Configuring Business Logic - Business Components Online Training

Abstract:

This training will delve deeper into Business Components. You will learn about the Business Component Repository, and how it is structured. You will also learn how to configure Business Components to accomplish a specified task.

Detailed objectives. After the training you will:

- Find and understand the Business Component Repository
- Configure Business Components in an operation
- Troubleshoot and test results of using Business Components

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:

- Intermediate Process Builder users

What Is a Business Component

Business Components are units of functionality that manage a single business concept. They can be compared to mini-programs, which require certain inputs and generate outputs, provide defined processing and validations.

Business rules are encapsulated in such an entity, which allows them to be directly integrated into a Process or Standard Operation. Examples of pieces of business logic that can be delivered as Business Components are:

- clocking in/out
- converting local time to UTC
- Reporting production
- validating an employee's password
- Creating a container

DELMIA Apriso comes with a hundreds of readily available Business Components (BCs) out-of-the-box.

These BCs cover categories as listed in a repository tree shown here.

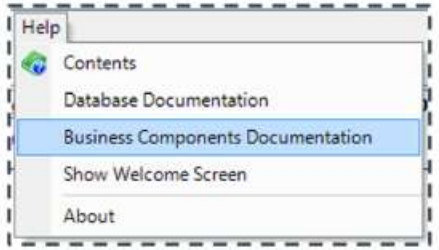
Business Component Repository

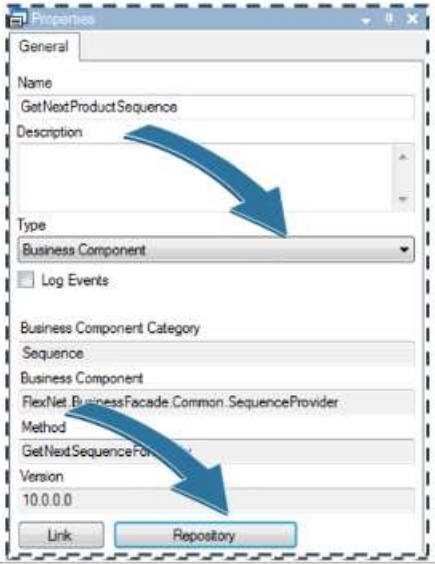
You can find all necessary details regarding a particular Business Component in the Business Component Repository. The documentation will cover:

- General purpose and description of BC
- Parameter information
- Validation (if applicable)
- Affected database tables and their columns

The Business Component Documentation can be found:

- either from the Process Builder menu bar under Help, or
- from the function properties of a Business Component Function. In this case, click the Repository button





How to Use Business Components

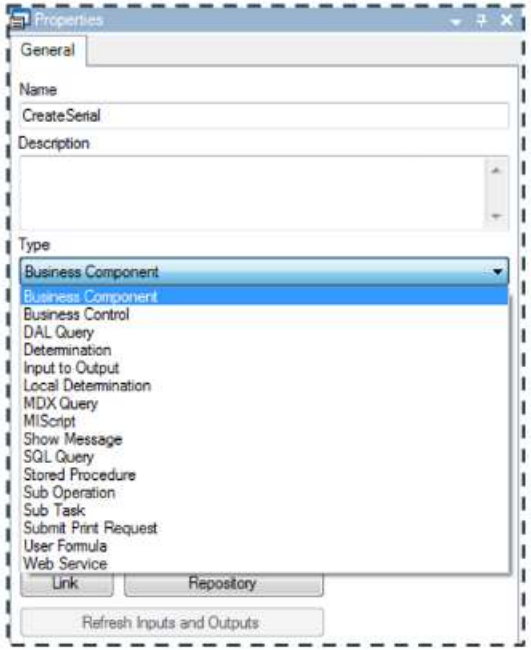
If you want to use a Business Component, select the Business Component function type on a new function. You will be presented with a list of BCs to choose from.

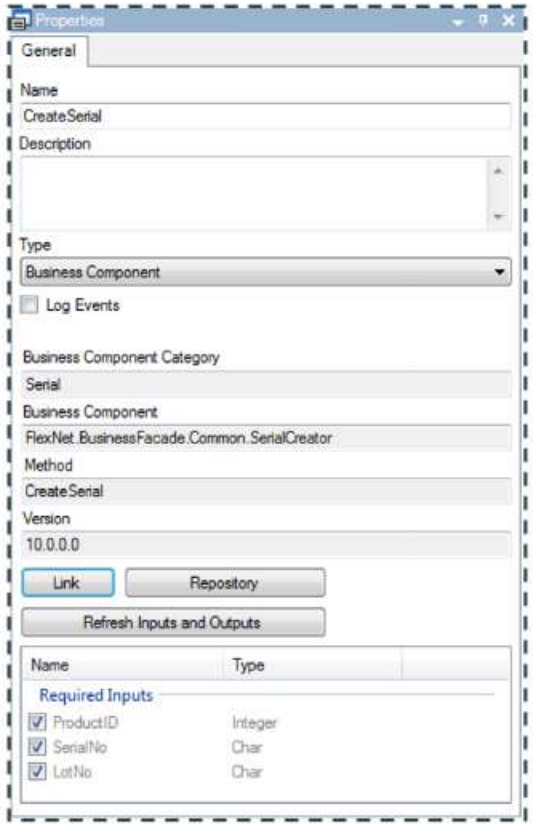
Once you make the change, the function properties will change automatically, and so will the name of the function (to reflect the name of the Business Component selected). If you want to have a different name for the function, edit it after selecting the BC.

The new BC function will appear in Process Builder, and you will be able to edit the inputs and outputs.

The BC inputs can be required or optional. Also, some BCs are very specific, and some are more general.

The more general ones can serve slightly different purposes depending on the set of inputs you will use on it. You will see an example in the lab part of this training module.





How Do You Know Which Business Components to Use When

Each Business Component affects at least one database table. Therefore which BCs you use in a project depends heavily on the data model of the project.

Once you know the data model, work with the Business Components documentation to find the BCs you should use.

Sometimes, the Detailed Design document (which is part of the project documentation) will include information regarding which BCs should be used in particular scenarios. This information will be provided especially in the case where detailed instructions are needed to correctly address a requirement from the customer.

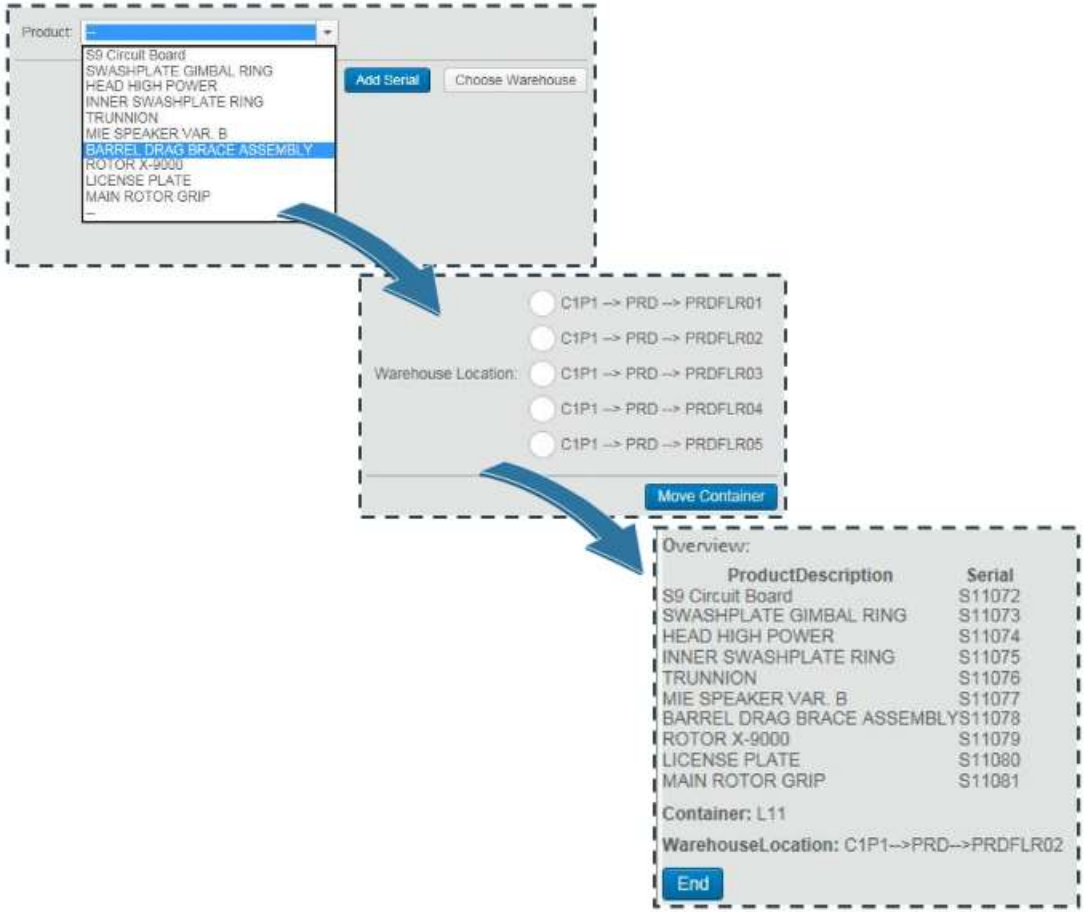
Case Study For Labs

In the following labs you will configure a simplified serial-tracked production flow:

- The first step will be to build screens to generate serial numbers for produced products (you may choose one, two, or as many as you like)
- After you are finished assigning serial numbers to products, select Choose Warehouse, which will create a new container, move the serialized products into that container, and move the new container to a storage location in a warehouse
- Finally, a summary screen is displayed

In order to achieve this, you will use 4 different Business Components, and you will understand how they work and how they can be used.

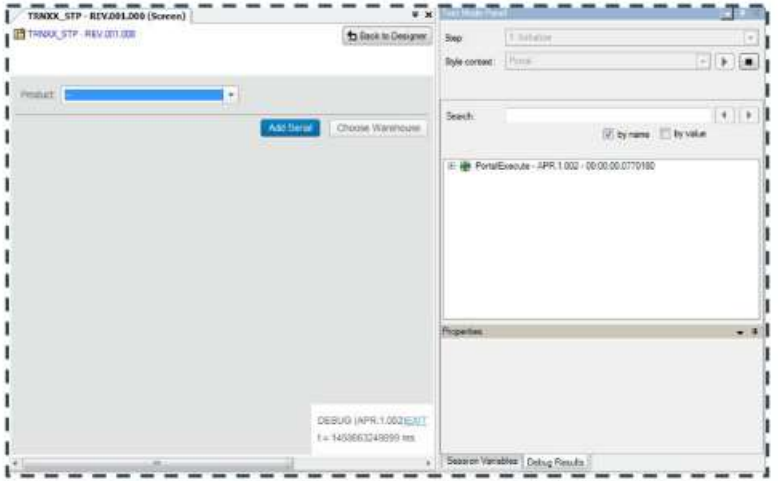
Also, you will work with the Debug Results in the test run to see if the Business Components worked, what data they processed, and how.



LAB 1: Create a Screen to Assign Serial Number to a Product

Task:

- Build a Screen where the user will be able to pick a product from a drop down list, and upon clicking a button, the system will generate a serial number, and assign it to the selected product piece
- The Screen will require an On Action Operation to generate the serial number. This Operation will use two Business Components: GetSequences, and CreateSerial



What you will learn:

- How to make drop-down lists that generate action
- How to use GetSequence and CreateSerial Business Components

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

35 min

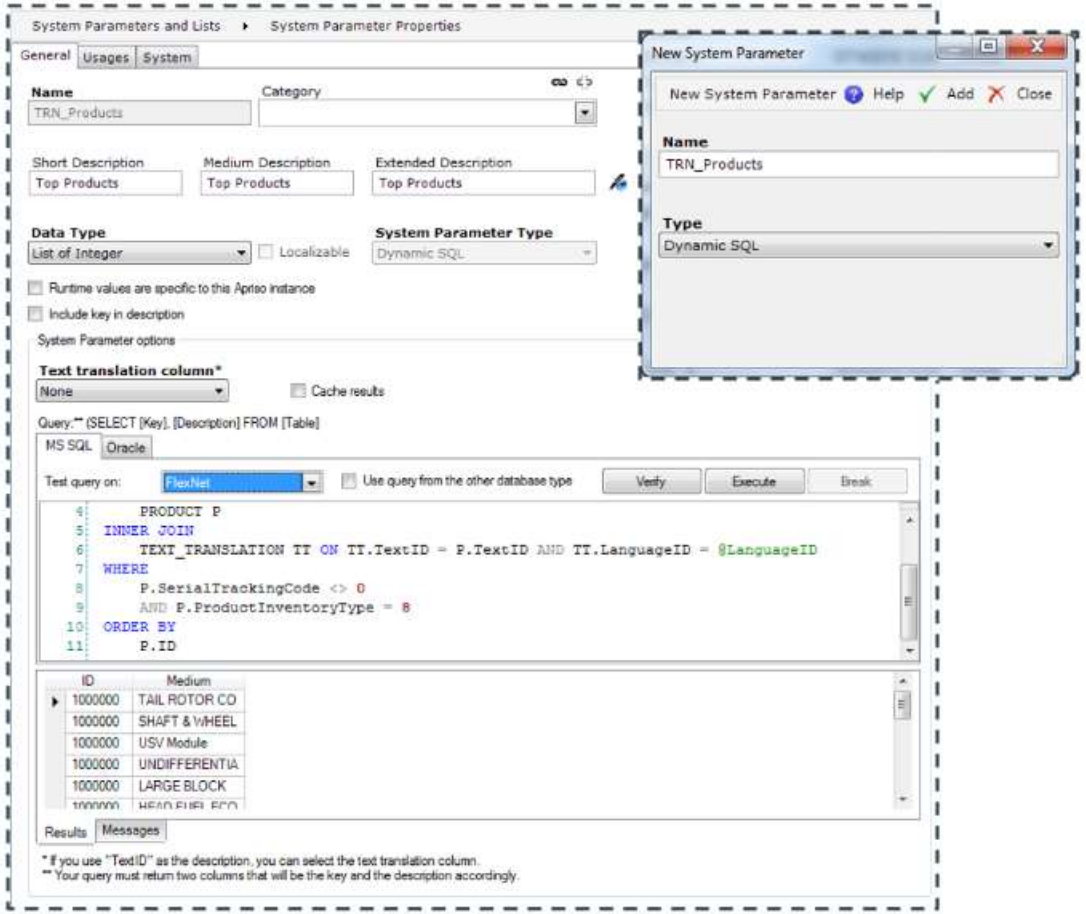
LAB 1: Create Product List

This list will populate the drop down to select the product for which you want to create a serial number.

- In Process Builder go to the Managers menu, and select System Parameters and Lists Manager
- Click Add and in new popup window enter:
  - Name: TRNXX_Products
  - Type: Dynamic SQL
- Set the properties as follows:
  - Short Description: Top Products
  - Data Type: List of Integer
- Put the SQL script from scripts:

SCRIPT FILE: Desktop/Training Materials/Level 1

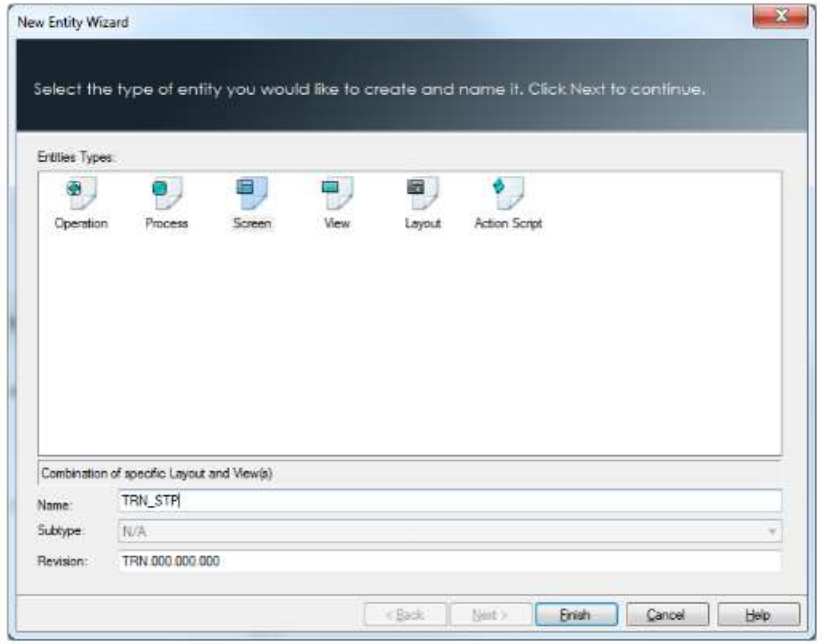
- Press the Save button



LAB 1: Create TRNXX_STP Screen

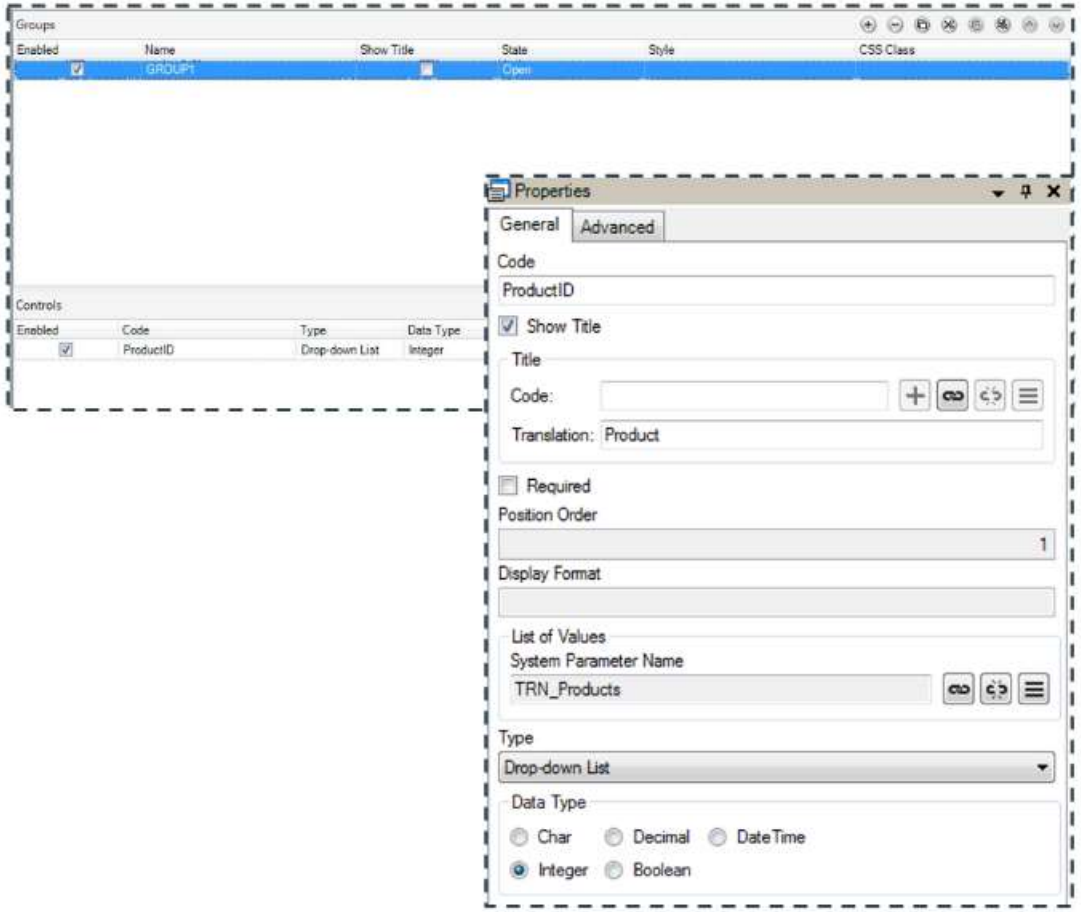
Now, you need to create a Screen, and then configure its View.

- Create Screen TRNXX_STP
- Base Screen: checked
- Layout: PortalOneWindow
- Header: none



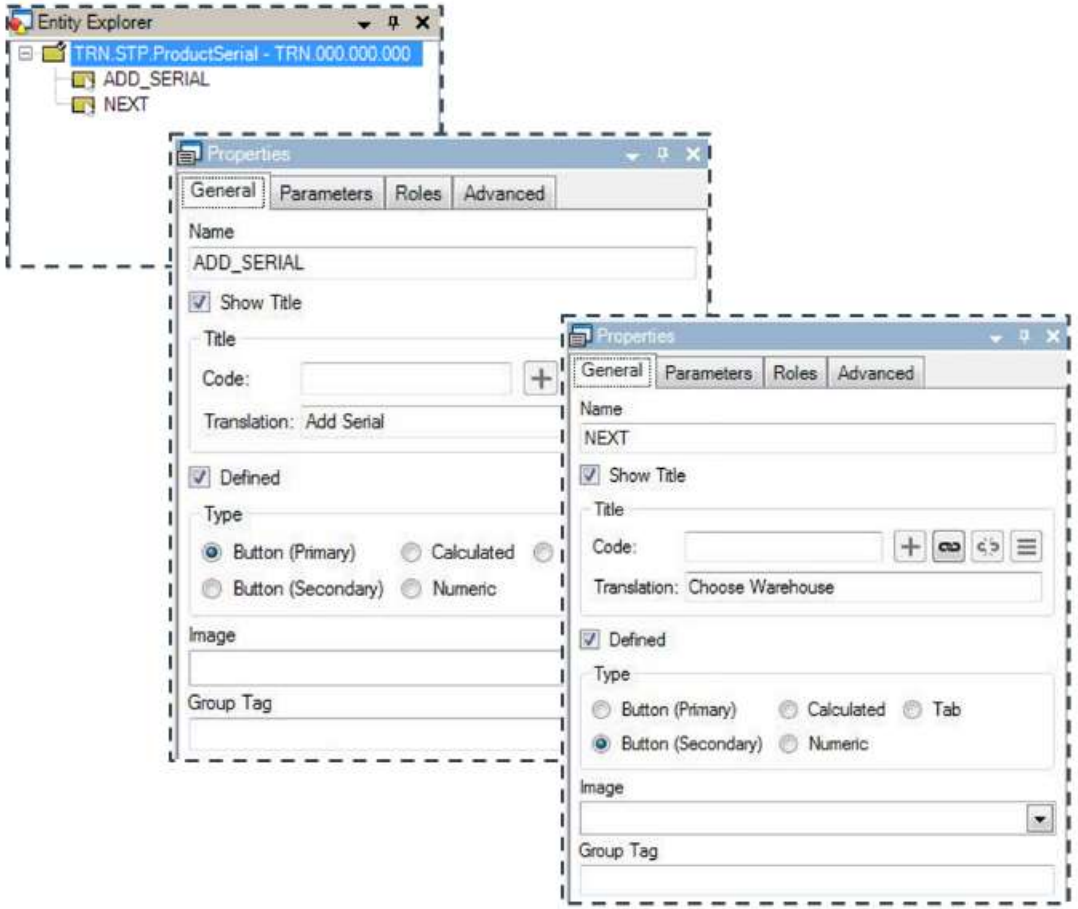
LAB 1: Copy and Link a View

- In the main panel, copy and link PortalForm View to TRN.STP.ProductSerial View
- Open the new View and delete GROUP2
- In GROUP1, uncheck the Show Title box
- Under CONTROLS, delete the check_box Control
- Under CONTROLS, change the text_ex Control to:
  - Code: ProductID
  - Title Translation: Product
  - List of Values: link the TRNXX_Products
  - Type: Drop-down List
  - Data Type: Integer



LAB 1: Change View Actions

- Go to the ACTIONS tab
- Change the BUTTON_1 Action to:
  - Name: ADD_SERIAL
  - Title Translation: Add Serial
  - Type: Button (Primary)
- Change the BUTTON_2 Action to:
  - Name: NEXT
  - Title Translation: Choose Warehouse
  - Type: Button (Secondary)
- Save the View and the Screen, and change to Prototype



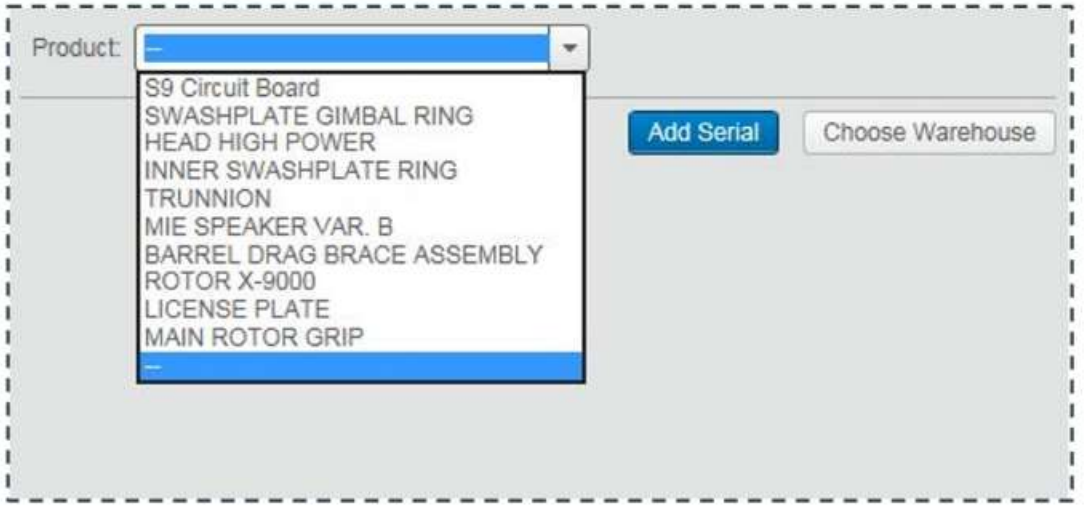
LAB 1: Test Run TRN_STP Screen

- Test run your Screen

The system parameter you created is linked to the drop-down Control, and allows you to select the product to create a serial number for. The Add Serial button sends the data and refreshes the Screen.

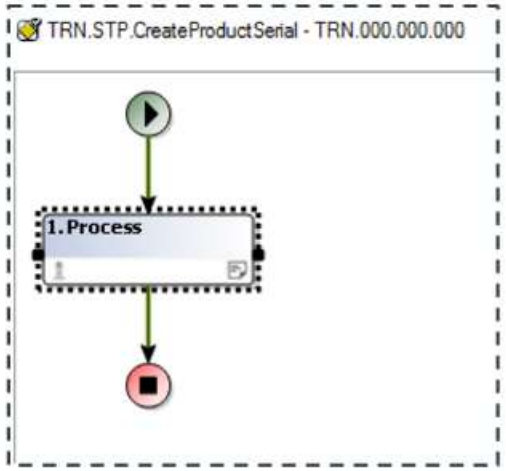
Since you will want to create a serial each time you select a product on this Screen, the next activity will be about attaching an On Action Operation to the ADD_SERIAL button.

We will come back to the NEXT button (Choose Warehouse) later.



LAB 1: Create a Standard Operation

- Create a Standard Operation TRNXX.STP.CreateProductSerial
- Make sure the Subtype is Action
- Add an Input to Output Function and call it GetExternalInputs
- Add 3 Pairs to this Function:
  - ProductID (integer, scalar)
  - ProductIDList (integer, list)
  - ProductSerialList (char, list)
- Create External Inputs for all



The ProductID will be taken from the product selected from the drop down. It will be used by further functions in this Operation Step to create a serial number and link the product to the serial.

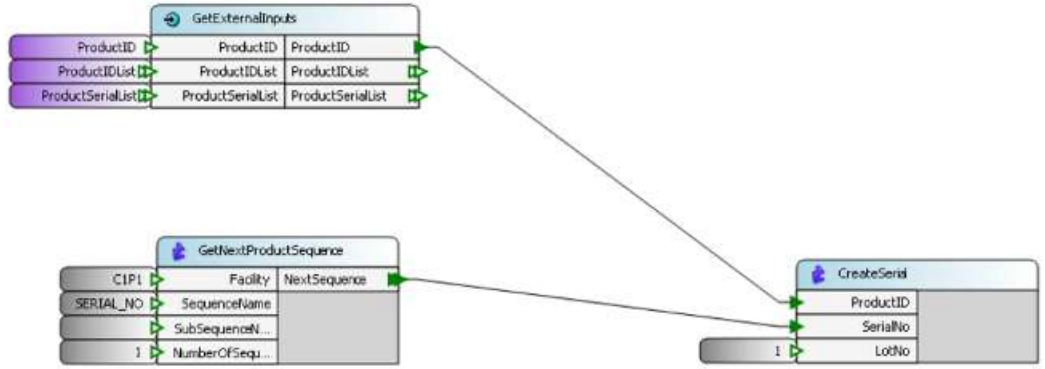
This Operation will be reused to create more serials, and this is why you are adding the ProductIDList and ProductSerialList. They are needed because every time this Operation is used, the new products and their serials will be added to the list, which will be used in the summary screen. Adding to the list will be done with a separate function, too.

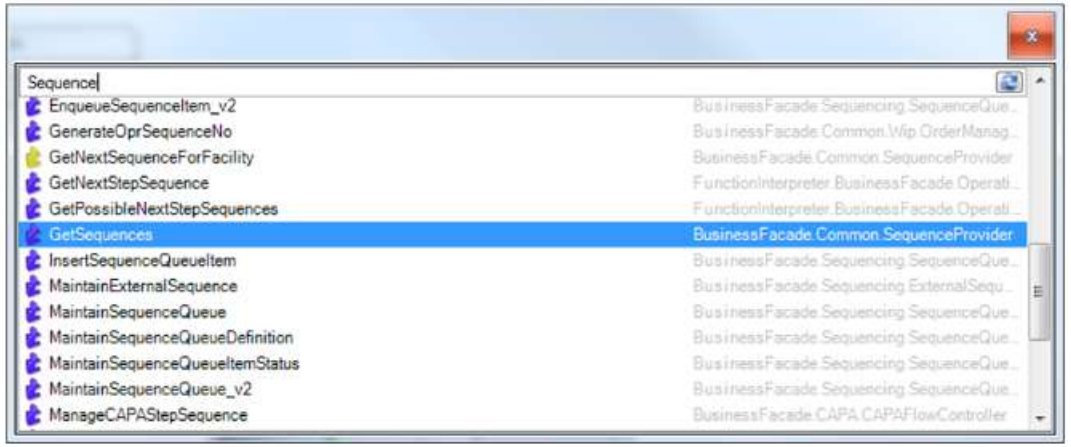
LAB 1: Add Business Component Functions

Now, add configuration to generate a unique serial number and assign it to the product piece that was just produced.

- Add a Business Component function
- Process Builder will show you a list of available Business Components. Select the GetSequences BC
- See how the name of the function changed when you chose the Business Component. Rename the function to GetNextProductSequence
- Populate inputs:
  - Facility: C1P1 (Constant)
  - SequenceName: SERIAL_NO (Constant)
  - NumberOfSequences: 1 (Constant)
- Add another Business Component function and link the CreateSerial Business Component.
- Provide the LotNo input as Constant: 1
- Connect the functions as shown in the picture

The next slides will explain the selected Business Components in more detail.

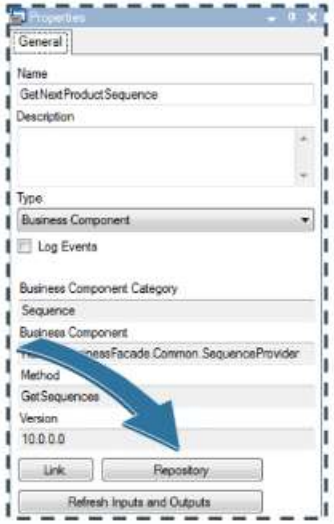




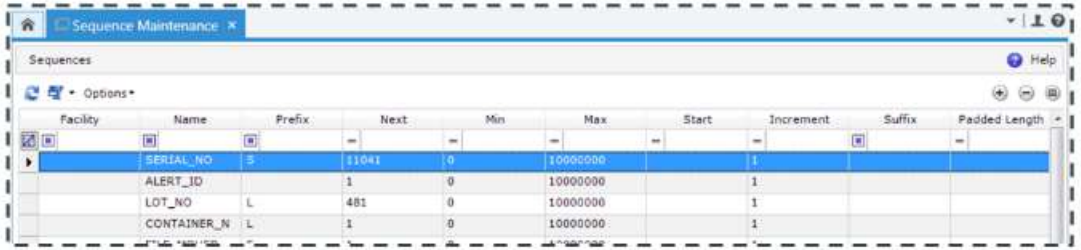
LAB 1: Use GetSequences Business Component

The GetSequences BC generates a unique alphanumeric string which can be used to assign a unique ID to an object, e.g. a container, or a product piece. The rules by which the sequence is returned are defined in the SEQUENCE_ table. You can find this table in the Database Documentation.

- When you are in the GetNextProductSequence function properties, click on the Repository button
- It opens the Business Components documentation. You can find there all available BCs, and there is an explanation for each what it does, how it works, and what are the affected database tables

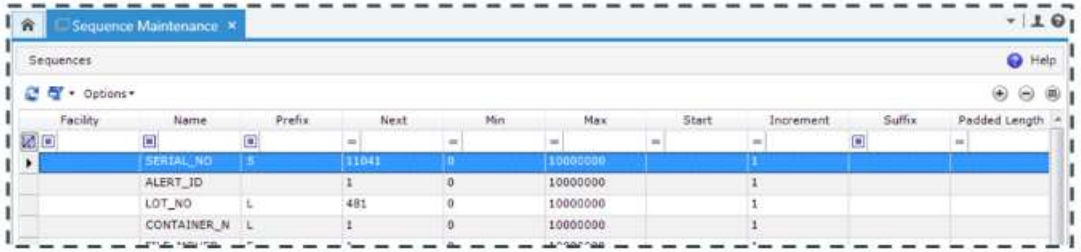


The Sequences are maintained through the Sequence Maintenance Screen in the DELMIA Apriso portal.



LAB 1: Sequence Maintenance

The Sequence you used on the GetNextProductSequence function is SERIAL_NO. You can find it in the Sequence Maintenance. Open the Sequence to see its details.

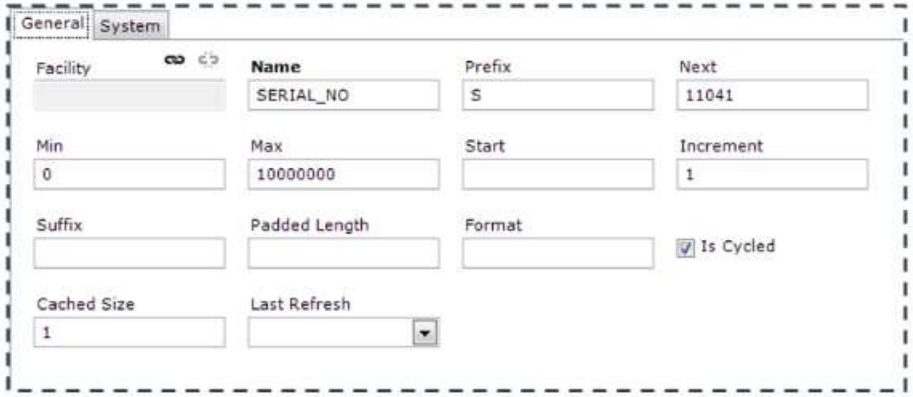


The SERIAL_NO Sequence will generate a string by these rules:

- Prefix S
- The actual sequence - what you can see in the Next field, e.g. 11041
- No Suffix

This and other parameters mean that the next time this sequence is called by the GetSequences BC, it will generate 1 sequence (Cached Size field), which will be S11041, then the value in the Next field will change to 11042 (Increment is 1). The Next field values will increase until 10000000 is reached (Max field). Then the sequence will restart (Is Cycled field) from 0 (Min field).

The Start field is used to provide continuity when the customer switches from difference sequence generating solution to DELMIA Apriso. This becomes the Next field value when the DELMIA Apriso starts controlling the sequence.



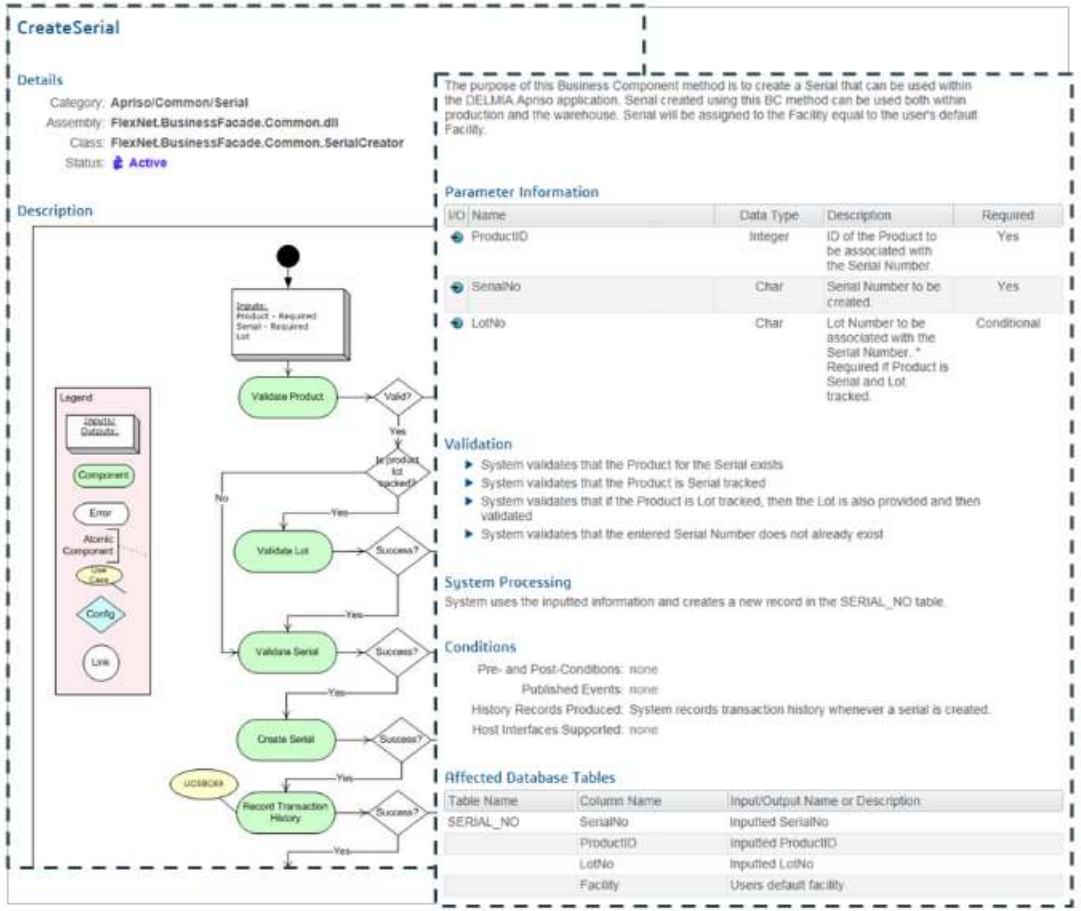
LAB 1: Add CreateSerial Business Component

The purpose of the CreateSerial Business Component is to create a Serial that can be used within the DELMIA Apriso application.

This BC requires the Product ID and the Sequence number, and combines them together in one record in the SERIAL_NO table.

Before the entry in the SERIAL_NO tables is made, the Business Component validates that:

- the Product for the Serial exists
- the Product is Serial tracked
- If the Product is Lot tracked, then the Lot is also provided and then validated
- the entered Serial Number does not already exist

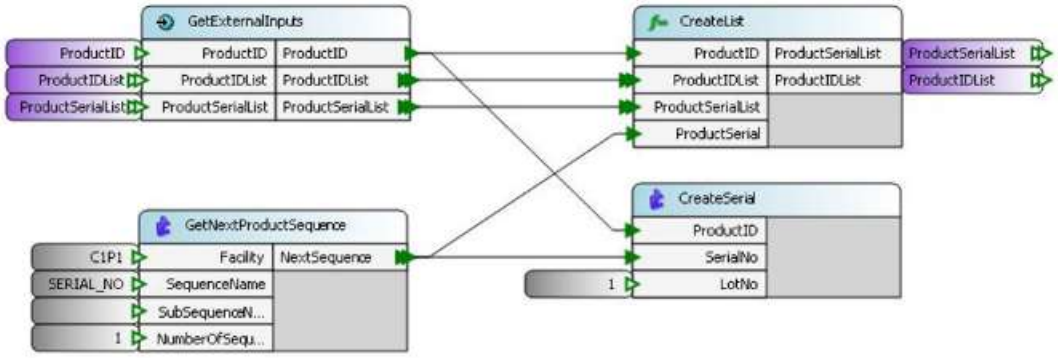


LAB 1: Modify CreateProductSerial Operation

So, in the TRNXX.STP.CreateProductSerial Operation, you are using the GetSequences BC to get the serial number which will be assigned to the Product ID you picked through the drop-down, and the CreateSerial BC will link this information together to create an entry in the SERIAL_NO table.

From then on, the particular instance of the product you picked is identifiable with its serial number.

The last function to be added is a User Formula type function, CreateList. It will expand the ProductSerialList and ProductIDList with the serial just created.

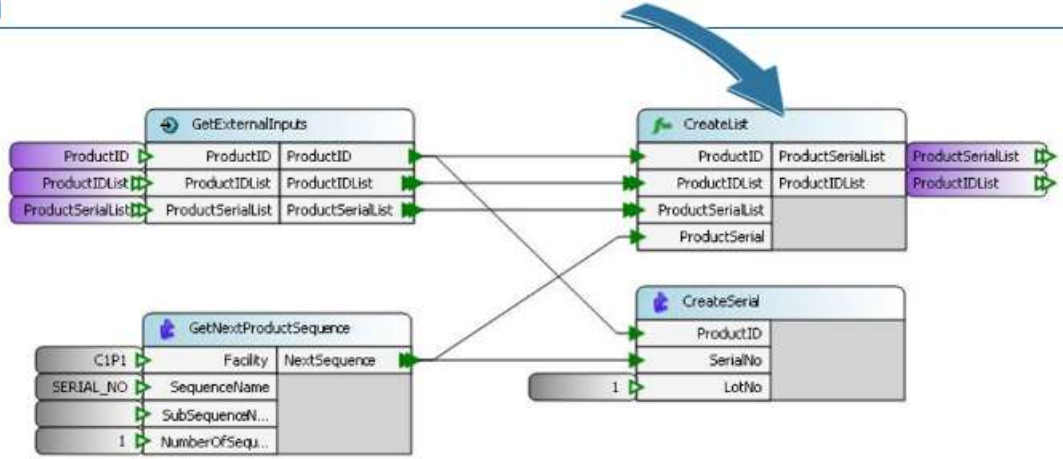


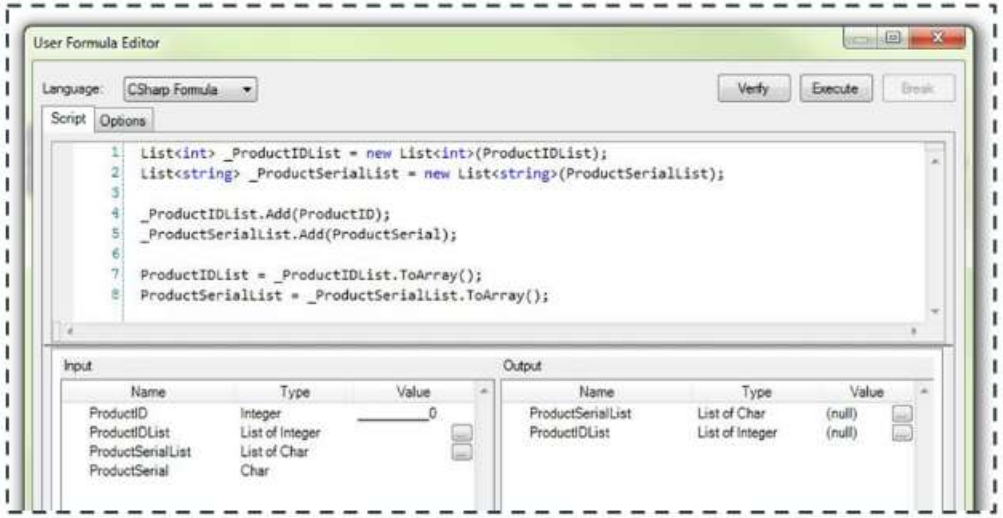
LAB 1: Add a User Formula Function

- Add a User Formula function named CreateList
- Create 4 inputs:
  - ProductID (Int)
  - ProductIDList (List, Int)
  - ProductSerialList (List, Char)
  - ProductSerial (Char)
- Link outputs from GetExternalInputs and GetNextProductSequence functions to inputs into the CreateList function to match the screenshot
- Add 2 outputs in CreateList:
  - ProductSerialList (List, Char)
  - ProductIDList (List, Int)
  - Add External Routings for both
- Add the C# script from the scripts file into the CreateList function:

SCRIPT FILE: Desktop/Training Materials/Level 1

- Close user Formula Editor
- Save the Operation and set to Prototype

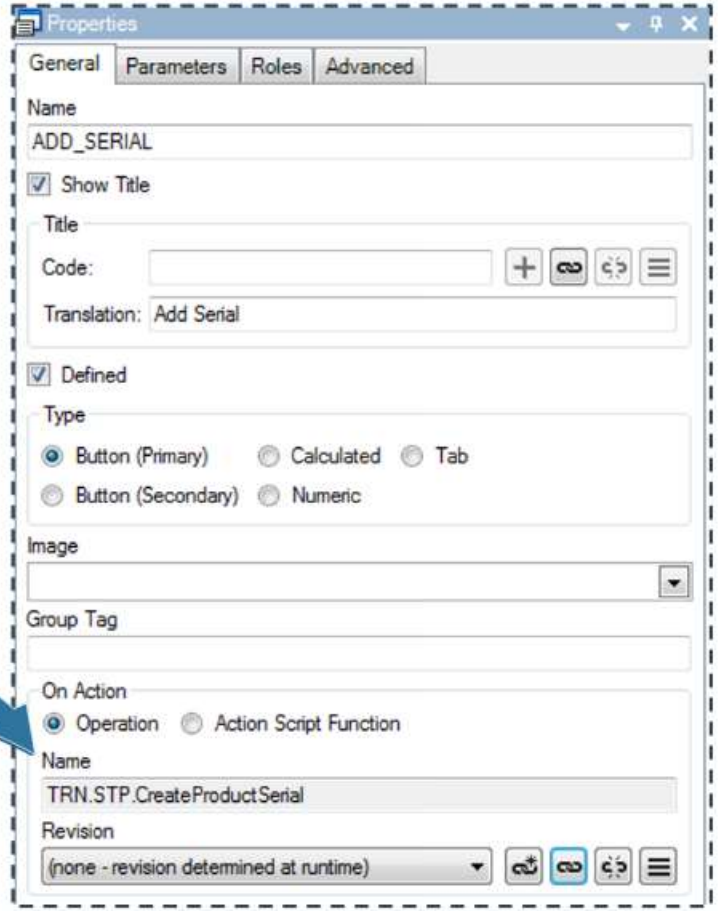




LAB 1: Connect the Operation to the Button

Now connect the Operation to the ADD_SERIAL button.

- Go back to the TRNXX.STP.ProductSerial View
- Go to the ADD_SERIAL button properties
- Use the Link button in the On Action Operation section
- In the window choose the Action Operation TRNXX.STP.CreateProductSerial
- Save the View



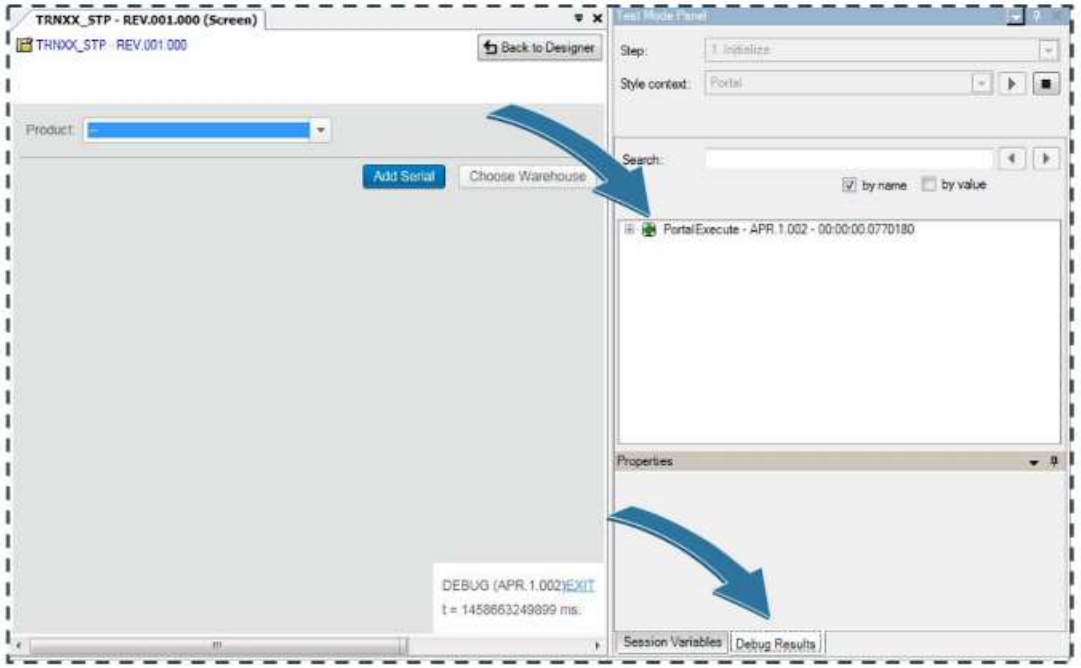
LAB 1: Test Run TRNXX_STP Screen

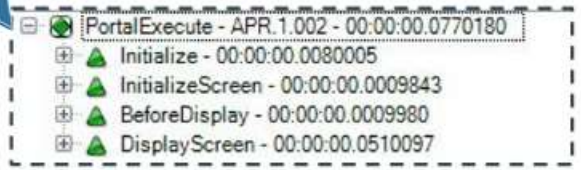
- Open your Screen in test mode. Don't click the Add Serial button just yet

You don't have any summary Screen yet to see what TRNXX_STP does in test run, but you can check these details in the Test Mode Panel on the right.

- Switch to the Debug Results tab
- Expand the PortalExecute Operation. The Portal Execute Operation is the Operation which makes all Screen Flow Management processing work. It is at the core of SFM, and you will never need to edit it
- You can see the main steps of this Operation, which you can further expand for more details

For now, you will be interested to see what happens to the ProductIDList and ProductSerialList variables.

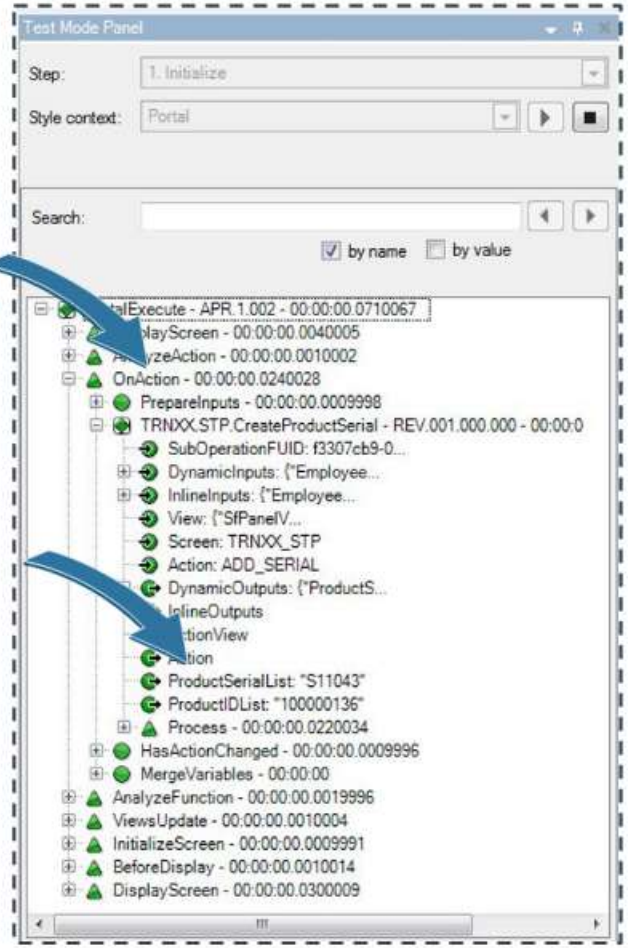




LAB 1: Test Mode Panel

- Choose a Product and click the Add Serial button
- Go to the Test Mode Panel, Debug Results, and check if the On Action operation was executed
- In the picture on the right you can see that new values were added to the Product and Serial lists

Get used to working with Debug Results, as this is a powerful tool to help you build high quality solutions in Process Builder.



LAB 2: Create a Screen to Assign Products to a Warehouse Location

Task:

- Build a Screen where the user will be able to select a warehouse location, in which the serialized products will be placed. The choice of locations will be displayed as radio buttons
- The Screen will require an On Action Operation linked to the Move Container button, which will do two things. First, it will create a new container. Second, it will move the serialized products into that container, and the container to the selected warehouse location. For this purpose you will use a new Business Component: Adjust_v92

What you will learn:

- Create a location for serialized product numbers
- Learn about Adjust_v92 Business Component

Training environment:

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

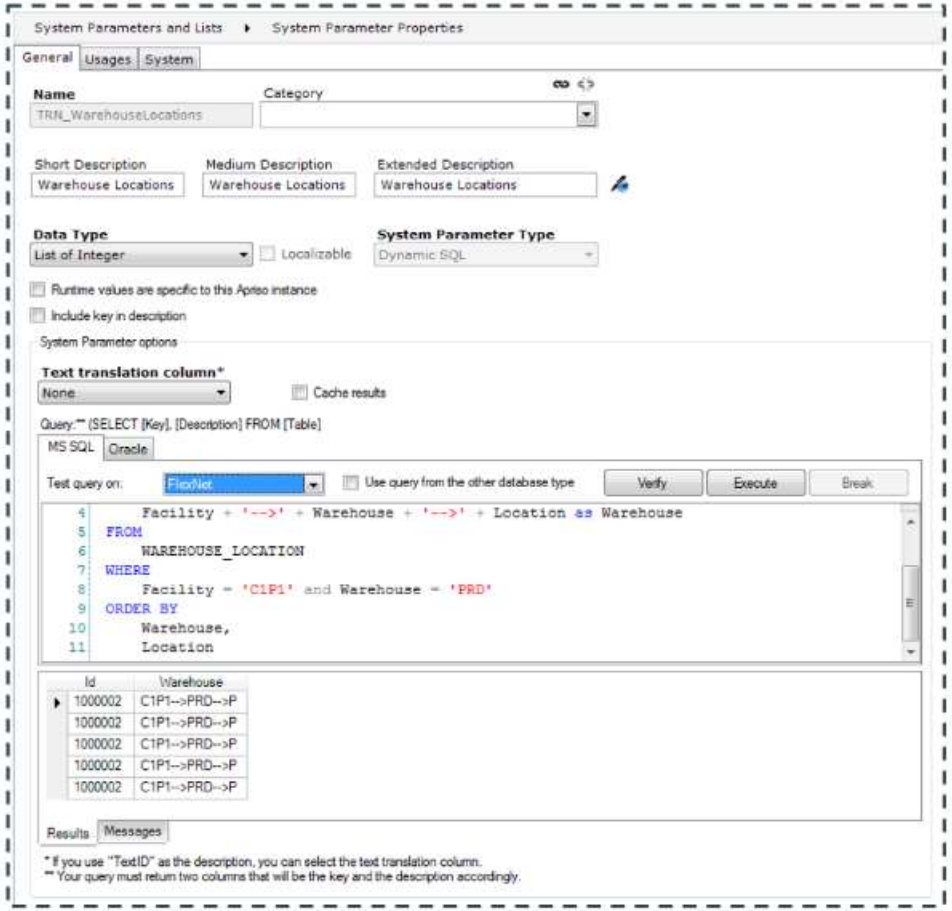
30 min

LAB 2: Create Warehouse Locations List

This list will populate labels for the radio buttons to select the warehouse location where the container with the serialized products will be stored.

- Go to the System Parameters and Lists Manager
- Create a new List. Set the properties as follows:
  - Name: TRNXX_WarehouseLocations
  - Short Description: Warehouse Locations
  - Data Type: List of Integer
  - System Parameter Type: Dynamic SQL
  - SQL Script - paste the query from the scripts:

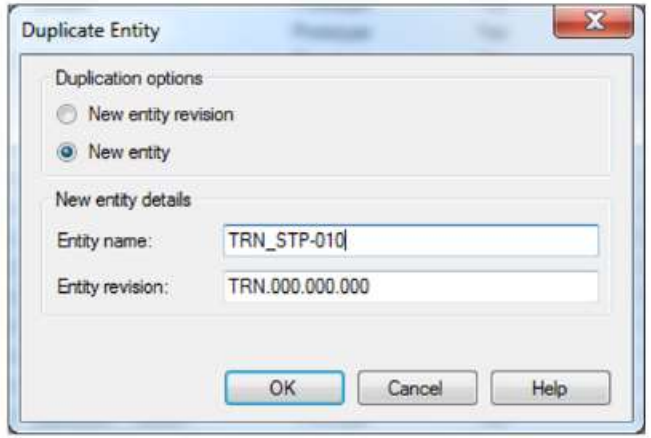
SCRIPT FILE: Desktop/Training Materials/Level 1



LAB 2: Create TRNXX_STP-010 Screen

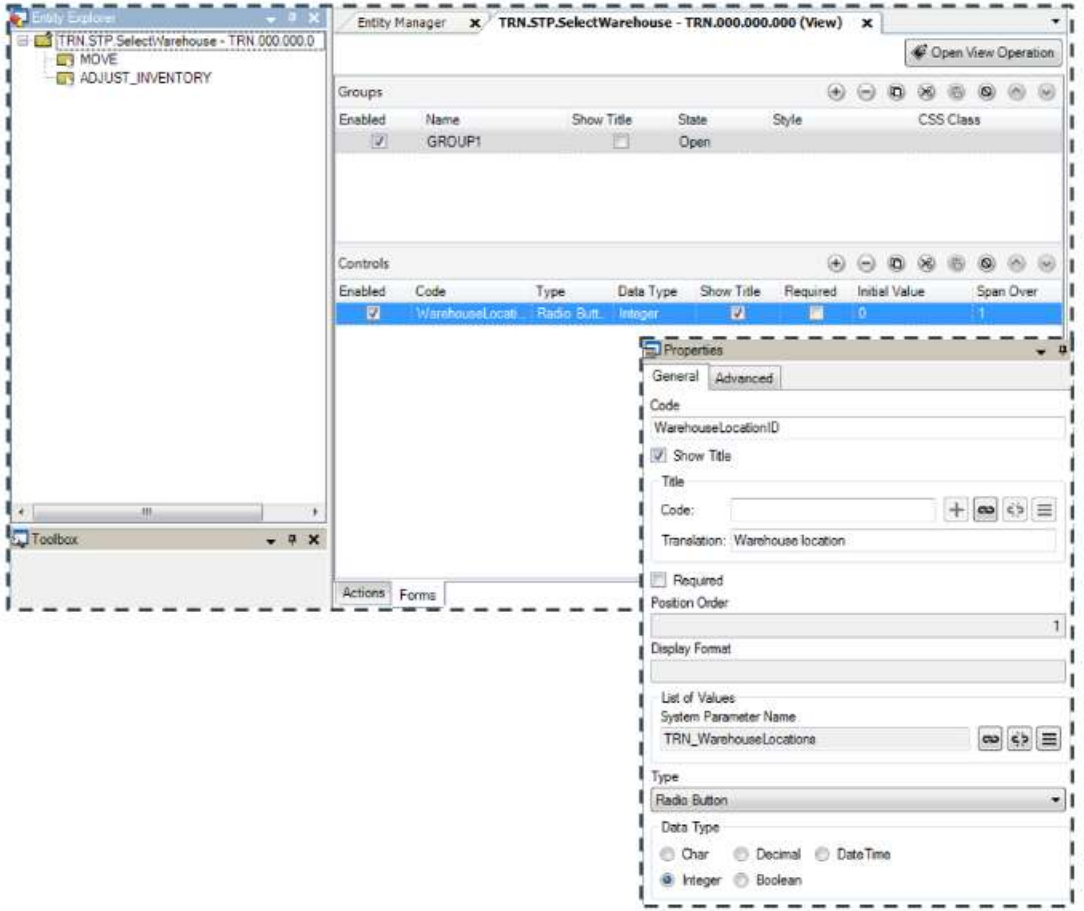
Create a Screen to show the choice of warehouse locations for the products.

- Right click on TRNXX_STP Screen and choose New Revision/Duplicate
- Name the new entity TRNXX_STP-010
- Set Entity revision
- Press OK button



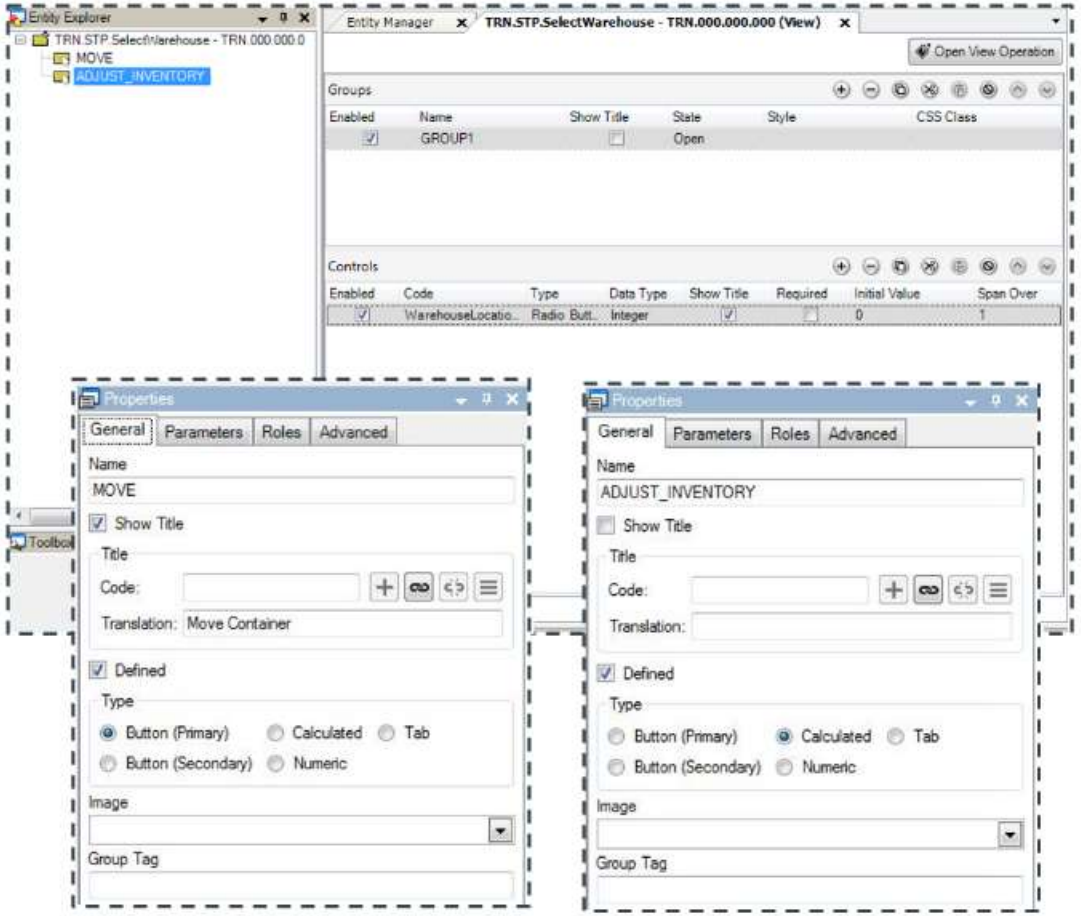
LAB 2: Copy and Link a View

- Go to Screen TRNXX_STP-010
- In sub-panel General Properties, in view section, click the Copy & Link button and choose PortalForm View
- In Duplicate Entity window set Entity name to: TRNXX.STP.SelectWarehouse
- Open TRNXX.STP.SelectWarehouse View
- Make sure the new View has one Group, with just one Control (Show Title unchecked):
  - Code: WarehouseLocationID
  - Title Translation: Warehouse location
  - Type: Radio Button
  - List of Values: link the TRNXX_WarehouseLocations List
  - Data Type: Integer



LAB 2: Change View Actions

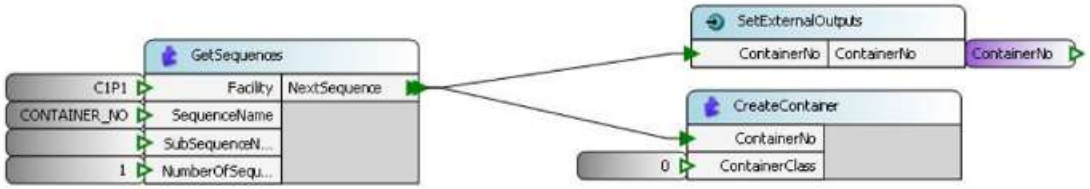
- Change BUTTON_1 Action:
  - Name: MOVE
  - Show Title: Checked
  - Title Translation: Move Container
  - Type: Button (Primary)
- Change BUTTON_2 Action:
  - Name: ADJUST_INVENTORY
  - Show Title: Unchecked
  - Type: Calculated



LAB 2: Create TRNXX.STP.CreateContainer Operation

- Create TRNXX.STP.CreateContainer Operation, Subtype Action. This will be used to create the new container
- Add a Business Component function. Link the GetSequences BC
- Provide values for inputs:
  - Facility: C1P1 (Constant)
  - SequenceName: CONTAINER_NO (Constant)
  - NumberOfSequences: 1 (Constant)
- Add another Business Component, CreateContainer
- Add a SetExternalOutputs function, with one input/output:
  - ContainerNo (Char)
  - Create an External Output for ContainerNo
- Connect the functions as shown in the picture
- Save the Operation and change status to Prototype

You are using the GetSequences BC again, but this time it is for the Sequence which provides unique identifiers for containers.



LAB 2: CreateContainer Business Component

The CreateContainer BC is used to create containers. Therefore the ContainerNo (which is the customer defined container identifier) is a mandatory input.

You may use a container class, if you need to configure more complex container structures.

The container class information is stored in the CONTAINER_CLASS table.

CreateContainer

Locate in Browser

Details

Category: **Apriso/Common/Container**
Assembly: **FlexNet.BusinessFacade.Common.dll**
Class: **FlexNet.BusinessFacade.Common.ContainerCreator**
Status: **Active**

Description

The purpose of this Business Component is to create a container.

Parameter Information

I/O	Name	Data Type	Description	Required
→	ContainerNo	Char	Container number of the container to be created.	Yes
→	ContainerClassID	Integer	Container class of the container to be created.	No

Validation

- System validates that the Container does not exists.
- System validates that the input Container Class exists.

System Processing

- System creates a Container record.

Conditions

Pre- and Post-Conditions: Container with same name should not exist in the system.
Published Events: none