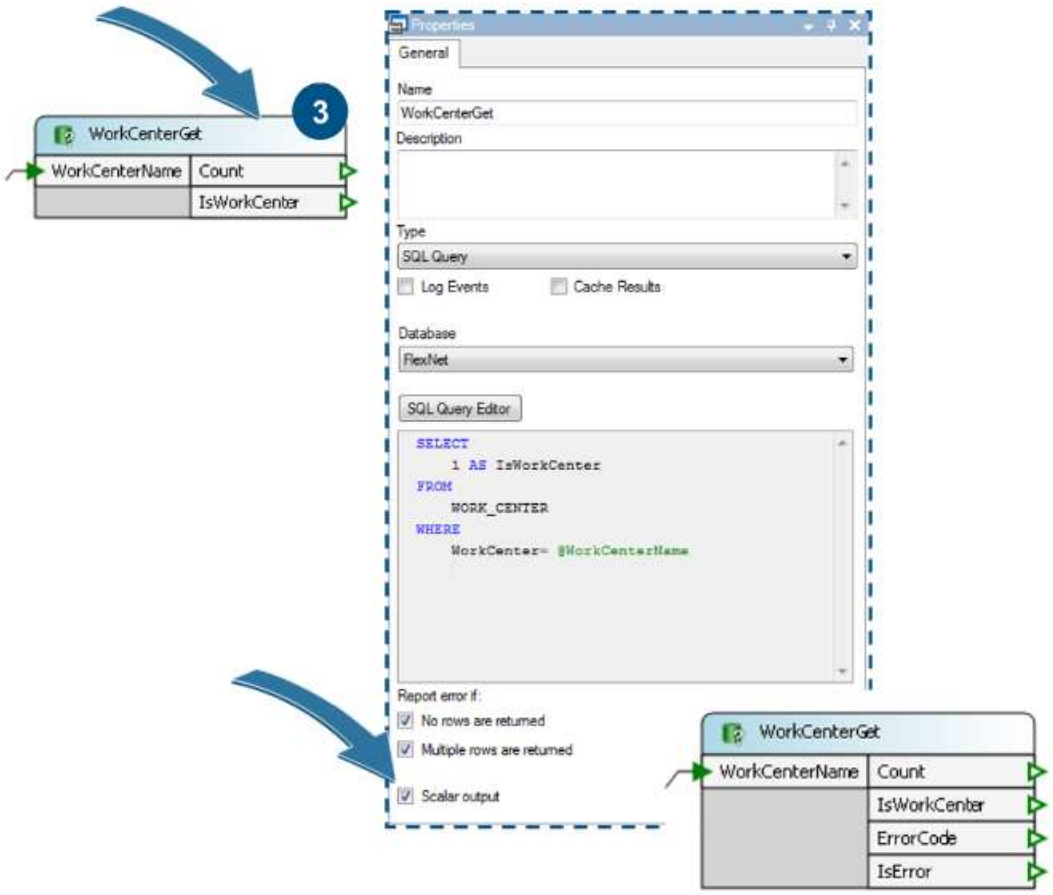
SCRIPT FILE: Desktop/Training Materials/Level 1

- Click Verify to make sure the script is valid



LAB 2: Configure WorkCenterGet Function

- Close the SQL Query Editor
- Note the WorkCenterGet function received a new output. It was generated automatically based on what was added in the SELECT statement of the SQL query
- Set IsWorkCenter Data Type to Integer Scalar
- Check the Scalar output box at the bottom of function properties
  - This will make the system add 2 more outputs to the SQL query function automatically. They can be used to generate error message should the query return either no or multiple records



LAB 2: Add ResourceGet Function

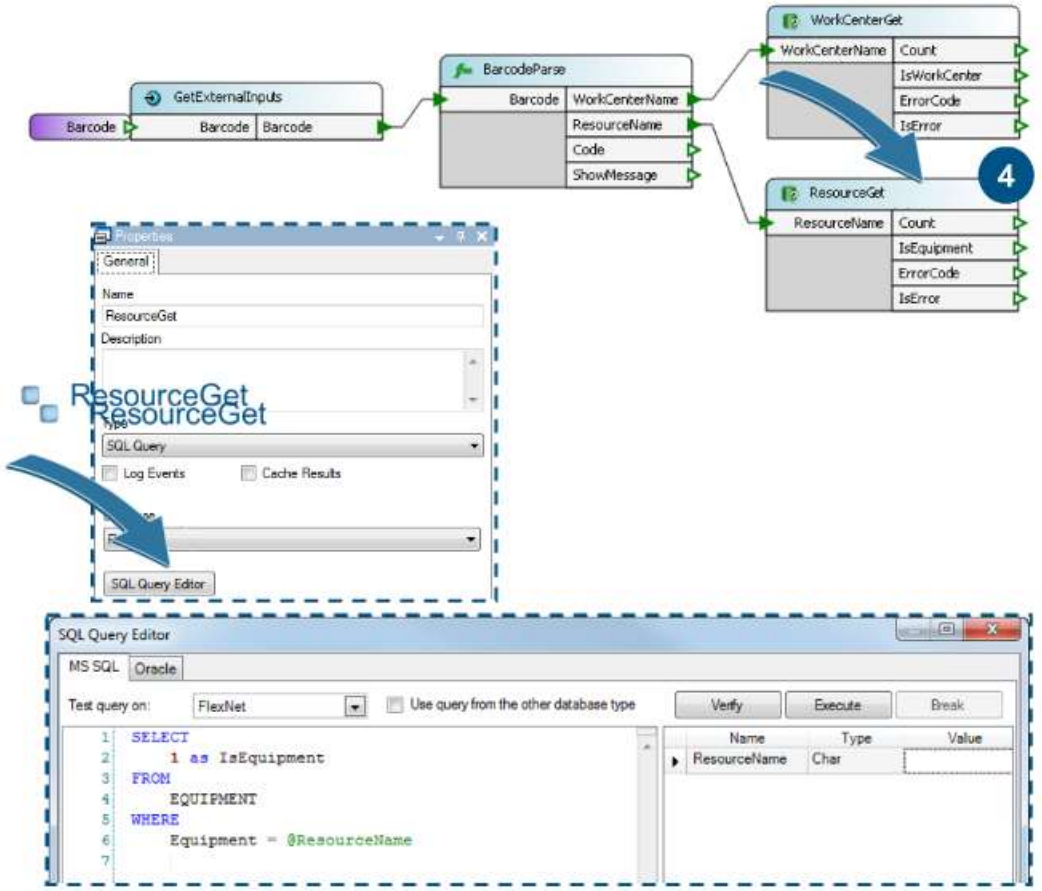
Now add a function to check the resource name.

- Add a new Function:
  - Name: ResourceGet
  - Type: SQL Query
- Drag and Drop the ResourceName output from the BarcodeParse function as an input to the ResourceGet function

- In the function's properties, click on the SQL Query Editor.
- Make sure the selected tab is MS SQL
- Paste the SQL script from the scripts:

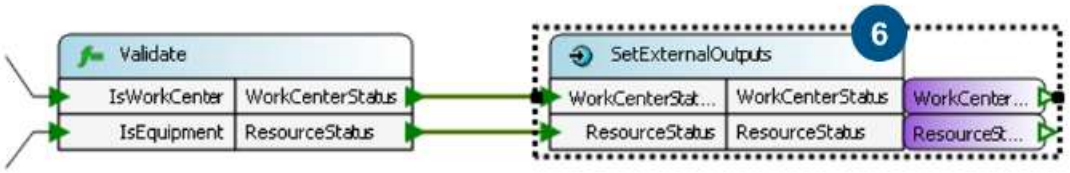
SCRIPT FILE: Desktop/Training Materials/Level 1

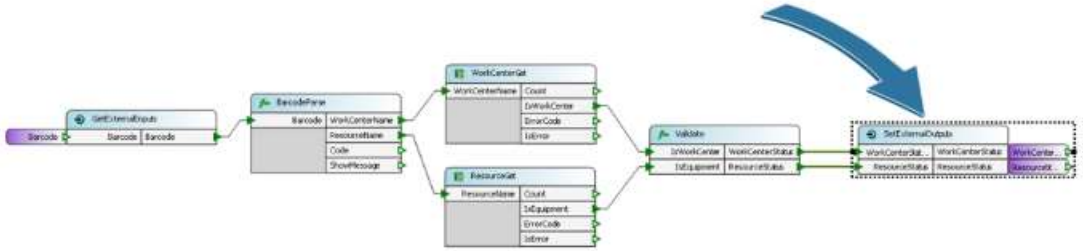
- Click Verify to make sure the script is valid
- Close the SQL Query Editor and check the Scalar output box
- Make sure that the IsEquipment Data type is set as Integer Scalar



LAB 2: Add SetExternalOutputs Function

- Add a new Function:
  - Name: SetExternalOutputs
  - Type: Input to Output
- Add 2 Pairs:
  - Name: WorkCenterStatus, Type: Char
  - Name: ResourceStatus, Type: Char
- Connect the outputs with corresponding names from the Validate function to respective inputs in the SetExternalOutputs function
- Right-click on each output in the SetExternalOutputs function and select the Add External Routing option

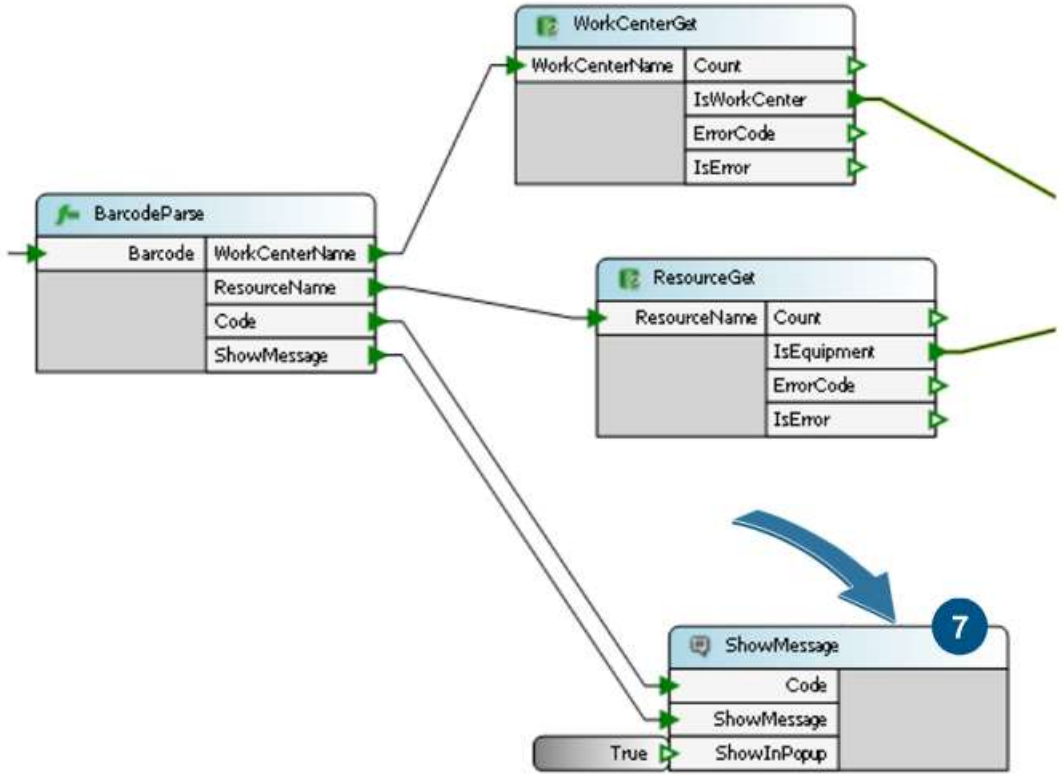




LAB 2: Add ShowMessage Function

- Add a new Function:
  - Name: ShowMessage
  - Type: Show Message
- Connect the ErrorCode output from BarcodeParse function to the Code input of the ShowMessage function
- Connect the ShowMessage output from BarcodeParse function to the ShowMessage input of the ShowMessage function

The ShowInPopup input controls if the error is shown in a red background bar at the top of the screen, or in a popup. You may set this input to True.

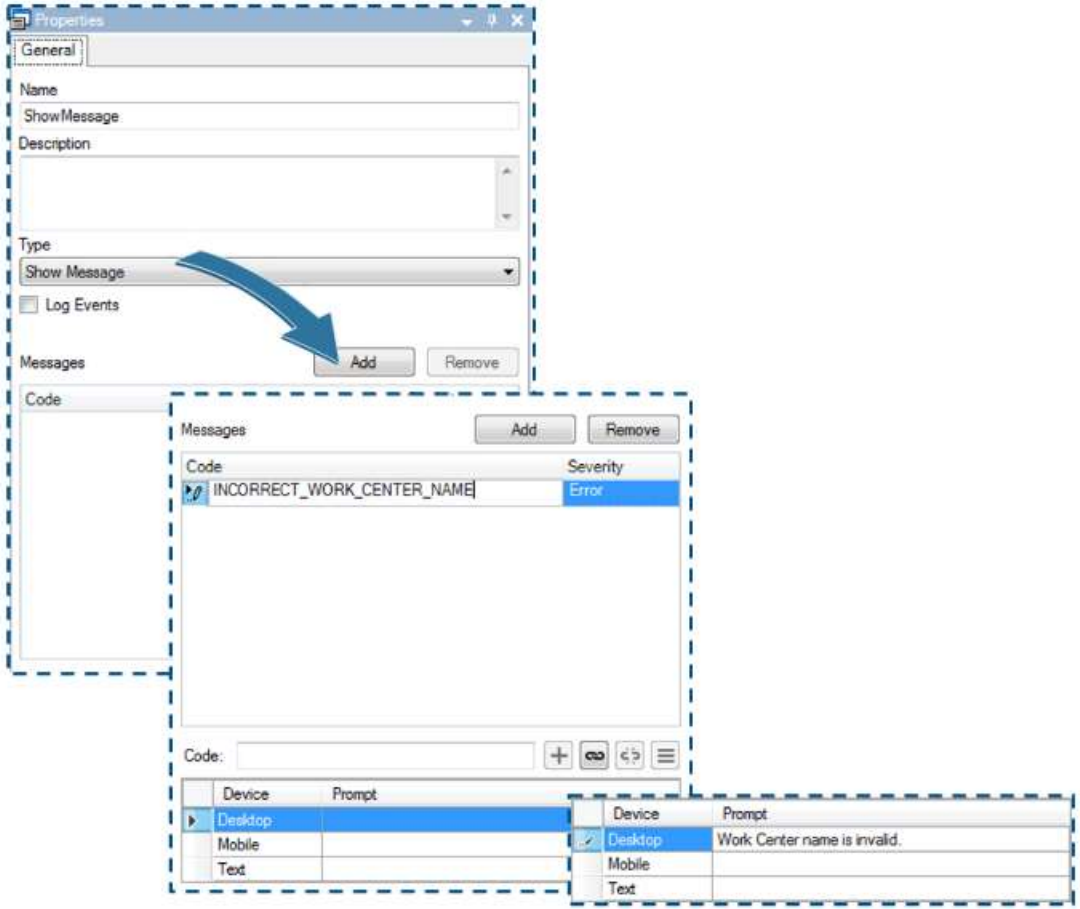


LAB 2: Add Error Messages

Now you will add the actual error messages to be shown, depending on what error code the BarcodeParse function returns. If you check the C# formula there, you will see they are:

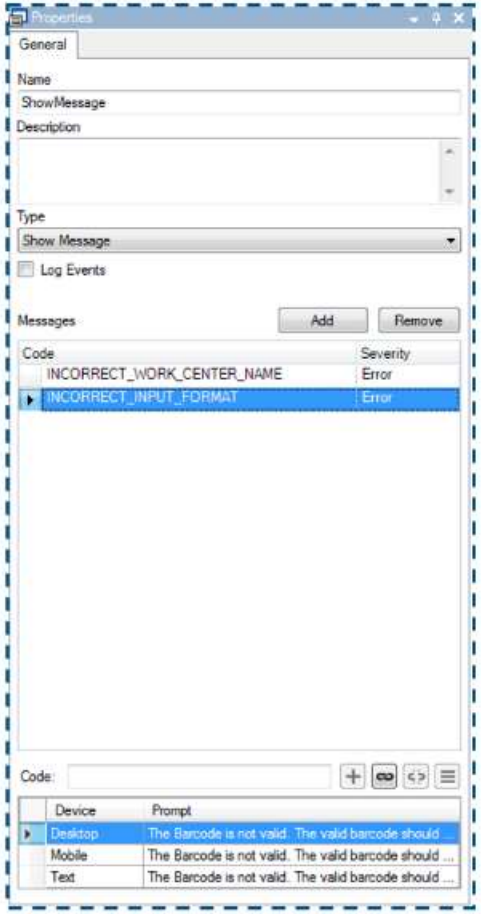
- INCORRECT_WORK_CENTER_NAME and
- INCORRECT_INPUT_FORMAT

- Go to ShowMessage function properties
- Click the Add button
- Paste INCORRECT_WORK_CENTER_NAME as the code
- The text which you will want to show to users for this error is Work Center name is invalid
- Paste it into the Prompt field in the Desktop row.
- When you hit Enter, this text will be populated for Mobile and Text rows, too



LAB 2: Add Text Error Messages

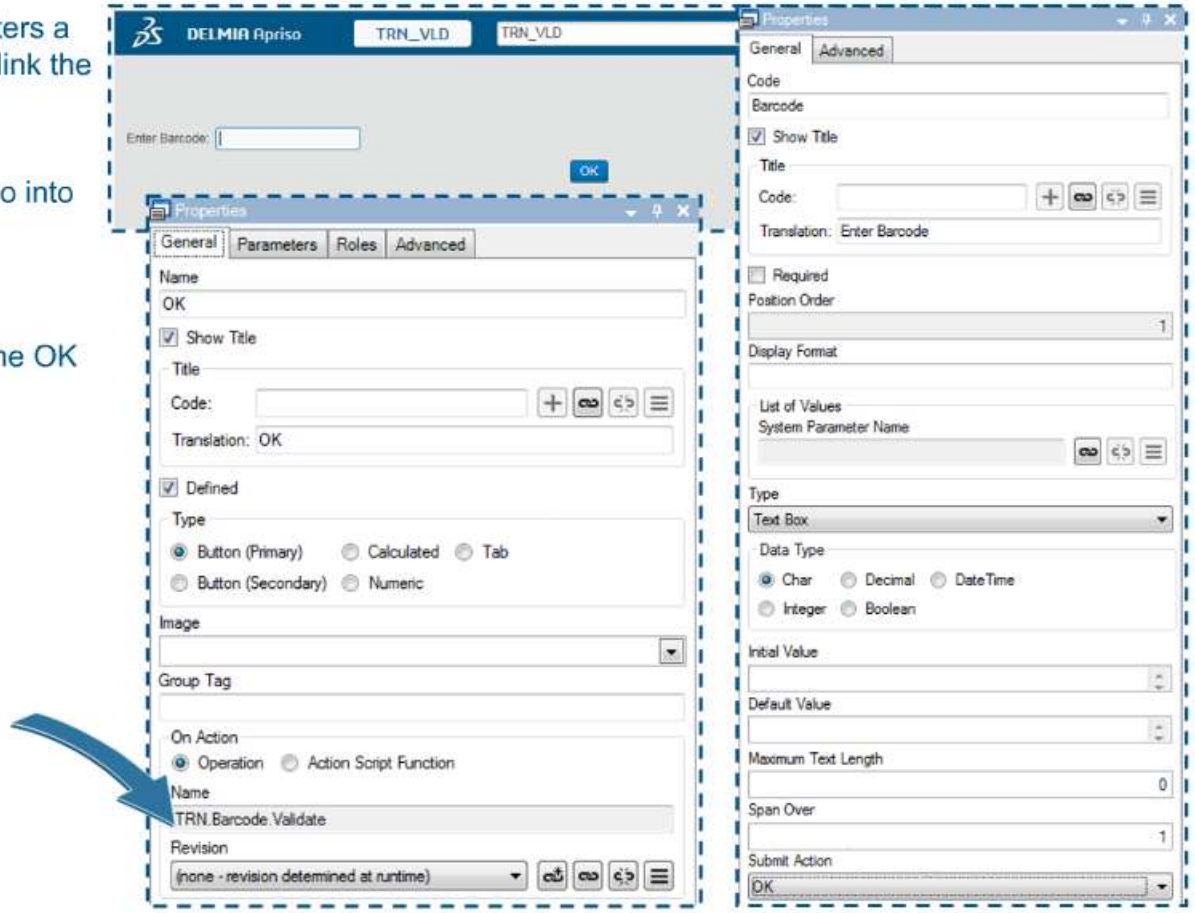
- Click the Add button again
- Paste INCORRECT_INPUT_FORMAT as the code
- Paste „The Barcode is not valid. The valid barcode should have the structure WorkCenter.Resource" in the Prompt field on the Desktop
- Save the Operation and set to Prototype



LAB 2: Link Operation TRNXX.BarcodeValidate

The Operation will be triggered when a user enters a barcode, and clicks on OK button. You need to link the Operation to this button.

- Go to the TRNXX.Barcode.Get View, and go into OK Action properties
- Set On Action option to Operation and link TRNXX.BarcodeValidate
- Go to Barcode Control properties and set the OK as the Submit Action
- Save the View



LAB 2: Test Run TRNXX_VLD Screen

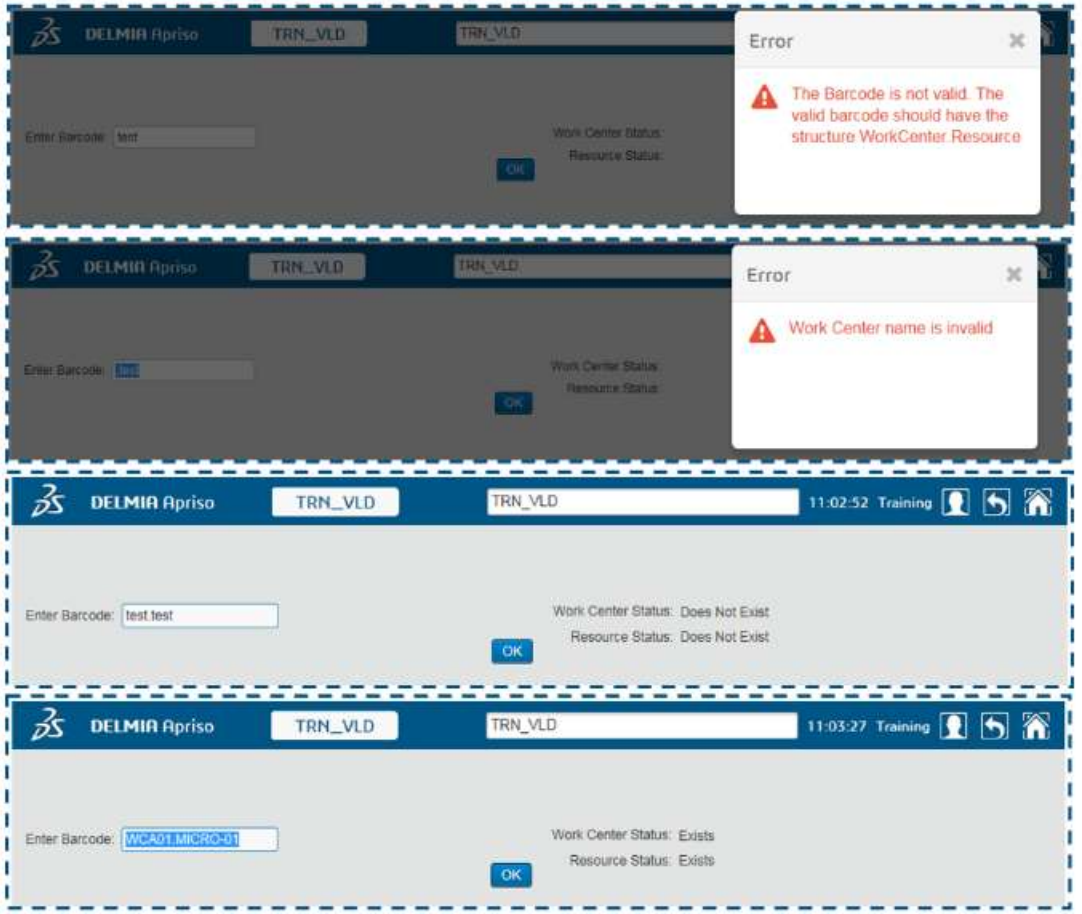
- Test run the TRNXX_VLD Screen

1. When you enter any text, without a full stop, you will get an error message: „The Barcode is not valid. The valid barcode should have structure WorkCenter.Resource"

2. When you enter a text with a full stopas the first character, you will get an error: „Work Center name is invalid."

3. When you add any text with a full stop in the middle, you get no error message, but will most probably see that neither the work center nor the resource exist

4. To see that the work center and the resource exist in the system, use this string as the barcode: WCA01.MICRO-01



Summary

Functions are key components of Standard Operations which constitute the business logic flow through the system.

You have just seen how you can use User Formula functions, SQL Query functions, and Show Message functions to configure business logic.

Process Builder has several more function types, and you will find more information about them in Process Builder Help.