blueprism®

Software Robots - the Virtual Workforce

Foundation Course

TRAINING GUIDE

Version: 5.0.1

For more information please contact:

info@blueprism.com | UK: +44 (0) 870 879 3000 | US: +1 888 7577476

www.blueprism.com

Contents

1. Introduction 6
1.1. Blue Prism's Robotic Automation 6
2. Process Studio 7
2.1. Running a Process 7
2.2. Basic Skills 7
2.3. Process Validation 10
2.4. Decision Stage 10
2.5. Calculation Stage 12
2.6. Data Items 13
2.7. Review 16
3. Process Flow 17
3.1. Decisions 17
3.2. Circular Paths 18
3.3. Controlling Play 19
3.4. Set Next Stage 19
3.5. Breakpoints 20
3.6. Collections and Loops 20
3.7. Layers of Logic 25
3.8. Pages for Organization 29
3.9. Review 31
4. Inputs and Outputs 32
4.1. Input Parameters 32
4.2. Stepping and Pages 35
4.3. Data Item Visibility 35
4.4. Data Types 37
4.5. Output Parameters 38
4.6. Start-up Parameters 41
4.7. Control Room 41
4.8. Process Outputs 46
4.9. Review 47
5. Business Objects 49
5.1. Object Studio 49
5.2. Business Objects 50

5.3. Action Stage 51
5.4. Inputs and Outputs 53
5.5. The Process Layer 53
5.6. Review 54
6. Object Studio 55
6.1. Creating a Business Object 55
6.2. Application Modeler 55
6.3. Spying Elements 57
6.4. Attributes 58
6.5. Attribute Selection 59
6.6. Launch 60
6.7. Wait 63
6.8. Timeouts 67
6.9. Terminate 69
6.10. Write 70
6.11. Press 74
6.12. Attach and Detach 75
6.13. Read 79
6.14. Actions 81
6.15. Action Inputs and Outputs 83
6.16. Data Items as Inputs 85
6.17. Review 86
7. Overview of Error and Case Management 87
8. Error Management 88
8.1. Exception Handling 88
8.2. Recover and Resume 88
8.3. Throwing Exceptions 90
8.4. Preserving the Current Exception 95
8.5. Exception Bubbling 97
8.6. Exception Blocks 98
8.7. Exception Handling in Practice 100
8.8. Review 103
9. Case Management 104
9.1. Review 104
9.2. Queue Items 107

9.3. Work Queue Configuration 110
9.4. Defer 113
9.5. Attempts 114
9.6. Pause and Resume 116
9.7. Filters 116
9.8. Reports 116
9.9. Review 116
10. Additional Features 118
10.1. Safe Stop 118
10.2. Collection Actions 119
10.3. Choice Stage 119
10.4. Logging 120
10.5. Log Viewer 121
10.6. System Manager 122
10.7. Process/Business Object Grouping 122
10.8. Process and Object References 123
10.9. Export and Import 124
10.10. Release Manager – Packages and Releases 124
11. Consolidation Exercise 127
11.1. Order System Process 127
11.2. Consolidation Exercise Checklist 133
11.3. Submitting Your Completed Solution 133
12. Advanced Features 134
12.1. Undefined Collections 134
12.2. Data Item Initialization 134
12.3. Data Item Exposure 135
12.4. Casting 138
12.5. Code Stage 139
12.6. Run Mode 141
12.7. Initialize and Cleanup 143
12.8. Attribute Match Types 143
12.9. Dynamic Attributes 144
12.10. Active Accessibility 146
12.11. Application Manager Mode 146
12.12. Global Clicks and Keys 147

12.13. Credentials 147
12.14. Environment Locking 148
12.15 Command Line 150
12.16 Resource PC 151
13. Further Application Types 152
13.1. Mainframe Applications 152
13.2. Java Applications 155
13.3. Match Index and Match Reverse 156
13.4. Surface Automation 157
14. Conclusion 158

Blue Prism Limited, Centrix House, Crow Lane East, Newton-le-Willows, WA12 9UY, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

1. Introduction

1.1. Blue Prism’s Robotic Automation

Robotic Automation refers to process automations where computer software drives existing enterprise application software in the same way that a user does. This means that unlike traditional application software, Robotic Automation is a tool or platform that operates and orchestrates other application software through the existing application's user interface and in this sense is not "integrated".

Blue Prism's Robotic Automation software enables business operations to be agile and cost-effective through rapid automation of manual, rules-based, back office administrative processes, reducing cost and improving accuracy by creating a "virtual workforce".

The virtual workforce is built by the operational teams or accredited Blue Prism partners using our robotic automation technology to rapidly build and deploy automations through leveraging the presentation layer of existing enterprise applications. The automations are configured and managed within an IT-governed framework and operating model which has been iteratively developed through numerous large scale and complex deployments.

2. Process Studio

A Blue Prism Process is created as a ***diagram*** that looks much like a common business flow diagram.

Processes are created in an area of Blue Prism named Process Studio which, as we will see, looks similar to other process modeling applications (such MS Visio) and uses standard flow diagram symbols and notation.

The key difference with a Blue Prism diagram is that it is not an inert two-dimensional representation of a Process. Rather, it is the graphical representation of a working computer program, one that will interact with applications, manipulate data, and perform decisions and calculations.

2.1. Running a Process

A Blue Prism Process is not a picture but a program in graphical form. We can run it in Process Studio and watch it working. One of the Process Studio toolbars provides commands to run the Process, similar to the buttons found on a DVD player.

Figure 1: Go and Reset Toolbar Buttons

Exercise 2.1.1 Opening and Running a Process

- Log into Blue Prism.
- Select the menu File>Open in the main window.
- Follow the wizard to open the Process named **Example Process**.
- Once you are in Process Studio, find the Go button,  , and press it to run the Process.
- See how the flow of the diagram is highlighted in orange.
- Press “Reset”() and then press “Go” again.
- Close Process Studio.

Key Point

- The Reset () button must always be pressed before re-running a Process.

2.2. Basic Skills

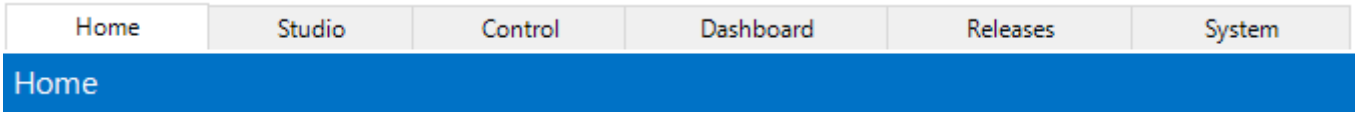
Process diagrams are comprised of various ***stages*** connected together using ***links*** to form logical structures.

Stages are selected from the toolbar and placed on the diagram using the mouse. They can be selected, moved, resized, formatted, cut, copied, pasted, and deleted, much as you might expect. Stages also have various properties that can be accessed by double clicking.

Links are created using the link tool by dragging the mouse from the center of one stage to the center of another. Dragging without the link tool can be used to select more than one stage at a time.

Exercise 2.2.1 Creating a New Process

- From the main Blue Prism window, select “Studio” from tabbed menu at the top of the screen.



Or select the Studio icon from the left-hand navigation menu

- Right click the Processes heading under studio and then select “Create a Process” from the context menu.

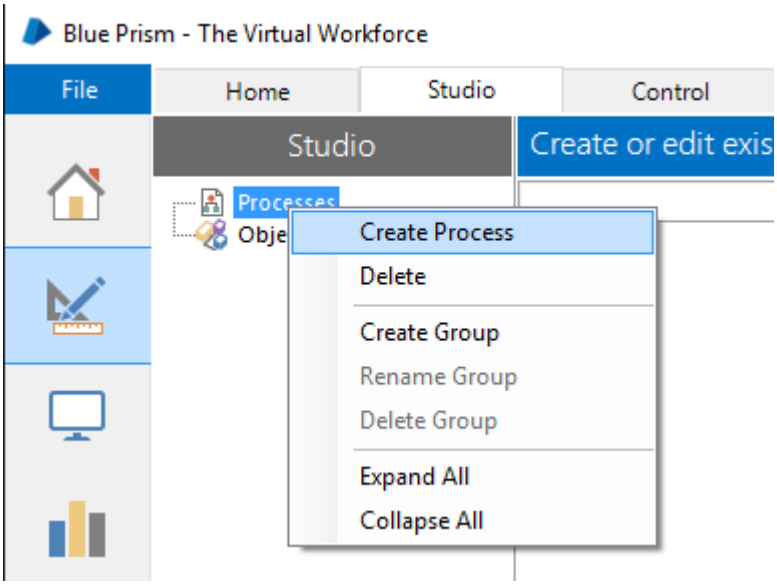


Figure 2: Create Process

- Name the new Process “First Process”.
- Enter a brief description of the Process.

  * *Tip: It’s good practice to add a brief description of what the Process is – to assist when reviewing, maintaining, or enhancing a Process.*

- The new process will appear beneath the Processes heading. Now double-click it to open it.
- Along the left-hand side of the screen is the Stages toolbar. To put a stage on the page click and drag it into position. Try this by dragging a few different stages onto the page.
- You will notice that one of the stages in the stages toolbar has a blue rectangle highlighting it. Whichever stage is highlighted will be the one added if you click directly on the page. Click on the calculation stage on the toolbar to highlight it.

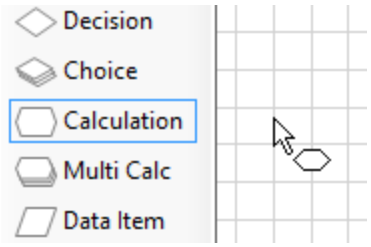


Figure 3: Process Studio Stages Toolbar

- Notice how the cursor changes and has the calculation stage icon next to it. Now click on the process page to add a calculation stage. This saves you having to return to the toolbar if you have several stages of the same type to add.
- Even with the calculation stage cursor you can still drag and place other stage types.
- When stages are on the page you can double click on various stages to view their properties.
- Press the Save button  but keep the diagram open.

    * *Tip: The first time a newly created Process is saved, no additional information is asked for.*

Space in Process Studio is effectively infinite, and pan and zoom tools can be used to maneuver around the diagram. The grid lines and "snap" settings are on by default but these can be switched off (via the View menu) if necessary.

Key Points

- Stages are added by clicking onto the page for the current cursor or by dragging from the toolbar.
- Links are created by dragging from "centre to centre", not from "edge to edge".
- Most stages have a single outbound link but some can have more.
- Most stages accept multiple inbound links but some (like the Start stage) cannot be linked to at all.
- Most stages appear on the diagram by a single shape but some have two parts.

Exercise 2.2.2 Saving a Process

Like any computer document, it is prudent to save a Blue Prism diagram every so often while you work on it. Keeping with the same diagram, in this exercise we'll make a modification and then save the changes.

- Drag and select ALL stages.
- Delete them by pressing the Delete key or by using the mouse menu (i.e. right-click).

    * *Tip: You may need to zoom out to select everything.*

- Notice that two stages have not been deleted - this is intentional. The "Start" and "Information" stages are permanent and cannot be deleted.
- Add a new End stage.
- Link the Start stage to the End stage.
- Press "Save". By default, each time a process is saved Blue Prism asks for a summary of the changes made.

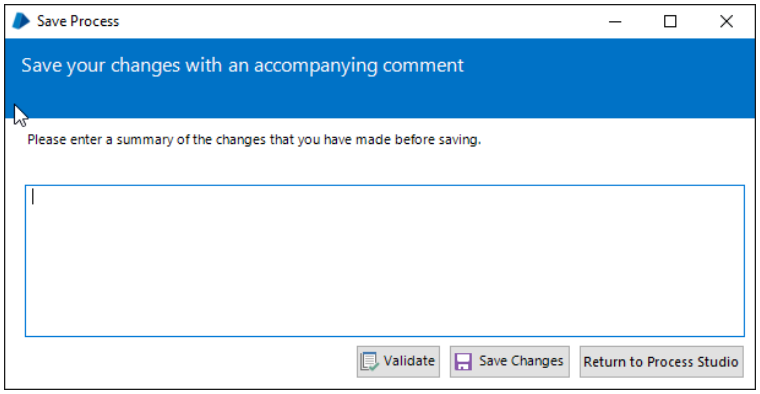


Figure 4: Save Process Window

Tip: It's good practice to add a brief description when you save a Process. Blue Prism uses your summary when it stores a history of Process changes.

Running the Process

- Press "Go", ,to run the Process. See how the flow of the diagram is highlighted in orange.
- Press "Reset" and "Go" again.

2.3. Process Validation

Basic configuration errors can be identified by clicking the Validation button on the toolbar. This will run a scan of the current configuration and a list of basic errors and warnings will be displayed.

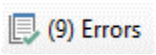


Figure 5: Process Validation Button

This functionality can be very useful for checking for basic "schoolboy" errors like typing mistakes or missing links, though it should not be relied on too heavily. If no errors are displayed then it should not be assumed that your Process is error free – the validation does not analyze the logic of a diagram and the onus is on the user to avoid problems like falling into an infinite loop.

Best Practice

- Get into the habit of using the validation tool as you go along, and in particular, run the validation before you close the Process.
- If you don't use the validation tool you will almost certainly have a glut of silly problems to fix during testing.

2.4. Decision Stage

A Process will rarely follow a single path along a straight line and we are likely to want to give our Process the option to make decisions and to travel one way or another.

Figure 6: The Decision Stage has two Outbound Links

The Decision stage has two outbound links. If the result of a Decision is true, it takes one direction. If the result is false, it takes the other.

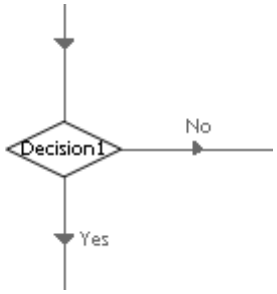


Figure 7: Decision Stage

The Decision stage is the simplest method to create multiple paths in a Process. The Decision stage works by evaluating the result of an *expression* as either *true* or *false*. Expressions are formulas used to calculate a value of some kind and are used in many other types of stages.

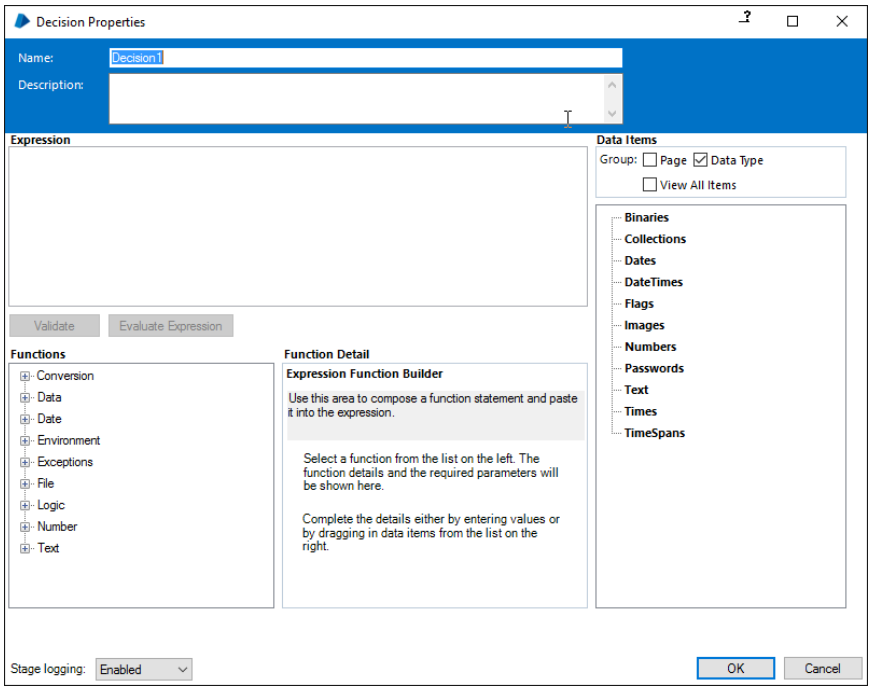


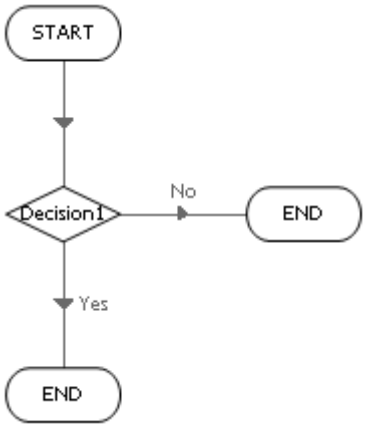
Figure 8: Decision Properties

Exercise 2.4.1 Adding a Decision Stage

Add a Decision stage to your *First Process* diagram by following these steps:

- Add a Decision stage and open its properties window with a double click.
- Type the expression 1>2 into the main Expression text area and press “OK”.
- Add a second End stage to the Process.
- Link the Start stage to the Decision stage.
- Link the Decision stage to both End stages.

Notice how one link is marked “Yes” and the other “No”. The resulting Process should look similar to the diagram below:



- Press “Reset” and “Go” to run the Process. Note which route the Process follows and which End stage it finishes at.
- Enter the Decision stage properties and change the expression to 1<2.
- Press “Reset” and “Go” again. Notice how the Process changes direction according to the expression.
- Right-click on the Decision stage and choose “Switch” from the mouse menu.
- Re-open the Decision stage properties to see that the expression has not changed (still True) - only the links that have been swapped over.

Key Point

- A page can have multiple end points but only one starting point.

2.5. Calculation Stage

Similar to a Decision stage, a Calculation stage uses an expression in its properties window but with one key difference: A Calculation stage needs to retain the result from its expression, whereas a Decision stage only uses its result (True or False) to choose one path or another. A Calculation stage aims to calculate a value and store it somewhere.

Figure 9: Calculation Toolbar Button

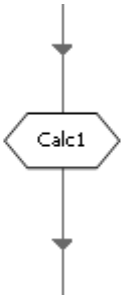


Figure 10: Calculation Stage

The Calculation stage contains a “Store Result In” field in the middle of the properties window, as shown below:

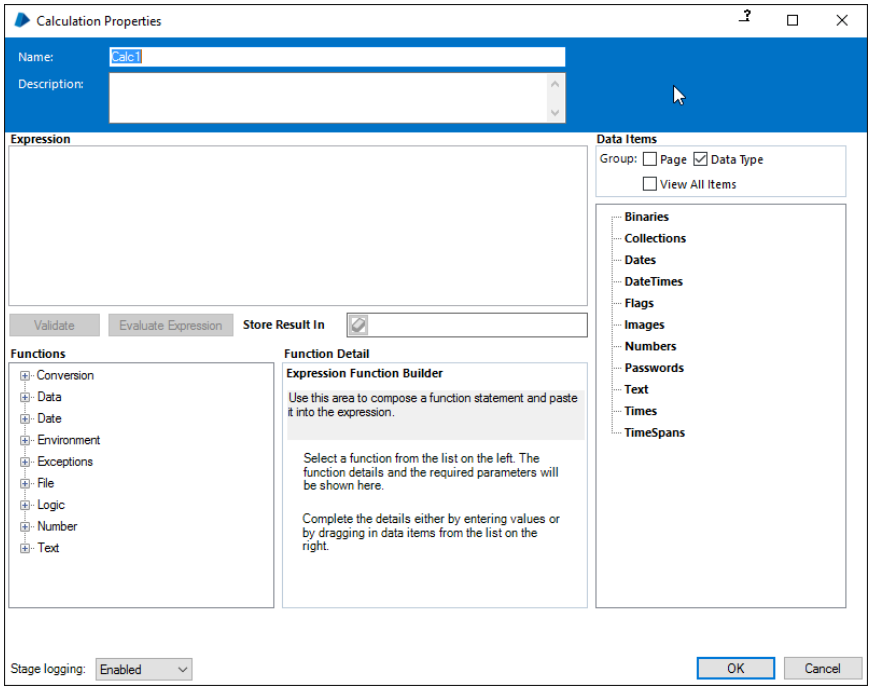


Figure 11: Calculation Properties

Results from a Calculation stage are stored in stages known as “Data Items” (see below). The related Data Item for a Calculation stage is noted in the “Store Results In” field in the Calculation stage properties.

2.6. Data Items

These parallelogram-shaped stages act as place holders for values such as numbers, text, and dates. They can be given meaningful names like “Account Number” and employed around the diagram. A programming comparison can be made by likening a Data Item to a *variable*.

Figure 12: Data Item Toolbar Button

Figure 13: Data Item Stage

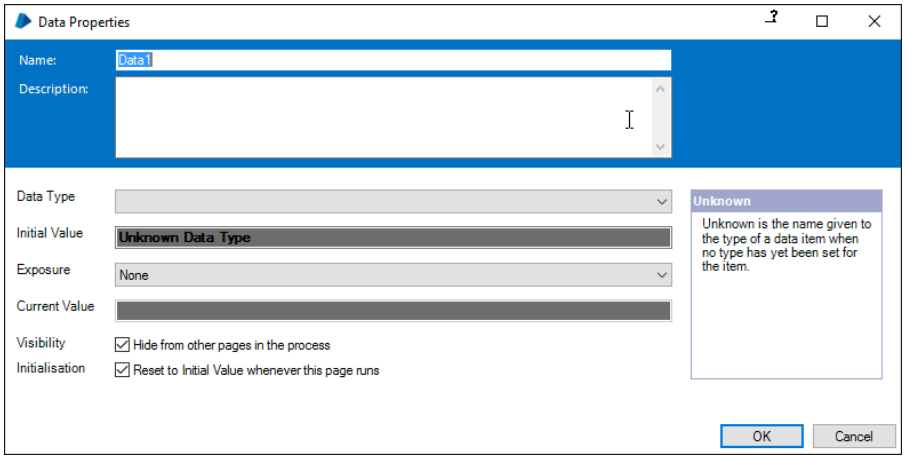


Figure 14: Data Item Properties

Key Point

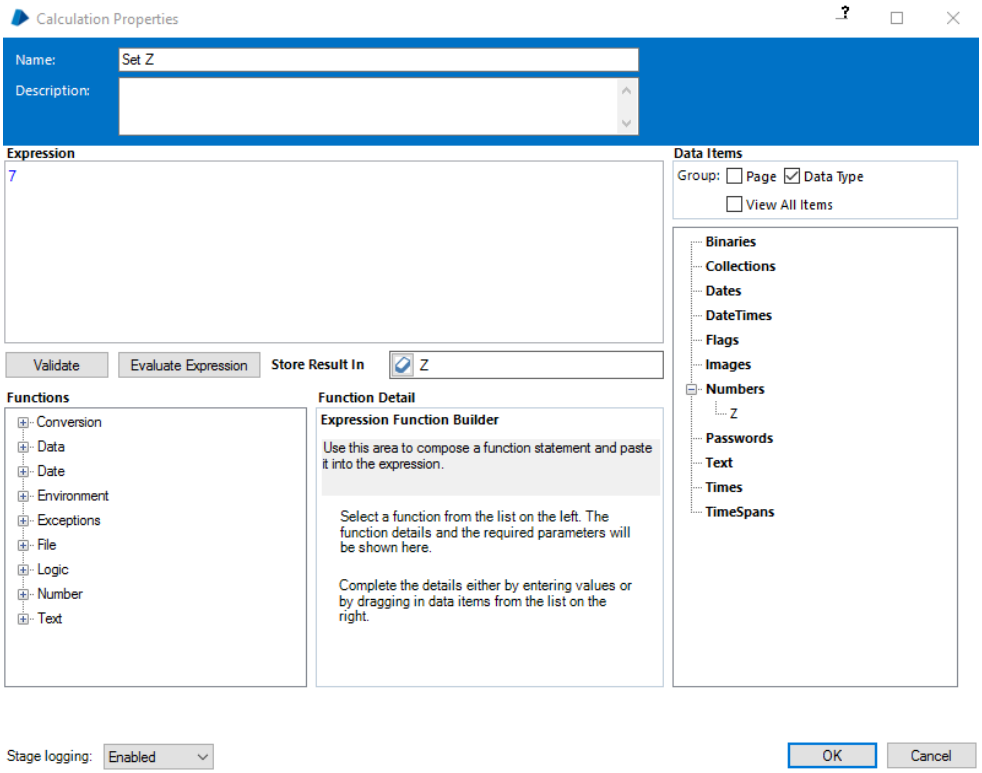
- Data Items are not linked into the logical flow, they "float" and cannot be physically connected to any other part of the diagram.

Exercise 2.6.1 Calculations

Continue working on your ***First Process*** diagram by following these steps:

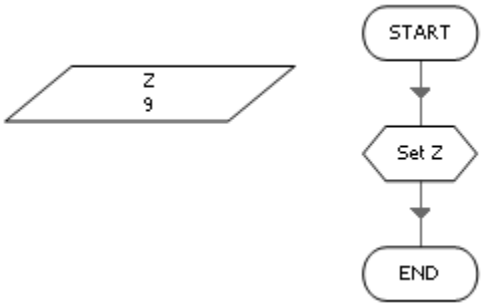
- Delete the Decision stage and one of the End stages. (Links are automatically deleted when you delete a stage.)
- Add a Data Item named **Z**, with a data type of **number** and an initial value of **9**.
- Add a Calculation stage, name it **Set Z** and open its properties.
- Find Z in the right-hand tree view list of Data Items. These are grouped by data type, so Z will be in the Numbers group.
- Drag Z into the **Store Result In** field in the Calculation stage properties window.
    - *Tip: You can also type the Data Item name into the "Store Result In" field. If a typed in Data Item name does NOT match an existing Data Item, a new Data Item will be created with the typed in name.*
- Enter the number **7** into the expression text area.

The Calculation stage is now set up to store the result of its expression (i.e., 7) in Data Item Z, and should look something like this:



Exercise 2.6.2 Calculation to Set Z

- "OK" the Calculation properties and link from the Start to the Calculation stage.
- Link the Calculation stage to the End stage.
- Press "Reset"
- The Process should look similar to the diagram below:

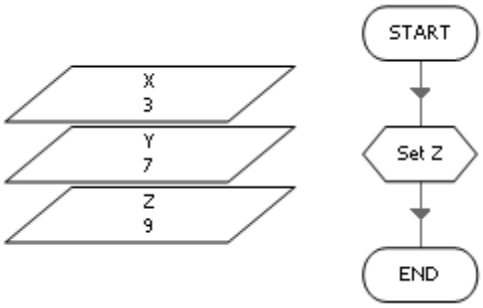


- Press "Go".
- Notice the value of Z change from 9 to 7.
- Open the properties of Z to see that the Initial Value and Current Value fields are different.

Exercise 2.6.3 Data Items in Expressions

A Data Item can be included in an expression by enclosing its name in square brackets. This is similar in concept to MS Excel expressions that use cell references, e.g., A3*1.175. Continue working on your diagram:

- Add two new **number** Data Items named **X** and **Y** with initial values of **3** and **7**.



- Edit the Calculation stage properties and change the expression to [X]+[Y].
- Press "Reset" and "Go".
- Open the properties of Z to see that Current Value is now 10.

  * *Tip: It can help to think of "Store Result In" as meaning "Z becomes 10" rather than "Z equals 10".*

Key Point

- Square brackets must be used to include a Data Item in an expression, e.g., [Account ID].

Exercise 2.6.4 Expressions Using the "Store Result In" Data Item

The Data Item used in the "Store Result In" field can also play a part in the expression. Continue working on your diagram as follows:

- Change the Calculation expression to ([X]*[Y])+[Z]
- Press "Reset" and "Go"
- Open the properties of Z to see that Current Value is now 30

  * *Tip: The standard logic for using brackets applies to Blue Prism expressions.*

2.7. Review

- Creating a Process is like building a flow diagram
- Toolbars are used to place, move, resize, copy, delete, and link stages
- The Go button starts the Process
- The Reset button readies the Process to run again – resetting Data Items to initial values
- Flow direction is changed using a Decision stage to work out an expression that is true or false
- Data Items are used as place holders for values
- Calculation stages use expressions to calculate new values for Data Items
- Data Items can have an initial value and a current value

Blue Prism Keywords

Stage, Start, End, Link, Decision, Expression, Calculation, Store In, Drag and Drop, Data Item, Data Type, Initial Value, Current Value, Go, Reset.

3. Process Flow

We have seen how Decision stages are used to create alternate paths in a Process. If you have any programming knowledge, you may like to relate a Decision stage as to an ***IF THEN ELSE*** statement. The expressions we have looked at have been very simple, but they can become complex.

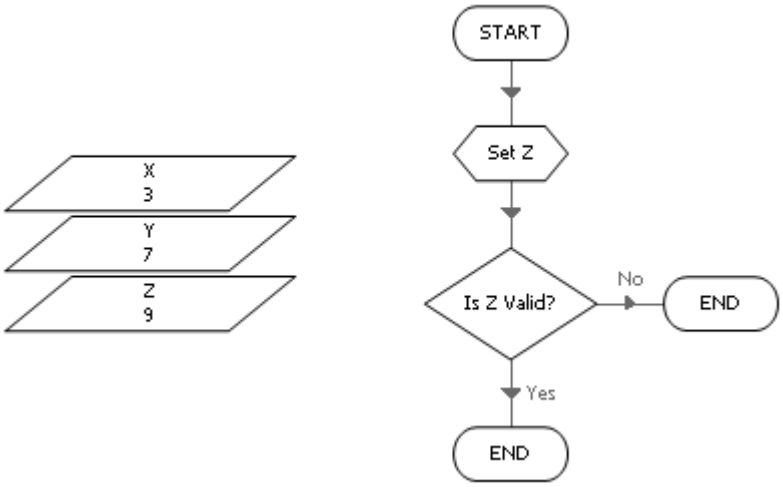
3.1. Decisions

Exercise 3.1.1 Validation Decision

A Decision can be used to check the value of a Data Item in order to decide which way a Process should go. Return to the ***First Process*** diagram from the previous exercises and follow these steps:

- Drag the End stage down to make a gap under the Calculation stage.
- Add a Decision after the Calculation and name it **Is Z Valid?**.
- Set the expression in the Decision to check if Z is negative or positive.
    - *Tip: A negative number is less than zero.*
- Link the Calculation stage to the new Decision stage.
    - *Tip: Rather than deleting the existing link connecting the Calculation to the End, just draw a new link to the Decision and the old one will be removed automatically.*
- Add a second End stage and link the Decision stage to the two End stages (it doesn't matter which End stages the Yes and No branches of the Decision link to).

The Process should now look similar to this:



- Press "Reset" and "Go". Notice which path the Process follows and which End stage it stops at.
- Change the Calculation expression to **[Z]-([X]*[Y])**.
- Press "Reset" and "Go". The Process should now stop at the other End stage.
- Save and Close the **First Process** diagram.

3.2. Circular Paths

The basic principle of Blue Prism is to automate repetitive work, and as such Process will probably need to repeat some steps over and over again. So far we have only looked at simplistic, ***linear*** paths but the reality is that we will want to create diagrams that follow some sort of ***circular*** path.

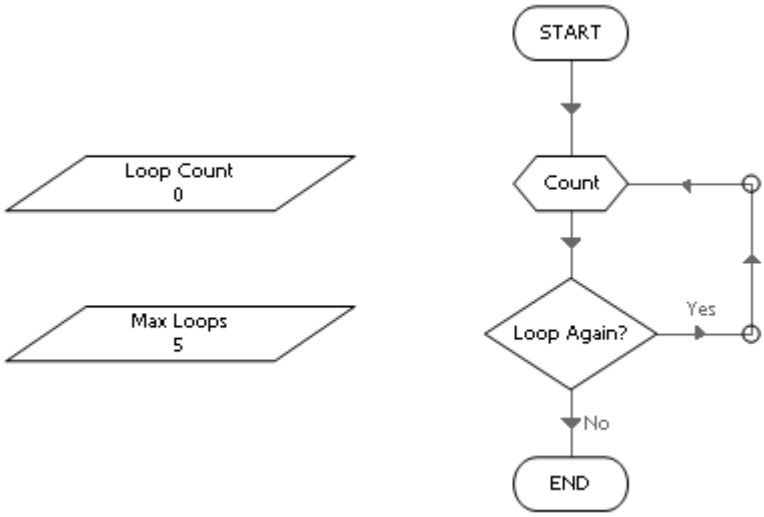
Although most stages have only one outbound link, there is no limit to the number of inbound links a stage can have. This means that a stage can be approached from more than one direction.

Exercise 3.2.1 Circular Paths

A common example of circular paths is to have a looping diagram with a Decision controlling the number of times the logic will flow around the circuit.

- Create a new Process named **Circular Path Exercises**.
- Add a Data Item named **Loop Count** with data type **number** and initial value **0**.
- Add another Data Item named **Max Loops** with data type **number** and initial value **5**.
- Add a Calculation that will increase the value of **Loop Count** by **1**.
    - *Tip: Remember the "Store In" Data Item can also play a part in the expression.*
- Add a Decision that compares **Loop Count** with **Max Loops**.
    - *Tip: Use any of the standard notation for "equals" (=), "less than" (<), or "greater than" (>) to compare values.*
- Link the Decision back up to the Calculation and down to the End.
    - *Tip: Use Anchor stages* ○ *to link around corners.*

Your diagram should end up looking something like this:



3.3. Controlling Play

You may have noticed when pressing the Go button that the default "running speed" is fairly slow. This can be changed using the drop-down sliding adjuster, just to the right of the Go button. Increasing the speed can be useful when working through multiple iterations.

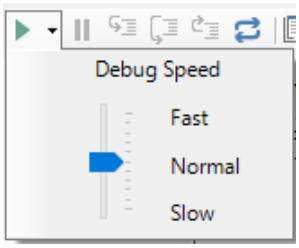


Figure 15: Debug Speed Adjuster

A running Process can also be paused and resumed using the buttons on the toolbar. And as an alternative to continuous play, a Process can be run one stage at a time. This is known as ***stepping***.

To step though the process one stage at a time click the step button 

Exercise 3.3.1 Controlling Play/Running at Speed

- **Change the Circular Path Exercises** Process so it will do 20 iterations.
- Set the running speed to maximum.
- Press "Reset" and "Go".
- Experiment by pausing, stepping, and resuming the Process.

Exercise 3.3.2 Changing Current Value

- Change the Circular Path Exercises Process to do 100 iterations.
- Run at maximum speed.
- Pause the Process and modify the Current Value of one of the Data Items so that the Decision will route the flow to the End stage.
    - *Tip: Think about how the Decision determines how the flow should stop going around the loop.*

3.4. Set Next Stage

A running process can be paused and made to jump directly to another position.

Exercise 3.4.1 Set Next Stage

- Run the Circular Path Exercises Process again and pause somewhere in the middle.
- Right-click on the End stage and select **Set Next Stage** from the mouse menu.
- Press "Go".

When running a Process, this feature can be very useful for skipping past a section of a diagram. However, be aware that Set Next Stage may have an undesired effect if the new position contains logic, dependent on the section that was skipped over.

Set Next Stage is also useful in "replaying" sections of a diagram by jumping back to an earlier position.

Key Point

- Set Next Stage does not “fast-forward” or “rewind” but simply “jumps” forward or back.

3.5. Breakpoints

A process can be set to pause on a specific stage if necessary.

Exercise 3.5.1 Breakpoint

- Right-click on the Calculation in the **Circular Path Exercises** Process.
- Select “Breakpoint” from the mouse menu. Notice the red highlight around the Calculation.

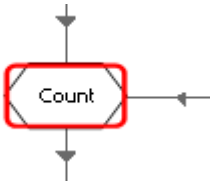


Figure 16: Breakpoint Stage

- Rerun the Process and see how it pauses at the Breakpoint and displays a message.

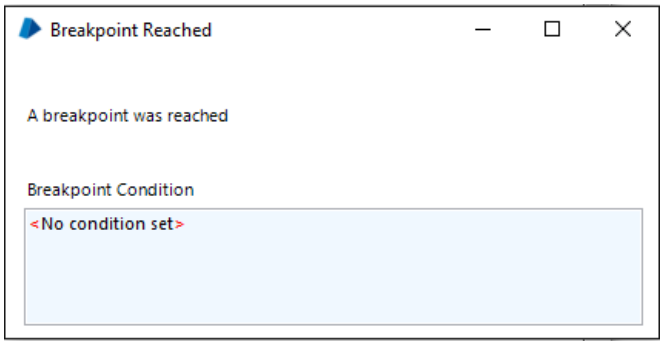


Figure 17: Breakpoint Message

- Continue running by pressing Go (or by closing the message and then pressing “Go”).
- When the Process pauses at the Breakpoint again, close the message, right-click on the Calculation, and select from the mouse menu to remove the Breakpoint.
- Continue running by pressing “Go”.
- Pause the Process, save, and then close **Circular Path Exercises**.

Key Point

- Breakpoints only take effect when the diagram is open. In the Production environment, the diagram is not displayed when a Process runs and Breakpoints are ignored.

3.6. Collections and Loops

The Data Items we’ve used so far have been used to hold single values but there is also way to hold ***multiple*** pieces of data together.

A Collection is a type of Data Item that can hold multiple values arranged like a table with columns and rows, similar to an Excel spreadsheet.

Collection data are accessed one row at a time using a Loop stage to move forward through the rows. Again, a programming comparison can be made by likening a Collection to an ***array*** and a Loop stage with a ***"for each"*** statement.

Exercise 3.6.1 Collections and Loops

- Open the example Process named ***Collection Example***.
- Find the Collection named ***My Orders*** and double click to open the properties window.
- See how the Collection has column definitions for Field Name and Type, plus tabs for rows of Initial Values and Current Values.

Figure 18: Collection Toolbar Button

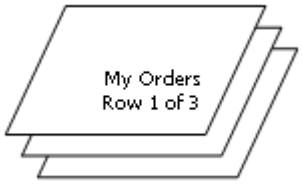


Figure 19: Collection Stage

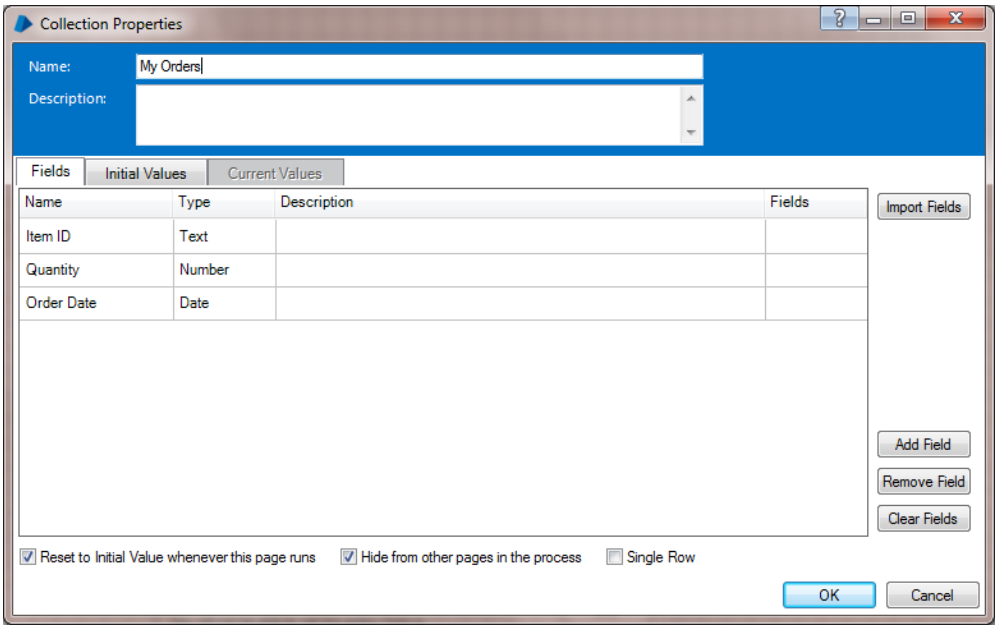


Figure 20: Collection Properties (Fields Tab)

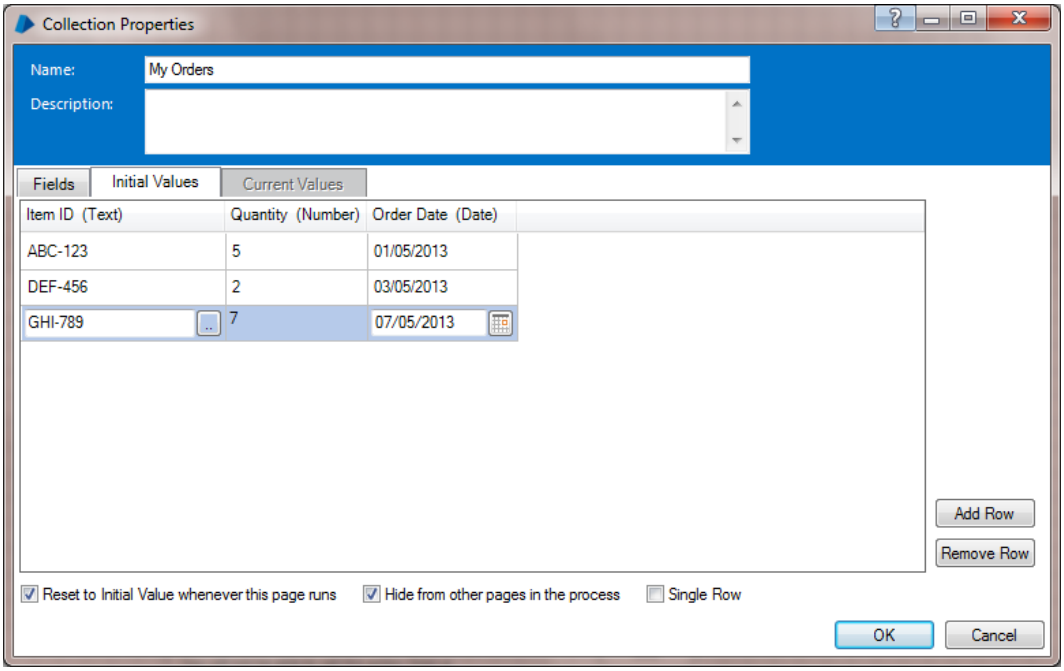


Figure 21: Collection Properties (Initial Values Tab)

- Close the properties window.
- Add a Loop stage to the diagram.

Figure 22: Loop Toolbar Button

- Notice that, unlike most stages, the Loop stage has two parts.

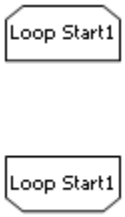


Figure 23: Loop Stage

Open the properties of the Loop start.

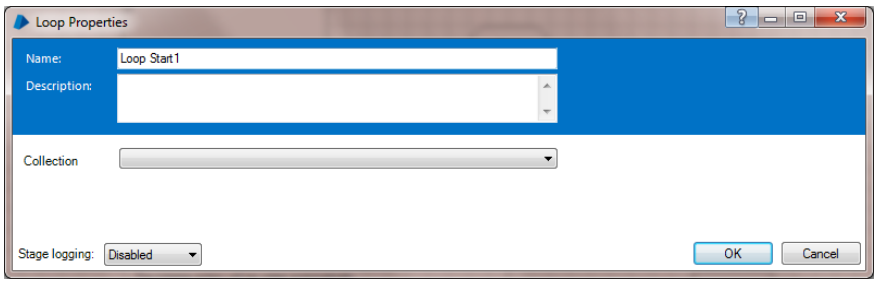
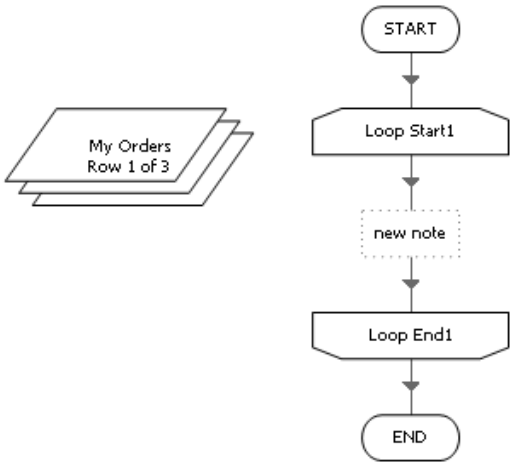


Figure 24: Loop Properties

- Set the Collection drop-down to **My Orders**.
- Add a Note stage between the start and end of the Loop (we will look at what Notes are for later in the course).
- Add links from the Start stage to the top of the Loop, to the Note, to the bottom of the Loop, and then to the End.

The resulting Process will probably look similar to the diagram below:



- Run the Process.
- See how the Process turns back on itself as it loops through the Collection.
- Notice also how the text in the Collection Data Item changes as the Process runs.

Exercise 3.6.2 Collections in Expressions

Collection fields can be used as part of an expression. For instance, we can perform Calculations using the Collection data with a Calculation stage positioned between the start and end of the Loop. As the Loop iterates through the Collection, each row is made available to the Calculation and can be used in the Calculation's expression.

To include a Collection field in an expression, we use what is known as dot notation. All this means is that both the Collection name and the field name must be included in the expression, and to do this you simply together using a full stop, for example **[My Orders.Quantity]**.

Key Point

- Dot notation is used to refer to Collection fields, i.e., "Collection Name.Field Name"

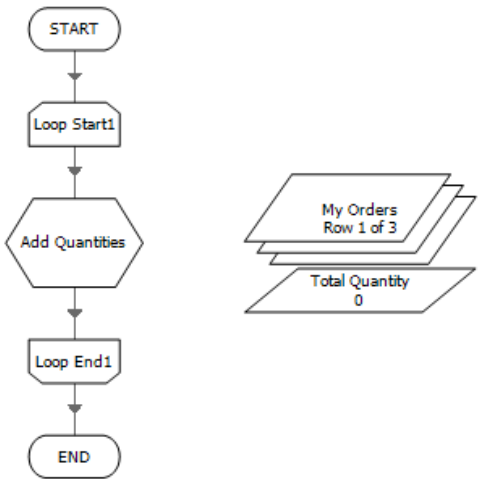
Exercise 3.6.3 Calculating the Total Quantity

In this exercise, we will continue with the ***Collection Example*** Process and add up all the Quantity values in the My Orders Collection.

- Add a number Data Item named **Total Quantity** and set the initial value to zero.
- Delete the Note and in its place put a Calculation stage named **Add Quantities**.
- In the Calculation properties, create an expression to calculate the total quantity of items.

- *Tip: Find My Orders in the list of Collections on the right of the properties window and drag in the appropriate column name.*
- *Tip: Think of Total Quantity as the running total when the loop iterates through the rows of the Collection.*
- *Tip: Remember that you can use the "Store Result In" Data Item name in the expression.*

Your ***Collection Example*** Process should now look similar to this:



And your Calculation expression should be **[My Orders.Quantity] + [Total Quantity]**

Exercise 3.6.4 Finding the Earliest Order Date

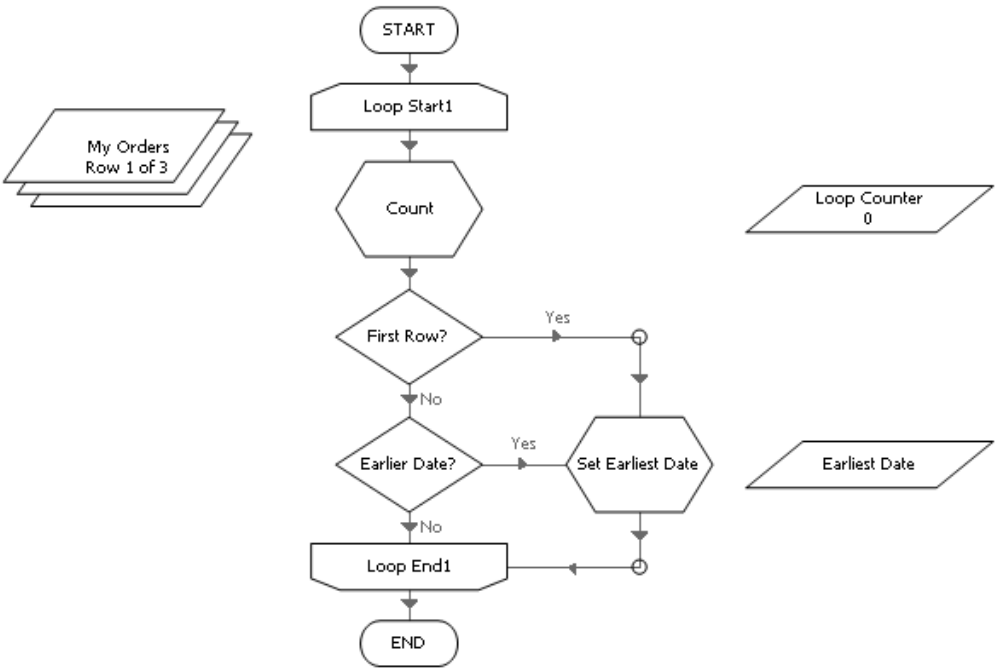
Continuing with the ***Collection Example*** Process, we will now find the smallest value of ***Order Date*** in the in the ***My Orders*** Collection.

- Delete the **Total Quantity** Data Item and Calculation stage added in the previous exercise.
- Make space inside the loop by dragging Loop End1 and End stages downwards.
- Add a Data Item named **Loop Counter** with data type **Number** and initial value **0**.
- Add a Data Item named **Earliest Date** with data type **Date** and initial value left blank.
- Add a Calculation stage named **Count** underneath the start of the Loop.
- Set the Calculation expression so that it will add **1** to **Loop Counter**.
- Add a Decision stage named **First Loop?** underneath the Calculation.
- Set the Decision expression to check if **Loop Counter** is equal to 1.
- If Loop Counter is 1, we must be on the first row of the Collection, meaning the date on that row is the earliest date **so far**, and we can use a Calculation stage to set the value of Earliest Date.
- If **Loop Counter** is not 1, we must have already moved **beyond** the first row and can assume that the **Earliest Date** Data Item contains a value.
- Add a second Decision stage named **Earlier Date?** Underneath the first Decision stage.
- Set the Decision expression to check if the **Order Date** value in the Collection row is **earlier** than the value in the **Earliest Date** Data Item.

- If the date in the Collection row is earlier, then it is now the earliest date so far and we can use a Calculation stage to change the **Earliest Date** Data Item.
- If the date in the Collection row is not earlier, then the **Earliest Date** Data Item still contains the earliest date so far and does not need to be changed.

> *Tip: To move to the next row of a Collection, link to the end of the Loop, not the start. Linking to the start resets the Loop to begin from the first row of the Collection.*

Your ***Collection Example*** Process should now look similar to this:



Save and Close the ***Collection Example Process***. We will look at ways to create and modify Collections later on.

3.7. Layers of Logic

Real-world processes contain large amounts of flow structures, stages, and data items and often a significant amount of space is required to arrange them.

Like other business flow diagrams, a Blue Prism Process can be spread over a series of pages arranged in a hierarchy.

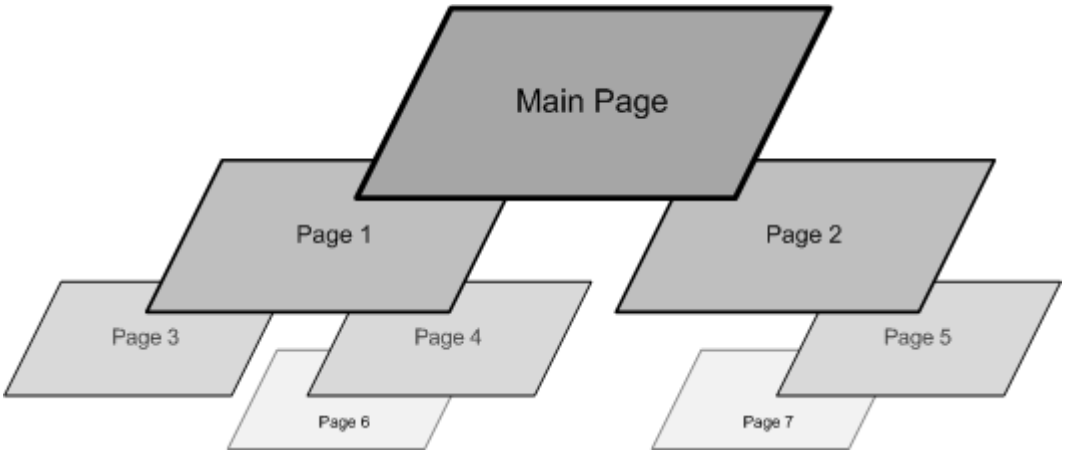


Figure 25: Page Hierarchy

Exercise 3.7.1 Adding a Page

- Open the **Circular Path Exercises** Process again.
- Right-click on the Main Page tab at the top of the diagram and select “New”.
- Accept the default name **Page 1**.
- Link the Start stage of the new page to the End stage.
- Go back to Main Page and add a Page Reference stage.

Figure 26: Page Reference Toolbar

- When you place a new Page Reference stage on the diagram, a window will appear asking which page you want to refer to. **Select Existing Page** and **Page 1**.

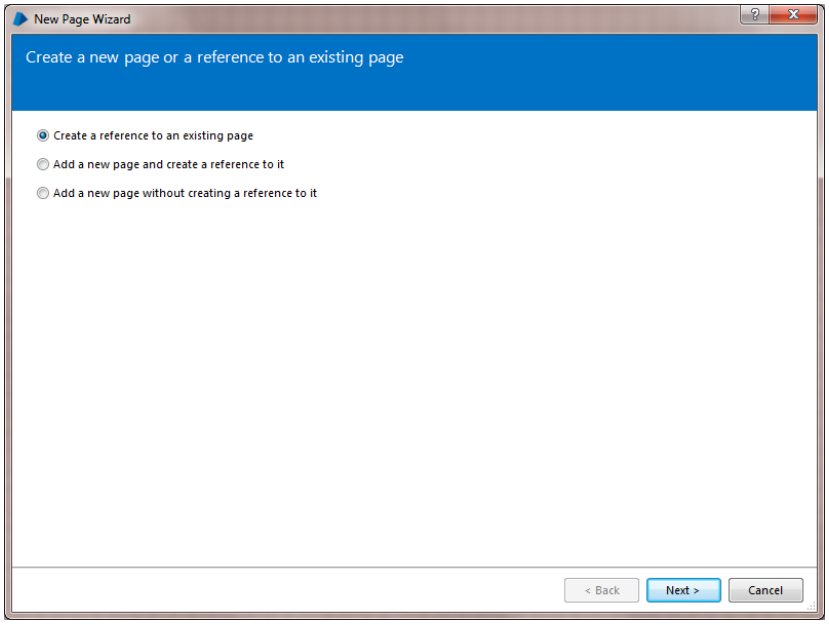


Figure 27: New Page Wizard

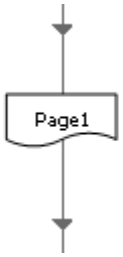


Figure 28: Page Reference Stage

- Open the properties of the Page Reference stage and see that it refers to the new Page 1 page.

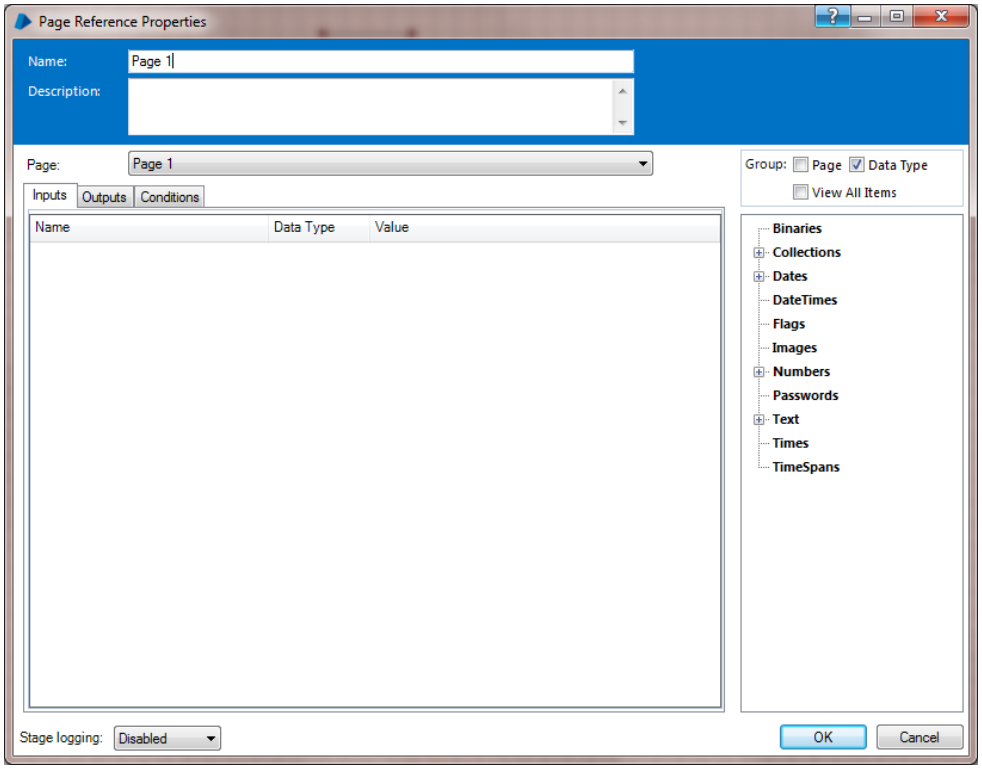
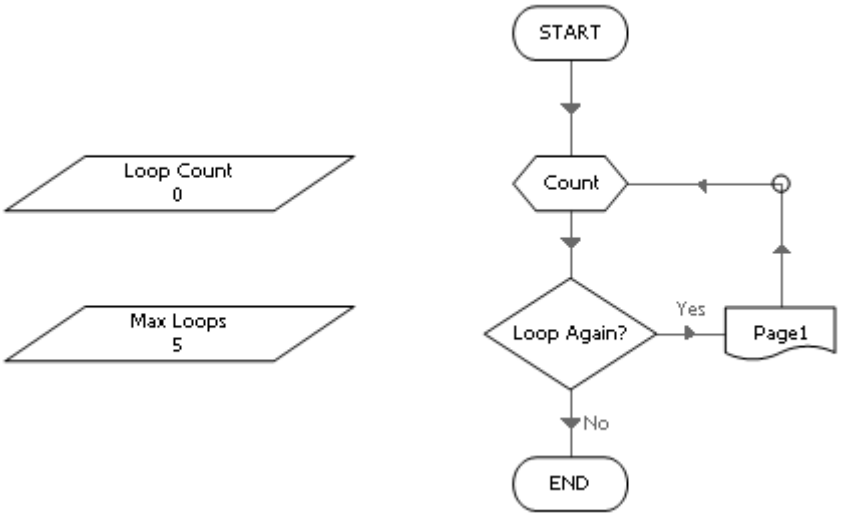


Figure 29: Page Reference Properties

- Position the Page Reference stage so that it sits on the path looping back up to the Calculation.
- Remake the links so that the Page Reference stage is connected into the diagram.
- Limit the maximum loops to five.
- Run the Process at normal speed.

Your ***Circular Path Exercises*** Process should now look similar to the picture below:



A Page Reference stage takes the Process down to the start of another page. When the end of that page is reached, the Process comes ***back up*** to the Page Reference stage that called it.

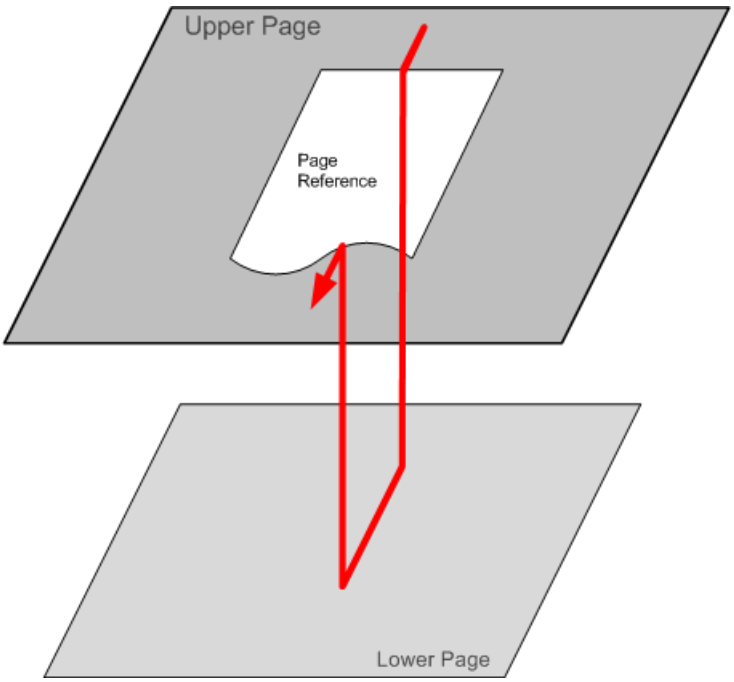


Figure 30: The Logical Flow through a Page Reference Stage

Exercise 3.7.2 Nested Page References

A Process can have multiple levels of interconnected pages, i.e., "pages within pages". This sort of hierarchical arrangement is sometimes known as nesting. Staying with the ***Circular Path Exercises*** Process, follow these steps to see how this works:

- Add another new page and as before keep with the default name **Page 2**.
- On the new page, link the Start stage to the End stage.
- Go to **Page 1** and add a new Page Reference stage that refers to **Page 2**.
- Position the new Page Reference between Start and End and link the three stages together.

- Go to the **Main Page** and run the Process again.

Exercise 3.7.3 Deleting a Page

Pages can be deleted if necessary but care must be taken with any references to a deleted page. Follow these steps to see the effect on your Circular Path Exercises Process:

- Right-click on the Page 2 tab and select Delete from the mouse menu. Answer "Yes" to the subsequent confirmation message

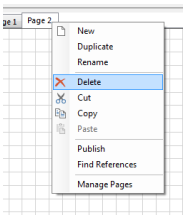


Figure 31: Page Mouse Menu

Exercise 3.7.4 Page Mouse Menu

- Go back to **Page 1** and open the properties window of the **Page 2** reference. Notice that the Page drop-down is set to **None** – this is because the Page Reference has been **orphaned**

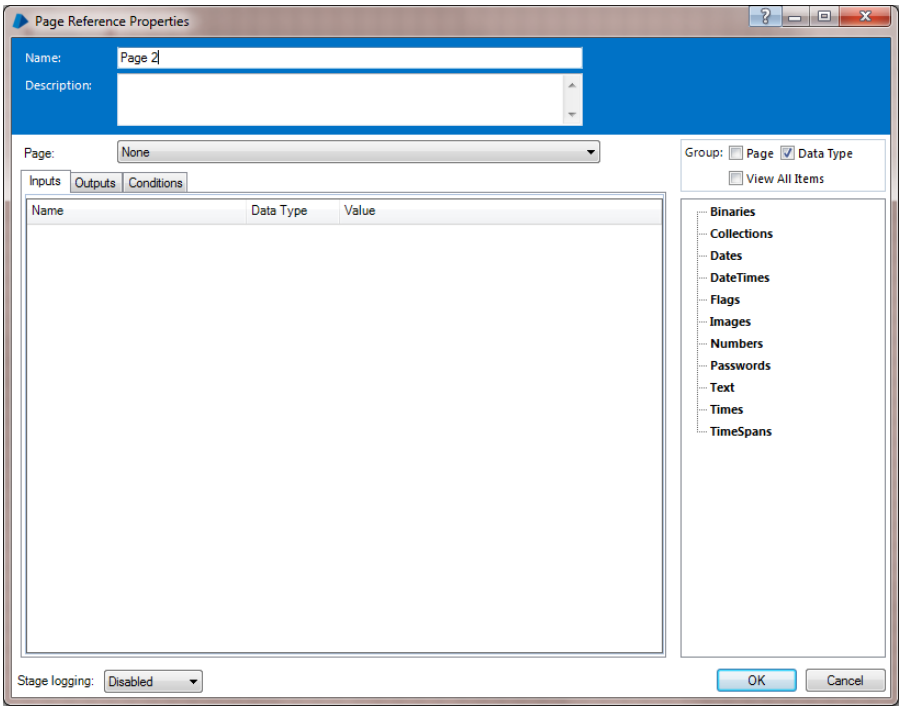


Figure 32: An Orphaned Page Reference

Exercise 3.7.5 Orphaned Page Reference Stage

- Delete the orphaned Page Reference stage.

3.8. Pages for Organization

When you come to develop "real-life" Processes, you will find diagrams can become large and unwieldy, and separating the logic out on to more pages will help to keep your diagrams tidy and readable. You will almost

certainly not be working alone on a Blue Prism solution and, as with any shared document, it makes sense to keep things clear and intelligible to others.

Exercise 3.8.1 Cutting and Pasting

For this exercise, we will imagine that the logic on the Main Page ***Circular Path Exercises*** should be moved to another page so that it can be re-used.

- Delete the Page Reference from Main Page and replace it with an Anchor stage.
- Draw a selection box around everything on Main Page apart from the Start, End, and Information Box.
  - *Tip: Remember to select the Data Items as well as the stages.*

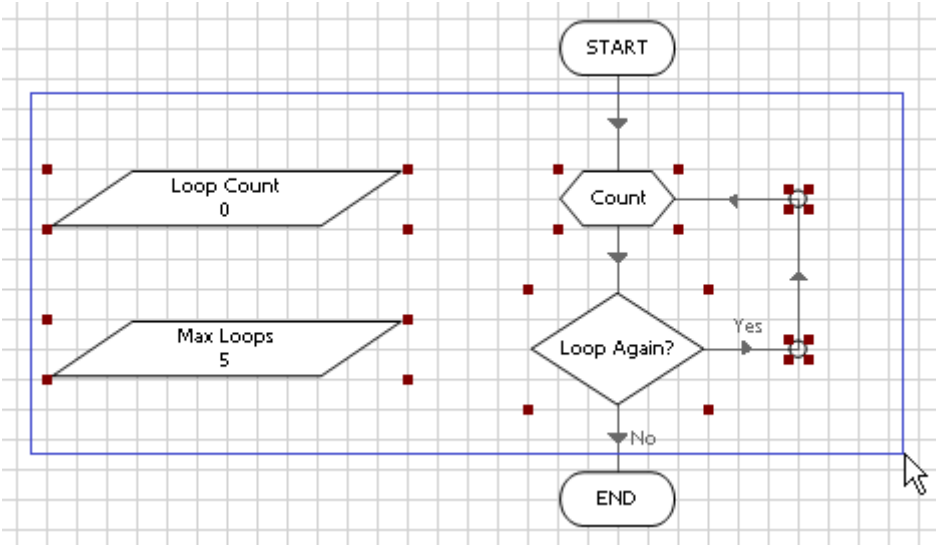
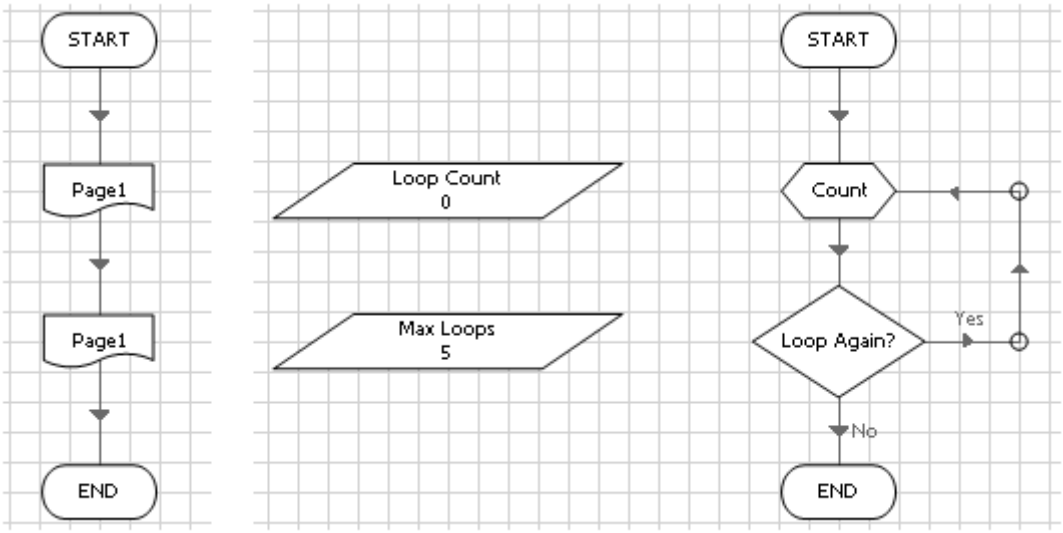


Figure 33: Selecting Stages

- Cut the selection off Main Page.
- Tip: Use either the menu Edit->Cut, CTRL+X, or the toolbar button,  .
- Paste the selection onto Page 1 and link the Start and End stages.
- Go back to the Main Page and add two new Page References that both call Page 1.
- The two pages of the **Circular Path Exercises** Process should now look like this:



- Press “Reset” and “Go” to see that the looping circuit runs is executed twice.

Save the Circular Path Exercises Process.

3.9. Review

- Set Next Stage and Breakpoint can be used when stepping through a Process.
- A circular path is a common logical structure.
- Collections are Data Items that can hold multiple pieces of data in columns and rows.
- Loop stages are used to move forward through the rows of a Collection.
- A Process diagram can run at variable speeds or be stepped through one stage at a time.
- Pages are used to break a large Process up into smaller sections.
- Pages are arranged hierarchically or nested.
- The Main Page is at the top of the hierarchy.
- A Page can reference other pages any number of times.
- A Page can be referenced by other pages any number of times.
- The exception is the Main Page, which cannot be referenced.

Blue Prism Keywords

Set Next Stage, Breakpoint, Loop, Iteration, Go, Pause, Step, Page, Page Reference, Nesting, Main Page.

4. Inputs and Outputs

Referring back to the ***Circular Path Exercises*** Process, the limitation of the circuit on Page 1 is that it will do the same number of cycles both times. It would be useful if the Main Page could tell Page 1 how many cycles to perform, then the Main Page could use Page 1 to do any number of iterations.

This can be achieved by "telling" each reference to Page 1 the number of iterations required. This information is known as an ***Input Parameter***, or Input for short.

4.1. Input Parameters

Exercise 4.1.1 Page Inputs

In this exercise, we will add an input to Page 1 of the ***Circular Path Exercises*** Process.

- Go to Page 1 and open the Start stage properties window.
- Look at the empty inputs list and try to add a new input parameter.

> *Tip: Notice that the right-hand column is named **Store In**. We have seen something similar before on the properties window of the Calculation stage.*

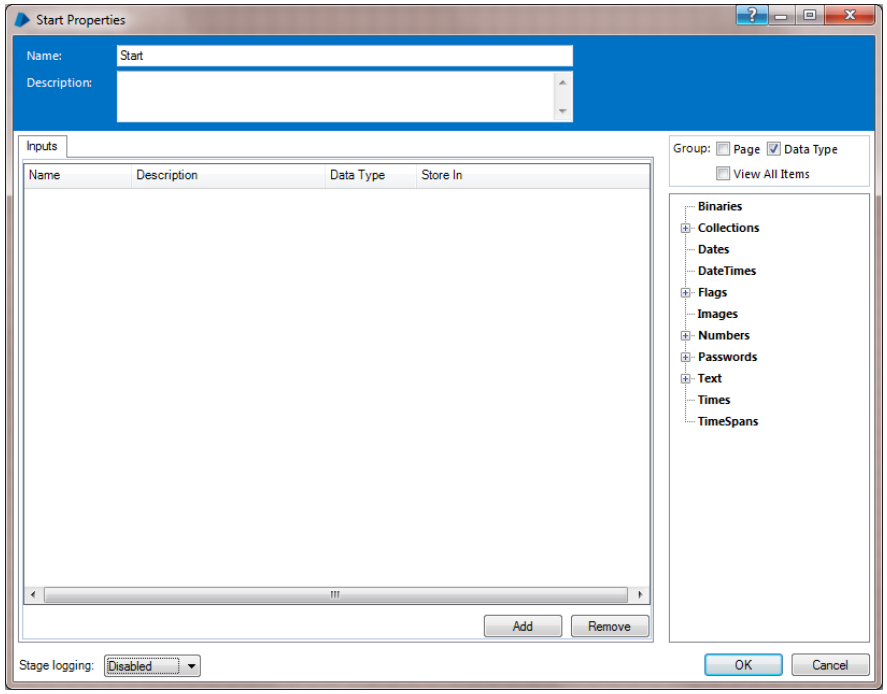


Figure 34: Start Properties

- Press the Add button to create a row in the Inputs table.
- Enter the **Number of Iterations** in the Name field.
- The Description field is optional, but its good practice to provide additional detail, e.g., "The number of iterations to be performed - must be zero or greater".
- Locate the Data Item **Maximum Loops** from the right-hand tree (under Numbers) and drag Maximum Loops into the "Store In" column.
- Notice how the Data Type is automatically set to that of the Data Item (i.e., Number).
- Close the Start properties window.

Exercise 4.1.2 Inputs and Page References

Now we have set up an Input Parameter on Page 1, we'll look how that has affected the Page Reference on the Main Page of the ***Circular Path Exercises*** Process.

- Go to the Main Page and open the properties of the Page Reference stage.
- Notice that the Inputs tab has a new row on it, and that the Value field is empty.

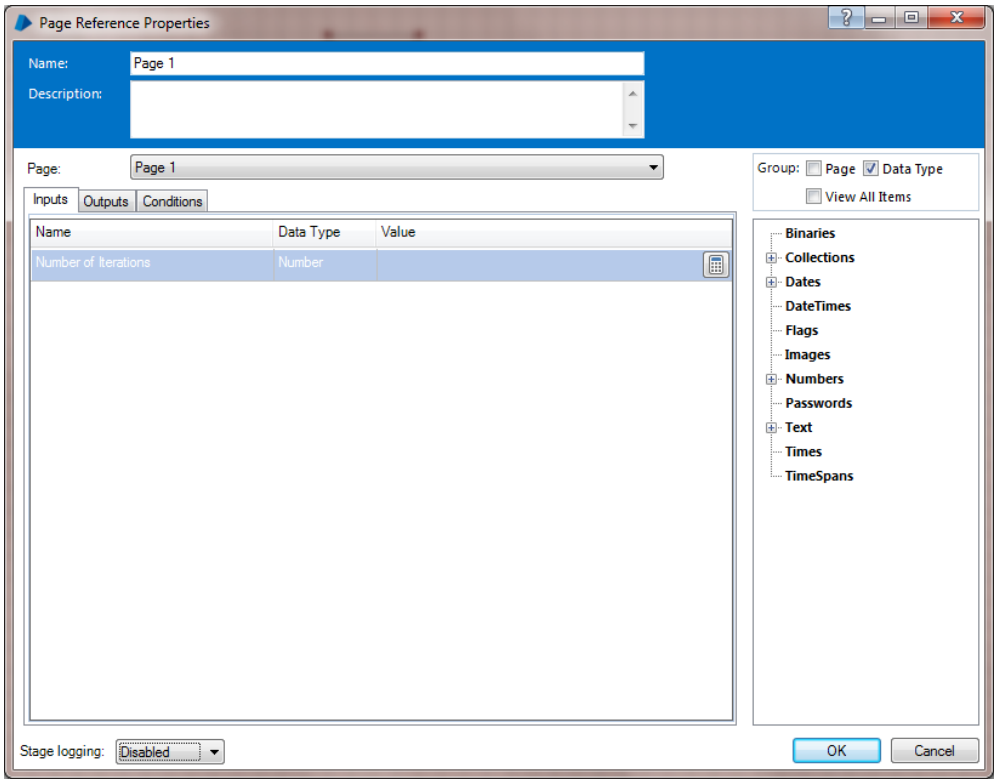


Figure 35: Page Reference Properties

- Press the Expression Editor button,  , in the Value column to see a familiar-looking screen appear.

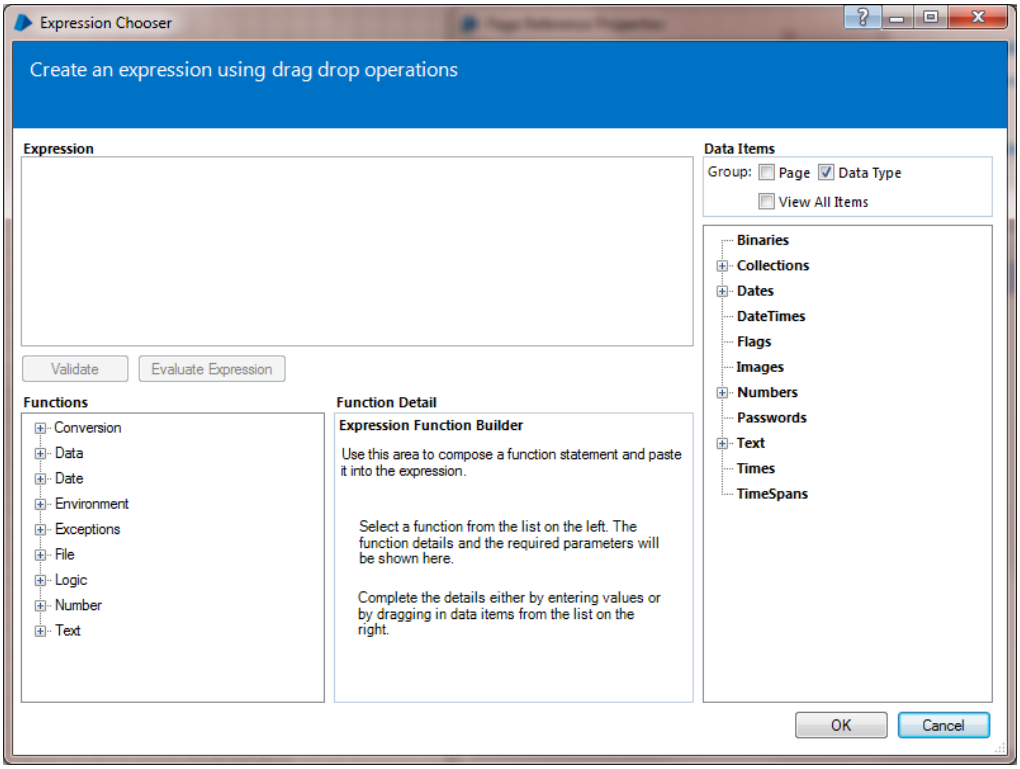


Figure 36: Expression Chooser Screen

- The Expression Chooser screen appears because the value for an input is supplied as an expression.
- Close the Expression Chooser screen and enter **2** in the Value field.

  * *Tip: You don't have to open the Expression Chooser – you can enter the expression directly into the Value field.*

- Open the second Page Reference and set it's input value to **3**.
- Press "Reset" and "Go".
- Notice how the Page Reference stages each transmit their values to the Start stage on Page 1. The Start stage then sets the current value of the Data Item specified in the "Store In" column.

Key Point

- As well as being used to sub-divide and organize a diagram, pages can also be used to make reusable functionality.

The Input Value used by the Page Reference stage is calculated from an expression and, as we have seen, an expression can use the values from Data Items.

Exercise 4.1.3 Input Values from Data Items

In this exercise, we will use Data Items instead of numbers typed directly into a Page Reference stage. So continuing with the ***Circular Path Exercises*** Process, follow these steps:

- Delete one of the Page Reference stages from Main Page and link the remaining Page Reference to the End stage.
- Add a new Data Item named **Number of Iterations** with data type **Number** and initial value **4**.

- Open the Page Reference properties and remove the expression in the Value field.
- Use the Data Item in a new expression, either by dragging the Data Item in from the right or by typing it in, i.e., **[Number of Iterations]**.
- Press "Reset" and "Go".
- Notice how the value of the Data Item on Main Page is transmitted by the Page Reference via the Start stage to the Data Item on Page 1.
- Save the Circular Path Exercises Process.

Key Point

- An input value is supplied as an expression.
- Usually this expression will simple (e.g., the name of a Data Item like [Full Name]) but could be more complex (e.g., [Forename] &" "& [Middle Name] & " " & [Surname]).

4.2. Stepping and Pages

We have seen that we can step through a diagram one stage at a time. We can also use Step Over and Step Out to traverse pages more quickly.

Exercise 4.2.1 Step Over

In this exercise, we'll see how Step Over can be used to execute a whole page in one go. So, within the ***Circular Path Exercises*** Process, follow these steps:

- Change the initial value of the **Number of Iterations** Data Item to **100**.
- Reset and step forward once, either by using the Step button,  , or by pressing F11.
  - *Tip: The Tools and Debug menus show you some other function keys available.*
- With the page Reference stage in focus (i.e., highlighted orange) press the Step Over button,  .
- Notice how the Process does not flow down to Page 1 but stays on Main Page.
- Notice also that there is a slight delay before the focus moves the End of Main Page. This is because the Process is actually executing the 100 iterations but **without displaying its movements** on Page 1.

Exercise 4.2.2 Step Out

In this exercise, we'll see how Step Out can be used to execute the remainder of a page in one go. So, within the ***Circular Path Exercises*** Process follow these steps:

- Reset and step forward a handful of times, either using the Step button,  , or by pressing F11.
- When you are somewhere in the middle of Page 1, press the Step Out button,  .
- Notice how after a slight delay the Process suddenly moves back up to Main Page. This is because the Process is executing the remainder of the 100 iterations but **without displaying its movements** on Page 1.

4.3. Data Item Visibility

By default, Data Items can only be used by stages on the same page as them, but this can be changed if necessary.

Exercise 4.3.1 Local and Global Data Items

In this exercise, we will alter the visibility of a Data Item to make available to all pages. So within the ***Circular Path Exercises*** Process, follow these steps.

- Go to Page 1 and open the Calculation stage properties.
- Look through the tree on the right-hand side and find the Data Item located on the Main Page named **Number of Iterations**.
- Notice that some Data Items in the tree are grayed out.
- If some items are not displayed please ensure "View All Items" is checked.
- Close the Calculation properties and go to the Main Page.
- Open the **Number of Iterations** Data Item properties and un-check"Hide from other pages in the Process" checkbox.
- Go back to Page 1 and open the Calculation properties again.
- See how the Data Item is no longer grayed out. Data Items that are visible to all Pages are known as **global** Data Items. Data Items visible only to their own page are known as **local** Data Items.
- Save the Circular Path Exercises Process.

Exercise 4.3.2 Visualizing Data Items

The diagram below will help you visualize local and global Data Items.

Exercise 4.3.3 Local and Global Data Items

- The three Local Data Items on Page A cannot be seen by any other Page. Similarly, the two Local Data Items on Page B are inaccessible to all other pages.
- The Global Data Item Global A1 lives on Page A, but is useable by all Pages.
- Global D1 is on Page D, but is also available everywhere.
- So even though there are no Data Items actually sitting on it, Page C can still make use of two Data Items from other Pages - Global A1 and Global D1.

Exercise 4.3.4 Using Global Data Items

You may be thinking that global Data Items (or ***Globals*** for short) could be very useful and wondering why all Data Items shouldn't be made global. Although that is a possibility, there are considerations to be made when using global Data Items:

- Global names must be unique, whereas locals can share the same name provided they "live" on different pages.
- A global can be manipulated from any page, and therefore you must be aware of all the points in a Process where it is used. The points affecting a local can only be on the same page as the Data Item.
- Keeping track of the current value of a global, particularly when working one case after another, needs careful attention. A global can present the risk of a value from a previous case affecting the next case – something that needs to be addressed in the design of a Process.

4.4. Data Types

So far, we have only used Data Items that hold numbers, but there are in fact ten data types to choose from. The data types most commonly used are as follows:

Data Type	Description
Number	Used for any numeric value
Text	Used for any alphanumeric value
Flag	Used for True or False values
Date	Used for date values
Password	Used for sensitive data

The less common data types are as follows:

Data Type	Description
Datetime	Used for date and time data
Time	Used for time values
Timespan	Used to store a length of time
Image	Used to hold an image in a Data Item
Binary	Used to store binary data, like the contents of a binary file

Exercise 4.4.1 Using Data Types

The data type you will probably use most often will be Text, particularly when reading from files or an application.

Some values may genuinely be Text, e.g., "Smith", and some may be other types of data presented as Text, e.g., "09/17/1982". Although it is possible to transform a value from one data type to another (and we will look at ***casting*** later on), consideration should be given as to whether it is actually necessary to do so.

For example, if a telephone number was read from an application as Text, the Data Item value would be an exact copy of the value in the application. However, if the value was read as a Number, the "leading zero" would be truncated. You will undoubtedly have already come across a similar issue in Excel, where you type "01234"into a cell and Excel displays "1234" - this is because Excel is assuming you are entering a numeric value and kindly transforms your input for you.

Similarly, if you have a value represented as Text that could be transformed into its "true" data type, consider whether it is actually necessary to do so.

For example, if you have read "Date of Birth" into a Text Data Item and need to work out "Patient Age", then at some point you will need "Date of Birth" stored in a date Data Item before you can use a Calculation stage to determine "Patient Age". However, if your Process does not need to use "Date of Birth" in any expression, then the best option is to make things simple and keep the value stored as Text.

4.5. Output Parameters

Mirroring how Start stages can accept Inputs, End stages can produce ***Output Parameters*** or ***Outputs*** for short.

We have seen that Pages are arranged in a hierarchy, and that an Input is used to transmit a value from a Page Reference stage on a higher Page down to the Start stage of the lower Page. An Output is used to transmit a value from an End stage on a lower Page up to the Page Reference stage on the higher Page.

Earlier we saw that Pages can be used to avoid duplication and the following exercises demonstrate how a Page with both Inputs and Outputs can provide some reusable functionality.

Exercise 4.5.1 Get Future Date

In this exercise, we will make a page that can calculate a date in the future. So within the ***Circular Path Exercises*** Process, follow these steps:

- Add a new Page to the Process and name it **Get Future Date**.
- Add a Data Item named **Number of Days** with data type **Number** and initial value left blank.
- Add a Data Item named **Date Result** with data type **Date** and initial value left blank.
- Add a Calculation stage that uses **Date Result** as the "Store Result In" Data Item.
- In the Calculation properties window, look at the bottom left corner where you will see a list of Functions.

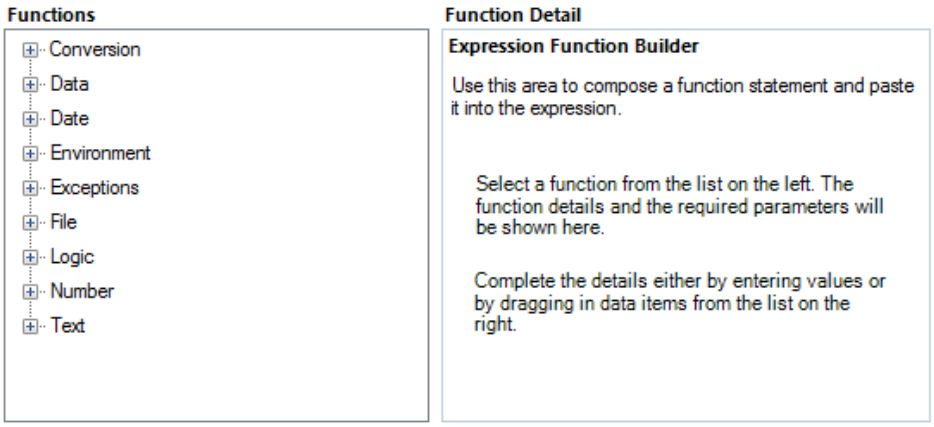


Figure 37: Expression Functions

Exercise 4.5.2 Expression Functions

- The functions are not unlike the functions you will have used in Excel, and you may recognize some of them. They are arranged in groups, and we want to look at the Date functions, specifically the "AddDays" function.

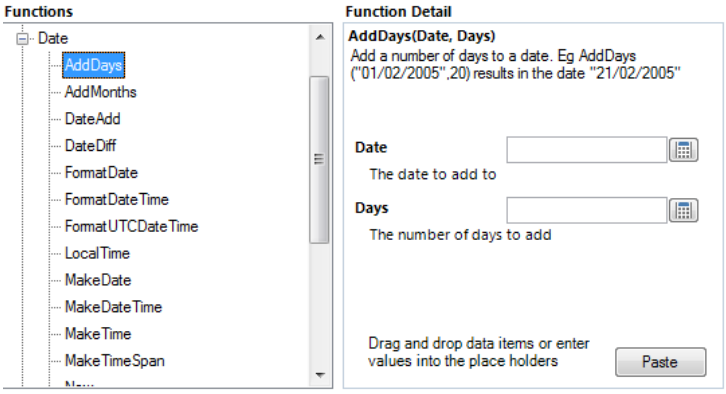


Figure 38: AddDaysFunction

Exercise 4.5.3 AddDaysFunction

- Notice how the "Function Detail" area displays a description of the selected function and offers input fields for the parameters of the function.
- From the tree view on the right, drag the **Number of Days** Data Item into the **Days** field.
- In the Date field, type in the expression **Today()** and then press **Paste**.
- Notice how the Expression area above is now populated with the expression **AddDays(Today(), [Number of Days])**.

    - *Tip: Once you are familiar with the syntax, you can type the function directly into the Expression area.*

    - *Tip: You can also drag Data Items directly into the Expression area.*

Exercise 4.5.4 Page Inputs and Outputs

In this exercise, we will modify the ***Get Future Date*** page of the ***Circular Path Exercises*** Process so it has both an input and an output parameter.

- Open the properties of the Start stage on the **Get Future Date** page.
- Add an Input named**"Days Hence"**, using the Data Item **Number of Days** in the "Store In" column.
- Open the properties of the End stage.

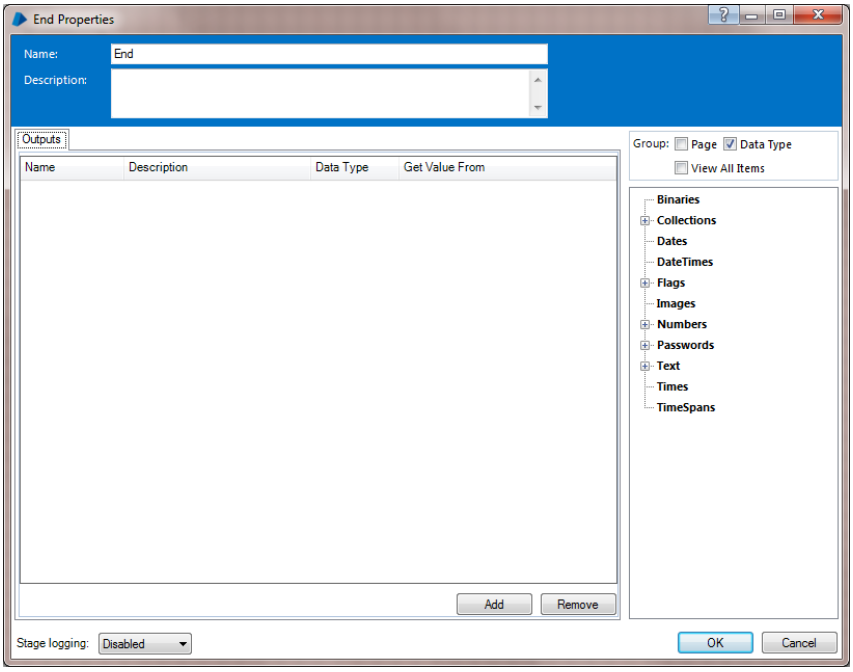
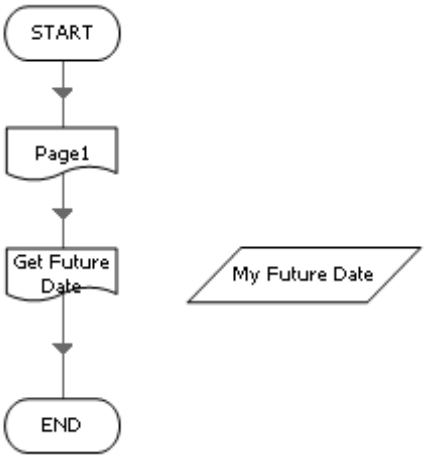


Figure 39: End Properties

- Add an Output named **Future Date**, using the Data Item named **Date Result** in the "Get Value From" column.

  Tip: Think of the Get Value From column as the opposite of Store In.

- Go back to Main Page.
- Add a new Data Item named **My Future Date** with data type **Date** and initial value left blank.
- Add a new Page Reference stage and set **Get Future Date** as the Page to refer to.
- Position the Page Reference so that your diagram looks something like this:



- Open the new Page Reference properties window.
- See how the Input tab has a row named **Days Hence** and the Output tab has a row named **Future Date**.

  Tip: Think of the Input row as the Page Reference "requesting" information.

Tip: Think of the Output row as the Page Reference "offering" information.

- Provide the Page Reference with data by entering **30** into the Value column on the Inputs tab.
- Accept data from the Page Reference by dragging the **My Future Date** Data Item into the "Store In" column on the Output tab.
- Close the Page Reference properties window.

The Main Page of the Process is now set up to exchange information with the Get Future Date page; Main Page will provide the number of days, and Get Future Date will return the date it calculated using the input it was supplied with.

- Step through the Process using **Step Over** to execute each Page Reference stage in one step.
- Notice that the Data Item **My Future Date** has been set to 30 days from today.
- Modify the Page Reference input so that a different date is calculated.

The End stage of the Get Future Date Page sends the value of the Date Result Data Item back to the Page Reference stage on the Main Page. The Page Reference stage then sets the Current Value of the My Future Date Data Item on the Main Page.

4.6. Start-up Parameters

Inputs can also be applied to the Process itself, enabling a Process to accept external values when it starts to run. Process inputs are known as ***Start-up parameters***.

Exercise 4.6.1 Start-up Parameters

In this exercise, we'll modify the ***Circular Path Exercises*** Process to give a Start-up parameter.

- Go to the Main Page and add an Input to the Start stage named **Number of Iterations**.
- Use the existing number Data Item **Number of Iterations** in the Store In column.
- Save and Close the **Circular Path Exercises** Process.

4.7. Control Room

Process Studio is only used for building and testing a Process. Although we can run a Process from the diagram, in an operational environment, Processes are not run in Process Studio; Processes are run from another part of Blue Prism named Control Room.

Key Point

- Diagrams are for development and testing purposes. Diagrams are not used to run Processes in the Production environment.

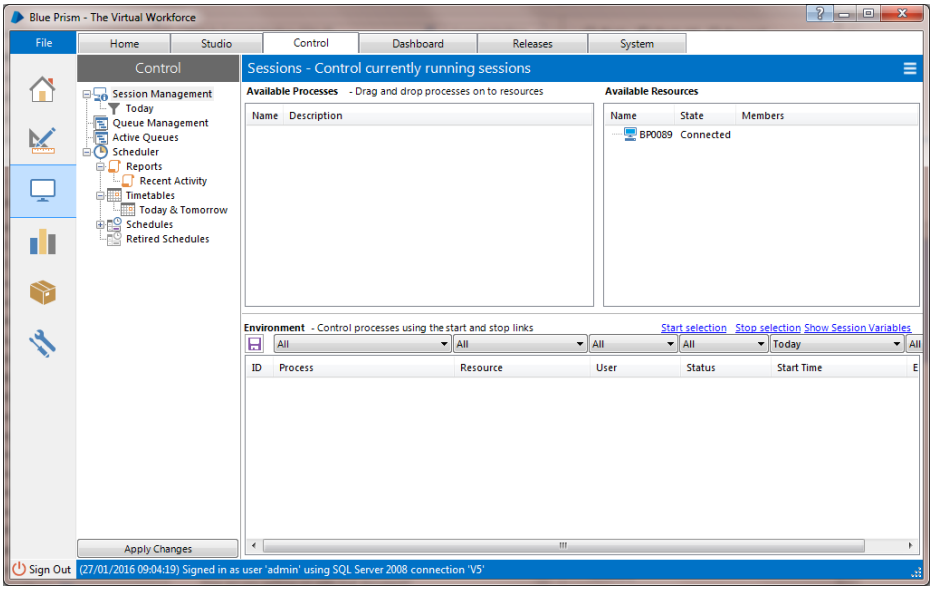


Figure 40: Control Room

Exercise 4.7.1 Publishing

By default a Process is not immediately available in Control Room. This is safety feature in Blue Prism to prevent unfinished Processes being run accidently.

- Go to Control Room to see that the Available Processes list is empty.

Only “published” Processes are visible in Control Room, and new Processes are always “unpublished” to begin with.

- Open the **Circular Path Exercises** Process and go to the Main Page. Open the properties of the Process Information stage (the large box, usually near the Start stage).

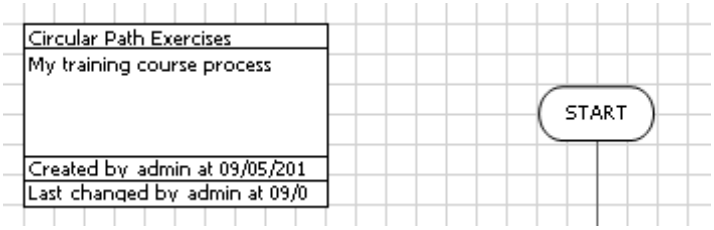


Figure 41: Page Information Stage

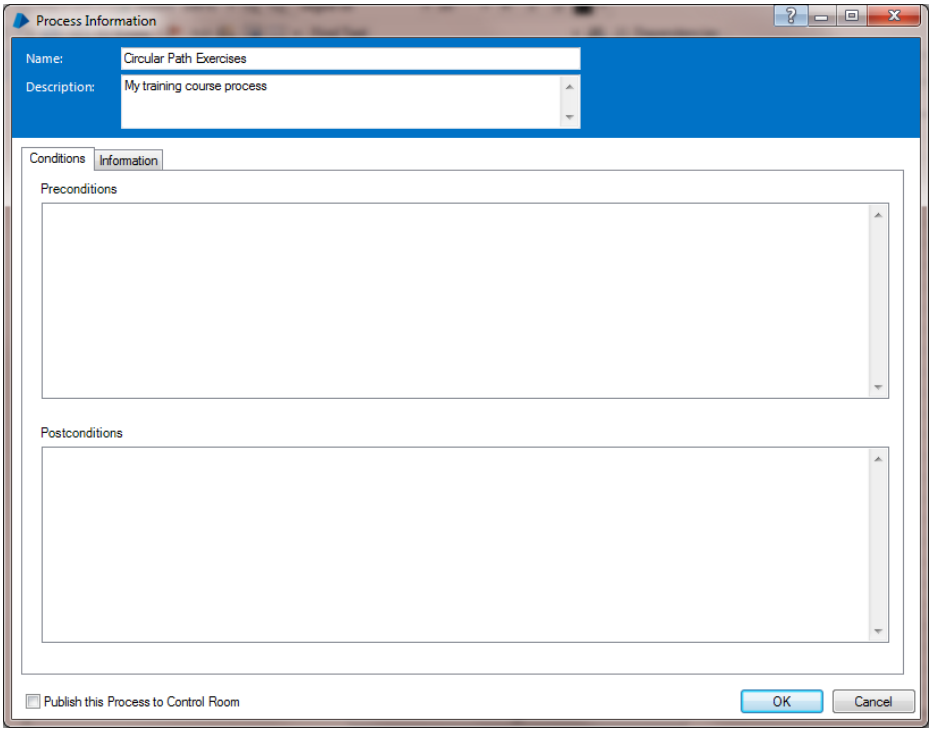


Figure 42: Process Information Properties

- Check the "Publish this Process to Control Room" checkbox at the bottom of the window.
- Save the Process.
- Go back to Control Room and press the refresh button.
- The Process should now be in the list of Available Processes.

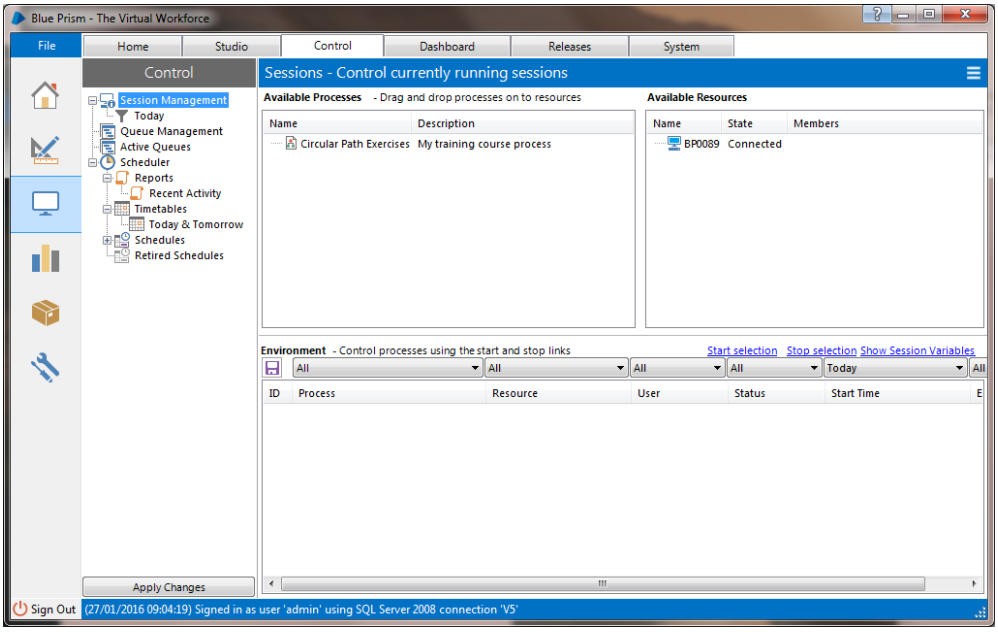
Although for these exercises we are running Processes on the local machine, it is likely that remote machines will be used in a real-world scenario.

Machines installed with Blue Prism are known as ***Resources***, and an instance of a Process running on a Resource is known as a ***Session***.

Exercise 4.7.2 Running a Session

In this exercise, we'll run a Session of a Process in Control Room.

- Go to Control Room and see how the Session management tree item is divided into three sections (we will come to the other tree items in due course).
- Find Circular Path Exercises in the Available Processes list on the left of the screen.



- See how your machine is displayed as a blue icon in the **Available Resources** list on the right of the screen.
- Drag the Process name onto the blue icon and see that a new Session row is added to the **Environment** list at the bottom of the screen.

  ✱ *Tip: You can also drag the blue icon onto the Process to achieve the same effect.*

- The Session is initially displayed in orange to indicate that its status is "pending", i.e., it is ready to start.
- Start the Session by selecting the orange row and clicking Start Selection.

  ✱ *Tip: The mouse menu provides the same functionality.*

- See how you are prompted to provide values for the Start-up parameter. Enter 100 and press the Start button.

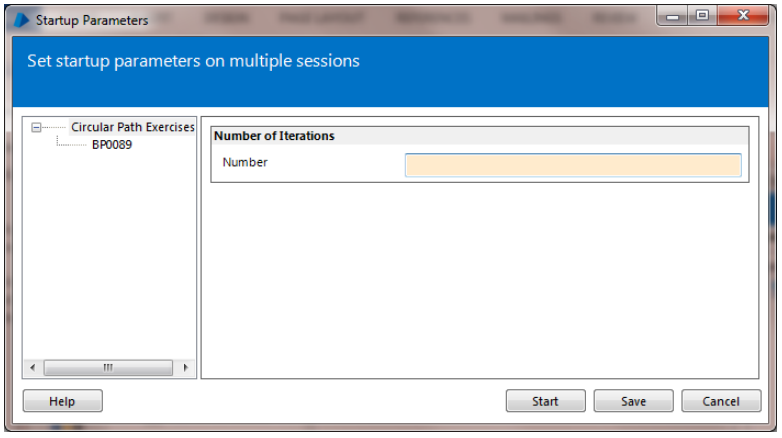


Figure 43: Start-up Parameters

- Notice the Status column change as the Process runs. Eventually the row should become blue to indicate the Session is complete.
- Create another session and give it a larger Start-up parameter value to make the session run for longer.

Key Point

- A Session can only run once; to run a Process again a new session must be created.

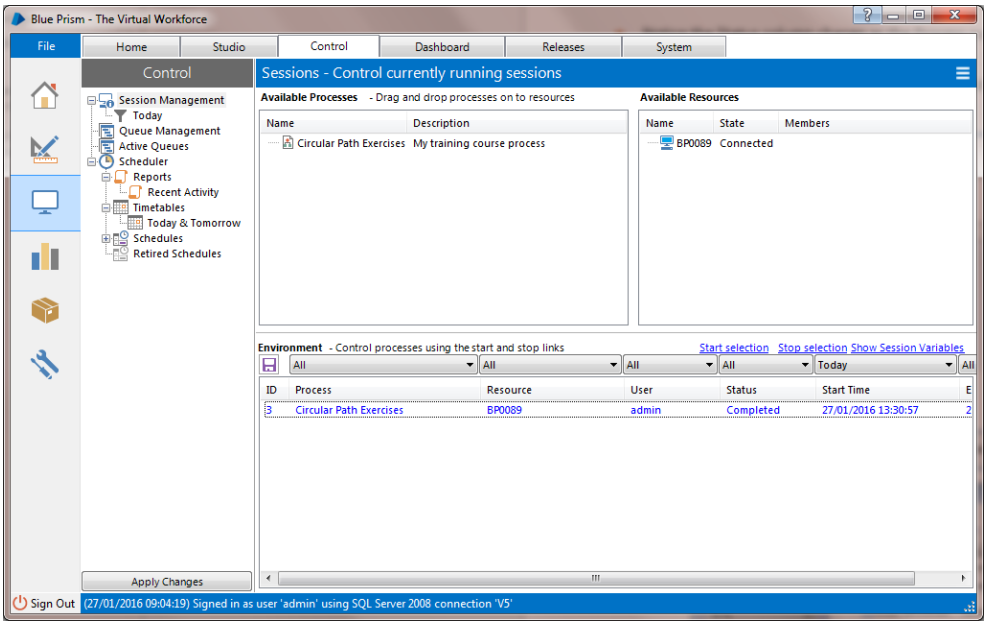


Figure 44: A Completed Session

Exercise 4.7.3 Running without Startup Parameters

In reality, Processes don't often use Start-up parameters because as you've just seen, they need to be entered manually from Control Room. In this exercise, we will change the Circular Path Exercises Process so it doesn't have any Start-up parameters.

- On Main Page open the Start properties and remove the Start-up parameter.
- Save the changes **without** closing the diagram.
- Go to Control Room and start another Session.
- Notice that now the Session starts without prompting you for input.

Exercise 4.7.4 Stopping a Session

A running Session can be stopped from Control Room if necessary. In this exercise, we'll start a long-running session of ***Circular Path Exercises*** and bring it to an early end.

- On Main Page, open the Start properties and recreate the Number of Iterations Start-up parameter as before.
- Save the Process.
- Go back to Control Room and run another session, this time with a large number (e.g.,**10000**) as the Start-up parameter.
- As the Session is running, select the (green) row and click Stop Session. Notice how the session status changes to "Stopped".

  ✱ *Tip: The mouse menu provides the same functionality.*

Exercise 4.7.5 Session Exceptions

A Process may encounter an error as it runs, causing the session to stop. In this exercise, we'll "break" ***Circular Path Exercises*** so that it does not run properly.

- Introduce an error into the diagram by deleting a link somewhere.
- Save the Process.
- Go back to Control Room and run another session. Notice how the session status changes to "Terminated".
- Fix the diagram error and confirm the repair by running another quick Session in Control Room.
- Save and Close the Process.

Exercise 4.7.6 Session Logs

Whenever a Process runs, it makes a record of each step it takes to create a ***Session Log***. You can access this log from Control Room.

- Select one of the Session rows in the Environment list in Control Room.
- Right-click to open the mouse menu and select **View Log**.
- See how each session records the details of each stage to create a log.
- Close the Log Viewer.

4.8. Process Outputs

A Process can call another Process using the Process Reference stage. This stage is similar to a Page Reference stage but refers to an entire Process rather than just one page.

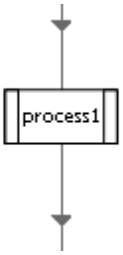


Figure 45: Process Reference Stage

A Process used in this way is often known as a sub-Process and a hierarchy of Processes, similar to a hierarchy of pages, can be created.

Like a page, a Process used in this way can also transmit outputs back up to the Process reference stage in the calling Process.

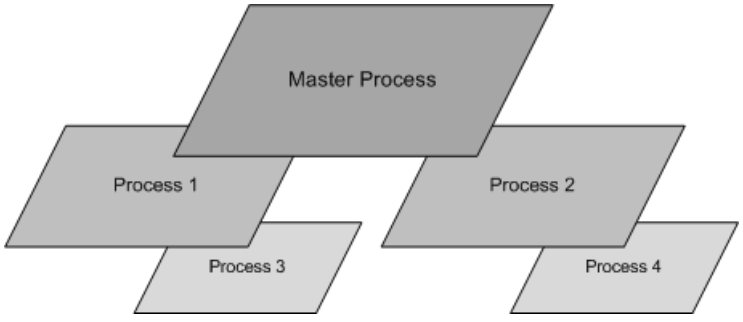


Figure 46: A Process Hierarchy

Exercise 4.8.1 Outputs and Sub-Processes

In this exercise, we'll add an Output to the ***Circular Path Exercises*** so it can be used as a sub-Process.

- Open the properties of the End stage on the Main Page.
- Create a new Output named **Future Date** using the value of **My Future Date** Data Item.
- Save and Close the Process.
- Create a new Process named **Circular Path Master**.
- Add a Process Reference stage to the Main Page and link it between the Start and End stages.

Figure 47: Process Reference Toolbar Button

- In the Process reference stage properties, select the **Circular Path Exercises** Process as the Business Process.

Exercise 4.8.2 Process Reference Stage

- Notice that the Process Reference **requests** an input and **offers** an output.
- Enter a small number value for the **Number of Iterations** input.
- Create a date Data Item and use it for the **Future Date** output.
- Save the new Process.
- Press "Go" to run Process in Process Studio.
- Notice how the whole Circular Path Exercises Process is executed in one step.

Exercise 4.8.3 Stepping into a Process Reference

Much in the same way as stepping down through a Page Reference to a lower page, it is possible to step down into a sub-Process via a Process Reference stage.

- Reset the **Circular Path Master** Process and press the Step button to advance to the Process Reference stage.
- Step again to see that the sub-Process **Circular Path Exercises** is opened in its own Process Studio window.
- Experiment with Step, Step Over,and Step Out to run the Process.

4.9. Review

- Data Items are normally local and only visible to their own page.
- Data Items can be made global so that they are visible to all pages.
- Inputs are used to transmit values from an upper page down to the start of a lower page.
- Outputs are used to transmit values up from the end of a lower page back to the upper page.
- Start-up parameters can also be used to transmit external values to the start of the Main Page.
- A Process can be called from another Process, and inputs and outputs can be used to transfer values between Processes.

Blue Prism Keywords

Step Out, Step Over, Data Type, Local, Global, Input Parameter, Output Parameter, Start-up Parameter, Control Room, Publish, Resource, Session, Process Reference, Sub-Process.

5. Business Objects

So far, our exercises have been theoretical examples intended to explain the mechanics of a Process diagram. A real Process would be required to do some useful tasks, and in order to do so it would invariably need to work with external applications.

The interface to an application is not contained within a Process diagram, but in a separate diagram known as a ***Business Object***.

Key Points

- A Business Object is not configured in Process Studio but in a distinct, parallel environment named ***Object Studio***.
- The purpose of a Business Object is to provide an interface to ***one*** application.
- A Business Object is not exposed to Control Room and is never used on its own – it is always used by a Process.
- Business Object logic can be thought of as forming a separate layer to Process logic.

5.1. Object Studio

Object Studio is used to capture the functionality of an application so that it can be employed by Processes. A Process is used to combine the application functionality with business rules in order to perform useful work. This application functionality is known as a ***Business Object***.

We will see that Object Studio looks very like Process Studio. There are key differences, which we shall look at in due course, but the principle points to know at this point are as follows:

- Object Studio is similar to Process Studio and a Business Object is configured as a diagram much in the same way.
- Object Studio offers some types of stage not available to Process Studio.
- Object Studio does not have a Main Page but it does have two default pages.
- Pages are organized as a flat group rather than the hierarchy we have seen in Process Studio.

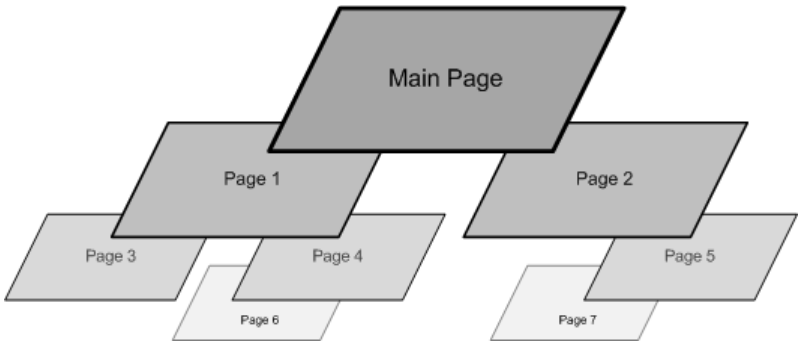


Figure 48: The Hierarchical Structure of Process Pages

Figure 49: The Flat Structure of Business Object Pages

5.2. Business Objects

A Process uses the pages of a Business Object to manipulate an application. Whereas a Business Process flows from a single start point on the Main Page, the flow through a Business Object can start on any page.

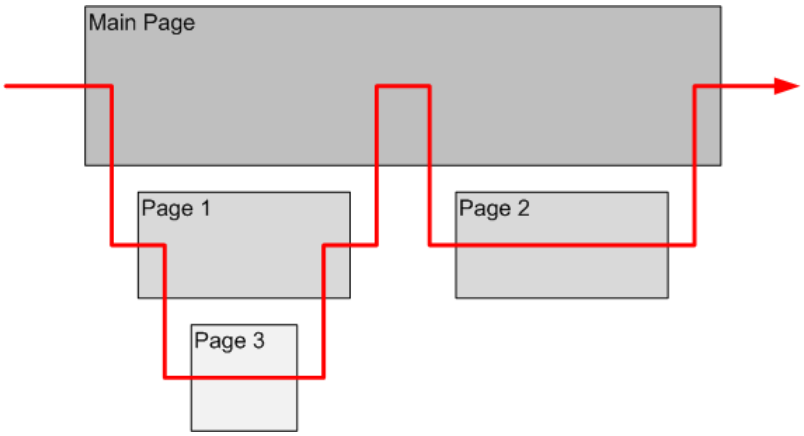


Figure 50: The Logical Flow through Process Pages

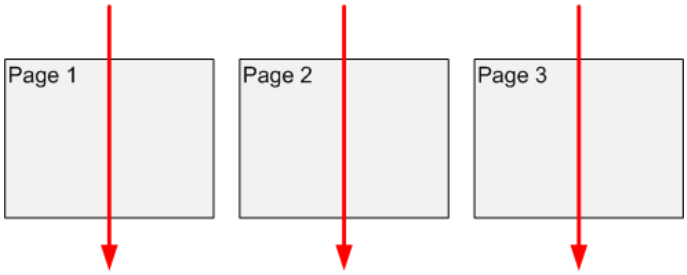


Figure 51: The Logical Flow through Business Object Pages

Exercise 5.2.1 The Logical Flow through Business Object Pages

Business Object pages run one at a time and are usually not interconnected like the pages of a Business Process. A Process can select which Business Object pages to use and use them in any order.

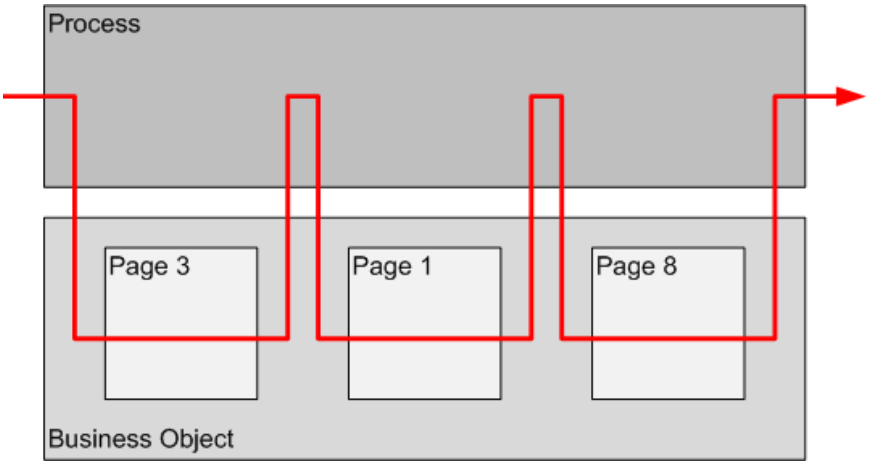


Figure 52: The Logical Flow through a Process and Business Object

Exercise 5.2.2 Notepad Business Object

In this exercise we will take a quick look at a ready-made example of a Business Object. We won't concern ourselves how it was made just yet, we'll just see how it works.

- Navigate to Studio and open the example Notepad Business Object.
    - *Tip: Every Business Object comes with two default pages. We will look at Initialize and Clean Up later.*
- Select the Launch page and then press the Go button.
- Select the Close page and press the Reset button before pressing “Go”.
- Repeat the Launch and Close sequences.
    - *Tip: Press the Reset button every time you want to run a page in Object Studio.*
- Close Object Studio without saving any changes.

5.3. Action Stage

So we have a basic Business Object that can launch and close the Notepad application. However, knowing that a Business Object can only be used via a Process, we will need to return to Process Studio to employ our Business Object.

An Action stage allows us to use a Business Object in a Process.

Figure 53: Action Toolbar Button

Figure 54: Action Stage

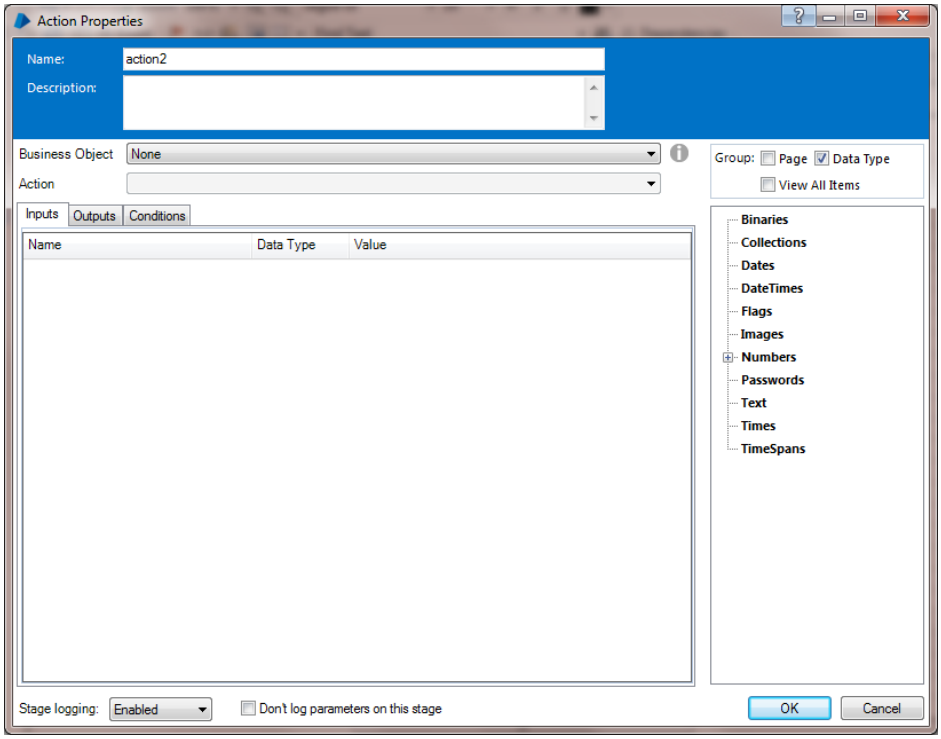


Figure 55: Action Properties

An Action stage is similar to a Page Reference in that it takes the flow of the Process down to another page. But instead of it being a page of the same diagram, it is the page of a Business Object, i.e., an entirely separate diagram.

Exercise 5.3.1 Action Stages

In this exercise we will return to Process Studio to create a Process that makes use of the Notepad Business Object.

- Create a new Process named **Notepad Exercises**.
- Add a new Action stage and open the properties.
- Select **Notepad** from the Business Object drop-down list.
- Notice how the Action drop-down list changes when you select the Business Object.
- Notice how the Action list contains the names of the pages we saw in Object Studio.
- Select **Launch** from the Action list.
- Add a second Action stage and set its properties to Close Notepad.
- Link the stages together so the Process will launch and then close Notepad.
- Save the Process (NB: you always have to save a Process before running for the first time).
- Run the Process to see that Notepad should open and close.

Exercise 5.3.2 Stepping into an Action

As mentioned earlier, it is possible to step from Process Studio into Object Studio, and we will see how that works in this exercise.

- Reset the **Notepad Exercise** Process.

- Step through the Process.
    - *Tip: F11 is the Step shortcut key and F10 is Step Over.*
- Notice how Object Studio is opened when you step into each Action stage.
    - *Tip: Because Process Studio and Object Studio look so similar, it may not be obvious at first that a new window has opened. You may want to resize or move a window to confirm this.*
- Keep stepping and see how Object Studio is automatically closed and you are returned to Process Studio.
- Once you have gone all the way through the Process, press the **Reset** button.
- Step through again, but this time use **Step Over** to see that the Action is executed in one step and Object Studio is not opened.
- Think back to the Process Reference stage and consider the similarities of stepping into or over an Action stage.

5.4. Inputs and Outputs

Earlier we looked at using inputs and outputs on the pages of Processes and sub-Processes. In fact, the most common usage for inputs and outputs is on the pages of Business Objects.

Just as a Business Object provides a Process with the functionality it needs to manipulate an application, it is the inputs and outputs that move data back and forth between a Process and a Business Object.

Exercise 5.4.1 Action Inputs

In this exercise we will look at how a Process uses an input to provide a Business object with information.

- On the **Notepad Exercise** Process, make a gap between the "open" and "close" Actions.
- Put a new Action stage into the gap.
- Edit the Action stage properties and select the **Write** action.
- Notice that a new row has appeared on the inputs tab. This row indicates that the Business Object is **requesting** information in order to execute the action.
    - *Tip: Imagine the input as the Business Object asking the Process, "What note do you want me to write?"*
- Enter the expression **"Hello World"** (remember the speech marks!) to supply the input value.
    - *Tip: Imagine the Process telling the Business Object "write this note."*
- Run the Process to see the Write stage in operation.
- Save and close Process Studio.

Key Point

- Data are transferred between a Process and a Business Object via input and output parameters.

5.5. The Process Layer

We have seen that a Process can be spread across a number of pages arranged in a hierarchy. So far we have assumed that the Process layer is comprised of a single Processes but this is not always the case.

We know that a Process can be called by another Process, much in the same way that a Process makes reference to a page. The Process layer can sometimes consist of a series of sub-Processes marshaled by a "master" Process.

Exercise 5.5.1 Stepping through Multiple Levels

In this exercise we will create a "master" Process to employ Notepad Exercises as a sub-Process.

- Create a new Process named **Notepad Master**.
- Add 2 Process Reference stages and link them between the Start and End stages.
- Edit each Process stage so that they will both call the **Notepad Exercises** as a sub-Process.
- Save the Process (NB: you always have to save a Process before running for the first time).
- Step (F11) through the Process.
- Notice how a second Process Studio window is opened when you step into the Process Reference stage, and then when you step into an Action stage, Object Studio is opened too.

  > *Tip: Because Process Studio and Object Studio look so similar, it may not be obvious at first that a new window has opened. You may want to resize or move a window to confirm this.*

- Keep stepping and see how the "child" windows are automatically closed and you are eventually returned to the original Process Studio.
- Experiment to see the effect of **Step Over** and **Step Out**.
- Save and close Process Studio.

5.6. Review

- Application logic is not contained in a Process.
- A Business Object is used to manipulate an application.
- Object Studio is used to configure Business Objects.
- Object Studio is much like Process Studio.
- A key difference is that a Business Object is not exposed to Control Room; it can only be used via a Process.
- A Business Object encapsulates the functionality of an application and exposes it to a Process.
- A Process uses an Action stage to employ a Business Object page.
- A Process flows through a hierarchy of pages.
- A Business Object flows through pages one at a time.
- Stepping into an Action stage will open Object Studio and stepping out will close it.
- Stepping into a Process Reference stage will open another Process Studio window in a similar way.

Blue Prism Keywords

Business Object, Object Studio, Action.

6. Object Studio

As we have seen, Business Objects encapsulate the functionality of applications. A Business Object is only intended to be used via a Process and cannot be exposed to Control Room.

6.1. Creating a Business Object

We have briefly looked at Object Studio and seen that it is like Process Studio but with some additional features. In this section, we will look at how Object Studio is used to configure Business Objects.

Exercise 6.1.1 New Business Object

In this exercise, we will create a Business Object that can launch and log into the Order System training application.

- Navigate to Studio and create a new Business Object named Order System. You could use any name but it makes sense to go with the name of the application.
- Once you are into Object Studio, you will have three pages - Initialize, Clean Up, and Action 1.
    - *Tip: Initialize and Clean Up are default pages present in every Business Object and we will look at their purpose later on.*

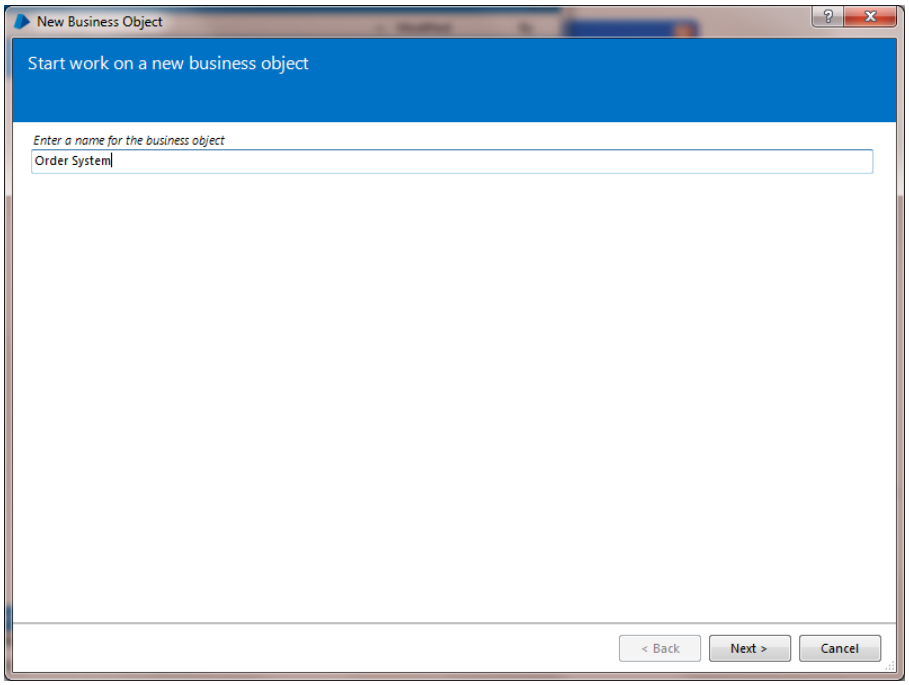


Figure 56: New Business Object Wizard

6.2. Application Modeler

We know that the purpose of a Business Object is to integrate with an application so as to provide a Process with the means to use the application. In order to create this integration, we must first "teach" the Business Object about the application by creating an ***Application Model***.

Exercise 6.2.1 Application Modeler Wizard

Object Studio has a feature named Application Modeler that enables us to create a logical representation of an application. In this exercise, we will see how to set up Application Modeler with the basic details of the target application, Order System.

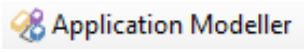


Figure 57: Application Modeler Toolbar Button

- Open Application Modeler by pressing the toolbar button. The first time you open Application Modeler, a dialog box will appear asking for initial setup details.
- Follow the steps through using these details:
- Select Define a new Application Model with the name Order System.
- Windows application type.
- Application will be launched from an executable file C:\BluePrism\Training\Applications\Windows\TrainingOrderSystem.exe.
- Step through the remaining steps in the Wizard by accepting the default values.

  * *Tip: We will look at advanced topics like command line parameters and invasive techniques later.*

- Once the steps are complete, you will be presented with the Application Modeler screen.

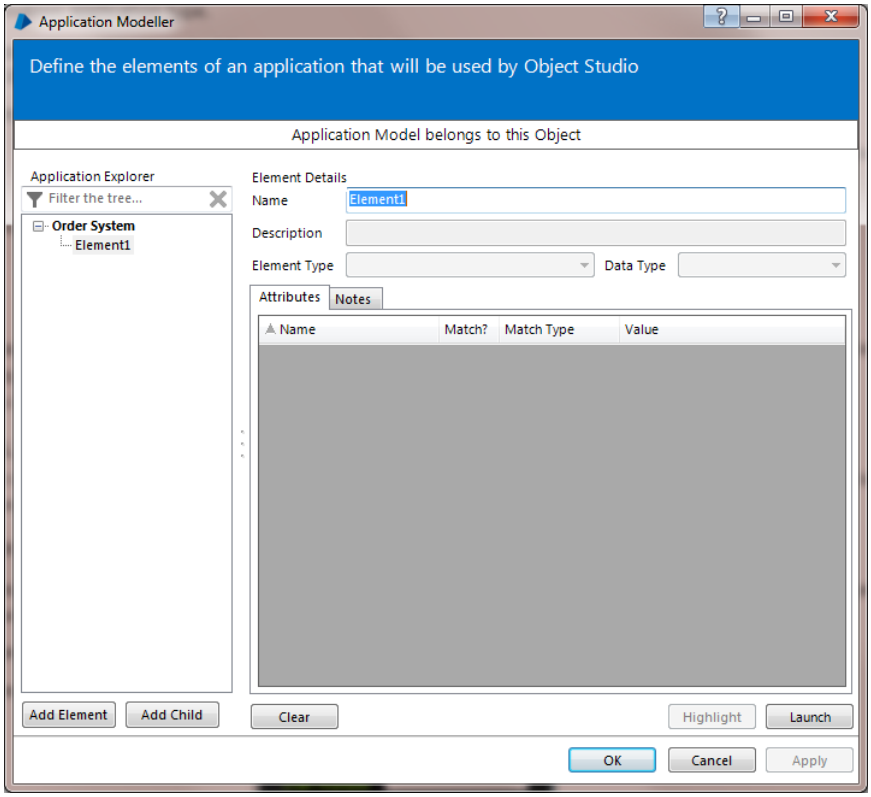


Figure 58: Application Modeler

The purpose of Application Modeler is to capture details of elements of the application user interface, things like fields, buttons, and windows. Once we have defined the elements we want to use, we can set about creating a diagram to interact with them.

Exercise 6.2.2 Identify Elements

In order to select the application elements we want to use, we first have to launch the application. In this exercise, we will launch Order System and look at how Application Modeler finds elements on the application screen.

- "OK" the Application Modeler to return to Object Studio.
- Resize the Object Studio to about two-thirds the size of the desktop. Although not essential, having Object Studio slightly smaller than the desktop will help when we have multiple windows open.
- Open Application Modeler again. Notice that this time the "initial setup" wizard does not appear.
- Press the **Launch** button and wait for the Order System Sign In window to appear.

Figure 59: Order System

- Press the **Identify** button (NB: the same button is used for "Launch" and "Identify", and changes name once the application is launched).
- Move the mouse over the Order System window. See that the individual elements of the window are being highlighted in red. Application Modeler is investigating the structure of the Order System window as the mouse moves over it. This is known as **spying** or **spy mode**.

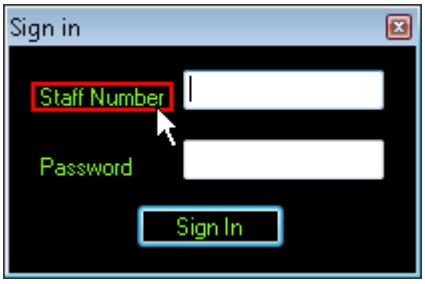


Figure 60: Spying Order System

Before learning to identify elements, we will first cancel this spying operation. Hold down the CTRL key and right-click anywhere on the Order System application to cancel the highlighting and return to Application Modeler.

6.3. Spying Elements

Application Modeler spies elements as the mouse is moved over the application window. Details of an element can be captured by holding the ***CTRL*** key and clicking the ***left*** mouse button.

Exercise 6.3.1 Spying Elements

In this exercise, we will spy part of Order System and look at the details Application Manager captures.

- Press the **Identify Element** button again and spy the Order System window. Move the mouse towards the top of the window so that the red highlighting surrounds the whole window.

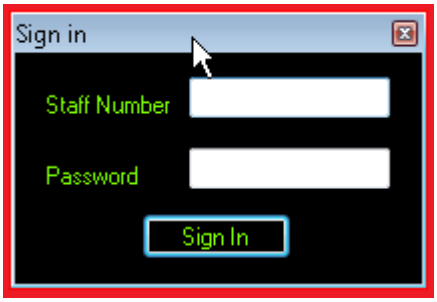


Figure 61: Spying the Order System Window

- Hold down “CTRL” and left-click on an open area of the window. See that Application Modeler is now populated with details of the window.

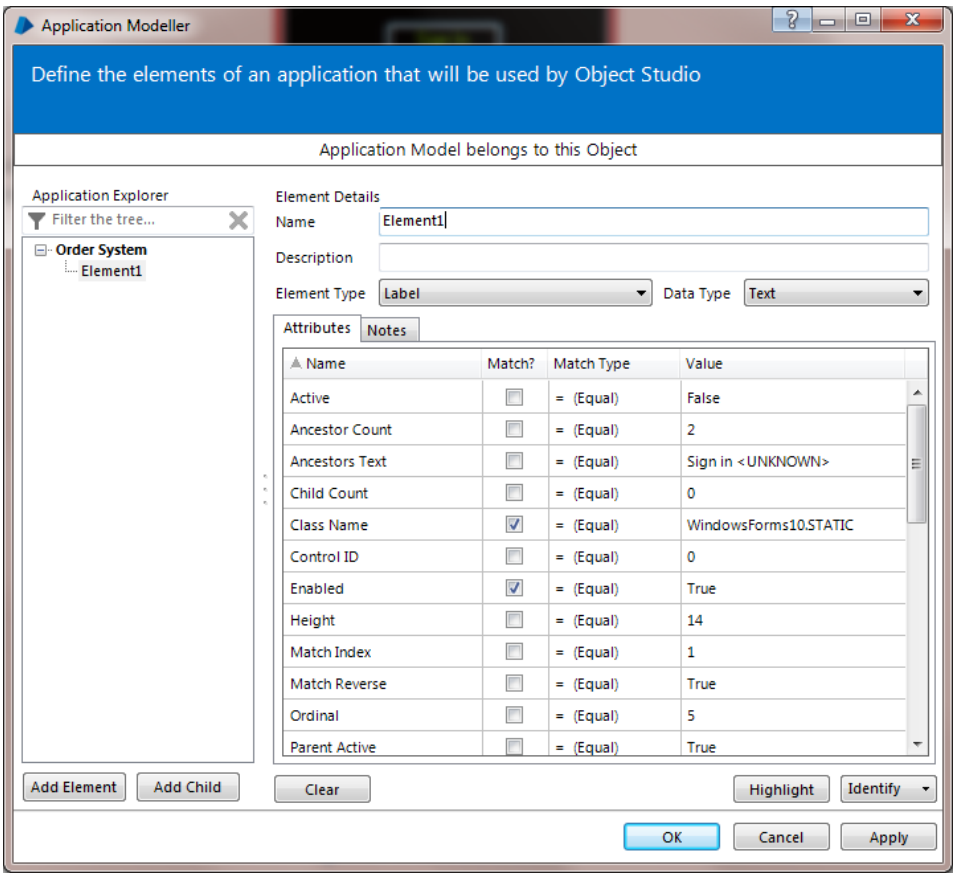


Figure 62: Window Element Details

Key Points

- Keep “CTRL” down until you’ve clicked the mouse.
- To exit spy mode, it’s CTRL+***right***-click.

6.4. Attributes

Application details are drawn from the application as a list of ***attributes***. These attributes are not data created by Blue Prism; attributes are characteristics of the element provided to Blue Prism by the application.

Once captured, this information can enable a Business Object to "remember" an element, and it does this by using a selection of attributes as the "fingerprint" of an element. This fingerprint is then used to identify and interact with the element from the Object Studio diagram.

Exercise 6.4.1 The Attribute List

- Take a moment to scroll down the list of attributes.
- Although some attributes may seem mysterious, you may recognize others, like "Window Text".

Key Points

- Attributes do not come from Blue Prism; the application provides these data to Application Modeler.
- Application Modeler forms these attributes into elements.
- Object Studio uses the elements to manipulate the application.

6.5. Attribute Selection

Application Modeler retrieves all available attributes from the application but typically only some of these are used to form the element fingerprint. The selected attributes are shown in the ***Match*** column, and can be checked or un-checked as required. The initial selection is a default choice made when the element is first spied, and although this first choice is often correct, it may well require manual adjustment to make a unique fingerprint.

There are many types of element (buttons, text boxes, etc.) used by many different kinds of applications (thick-client, thin-client, mainframe, etc.), each with a different set of attributes. Blue Prism cannot anticipate the construction of every kind of application it will ever encounter, and as such it is important to realize the initial selection of attributes made by Application Modeler is a ***suggestion*** and not an instruction.

Application Modeler cannot know for certain which combination of attributes will form a unique element fingerprint, but can posit a suggestion based on the type of element and type of application. The responsibility for finding the unique combination ultimately rests with you, the creator of the Business Object.

Key Points

- Application Modeler suggests a selection of attributes when the element is spied.
- The selection must uniquely identify the element and may require manual alteration.

In order to test whether Application Modeler has "remembered" the element correctly and that the element fingerprint is unique, we can ask it to try and "find" the element again.

Exercise 6.5.1 Highlighting Elements

In this exercise, we'll check if the current selection of attributes enables Application Modeler with to "find" the element properly.

- Press **Highlight Element** to instruct Application Modeler to find the element.
- Notice how the Order System window is highlighted in red. This means Application Modeler can "see" the element and that the attribute selection is acceptable.
- Highlight the element again - you can do this as often as you like.
- Rename the element from "Element 1" to "Log In Window" and press "OK" to close Application Modeler (NB: you do not have to press "Apply" before "OK").

We now have the beginnings of our model of Order System. Granted, the model only contains one element but it is enough for us to make a start in Object Studio. We will return to Application Modeler when we need more elements.

Key Point

- The application model should not be created in one go but built up gradually. After spying a few elements, return to Object Studio to check their use from the diagram.

6.6. Launch

We have told Application Modeler how to start the application by giving it the path to the executable (.exe) file, and we have defined the details of the Log In window in our model so that we can use it in Object Studio. We now have enough information about Order System to create a page to launch it.

Exercise 6.6.1 Adding a New Page

In this exercise, we'll make a page in the ***Order System*** Business Object that can launch the application.

- Right-click on the **Action 1** tab and rename it to **Launch**.
- Notice how the menu is similar the one in Process Studio - new pages are added in a similar way.

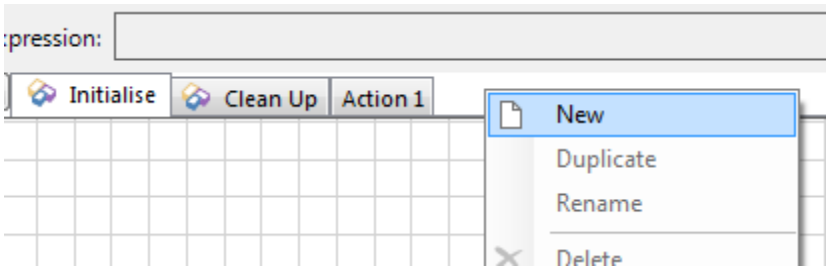


Figure 63: Object Studio Page Menu

Best Practice

- Be descriptive when naming a Business Object page – use a ***verb*** to communicate the ***purpose*** of the page. For example, "***Open*** Account History" or "***Close*** Account History" is much better than just "Main Menu".
- Open the Page Information box and provide a description. As you become more proficient you should also document your pre- and post-conditions.

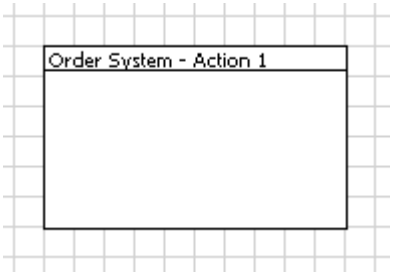


Figure 64: Page Information Stage

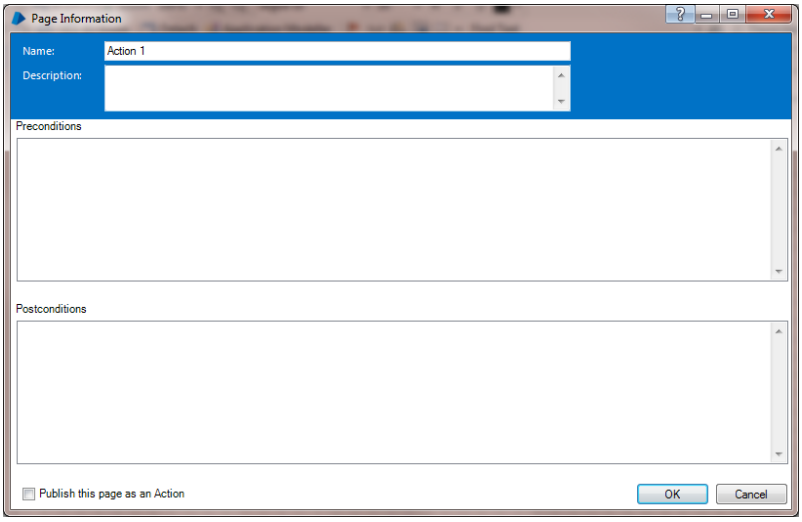


Figure 65: Page Information Properties

Exercise 6.6.2 Using a Navigate Stage

- On your Launch page, add a Navigate stage and link it between the Start and End stages.

Figure 66: Navigate Toolbar Button

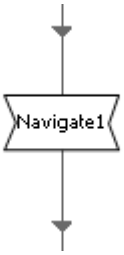
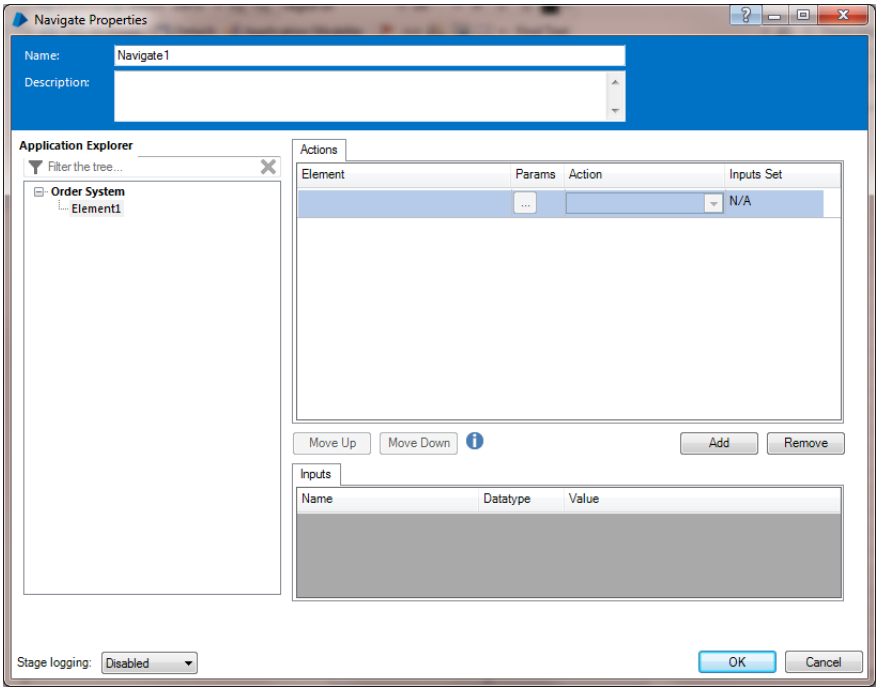


Figure 67: Navigate Stage

- Open the Navigate properties and see that the element list from Application Modeler is displayed on the left.



- Drag the very top (or root) element from the left-hand list into the Element column of the Actions list.
- Select **Launch** in the Action drop-down list. Name the Navigate stage **Launch** to match.
- Press "OK" to close the properties window and then link the Navigate stage between the start and End stages.
- The Navigate stage is now ready to launch the Order System application.

Key Points

- Just as in Process Studio, a Business Object must exist (i.e., be saved) in the Blue Prism database before it can run.
- You don't need to save every time, but you need to save before the very first run.

Exercise 6.6.3 Launching Order System

- Manually close Order System and save the Business Object.
- Run the page and see that Order System has been launched and the Log In window is displayed.
- Close Order System yourself and then run the page again.
- Is there anything we could check for to be sure the Launch action has worked correctly?
- It may seem that Order System takes the same amount of time to start up every time. However, in a real-life situation where we would have no control over the response time of an application, and it is not unthinkable that performance could vary.
- The Business Object needs to have patience be able to wait for a fickle application.

6.7. Wait

The Wait stage used to enable a Business Object to pause and wait for an application element. This allows a Business Object to deal with potentially erratic application performance. There are different element conditions to wait for but the most common is simply to wait for the element to exist.

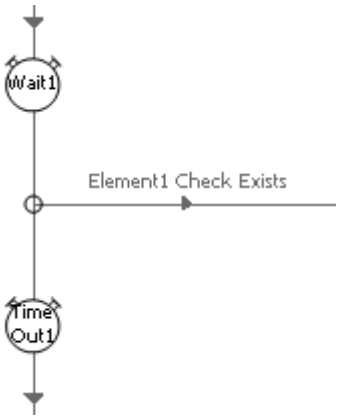


Figure 68: The Wait Stage

Exercise 6.7.1 Waiting for Order System

- Run the page and see that Order System has been launched and the Log In window is displayed.
- Add a Wait stage and connect the Navigate stage to the top of the Wait. The Wait properties window has three sections, a list of elements, a list of conditions, and a list of Data Items.

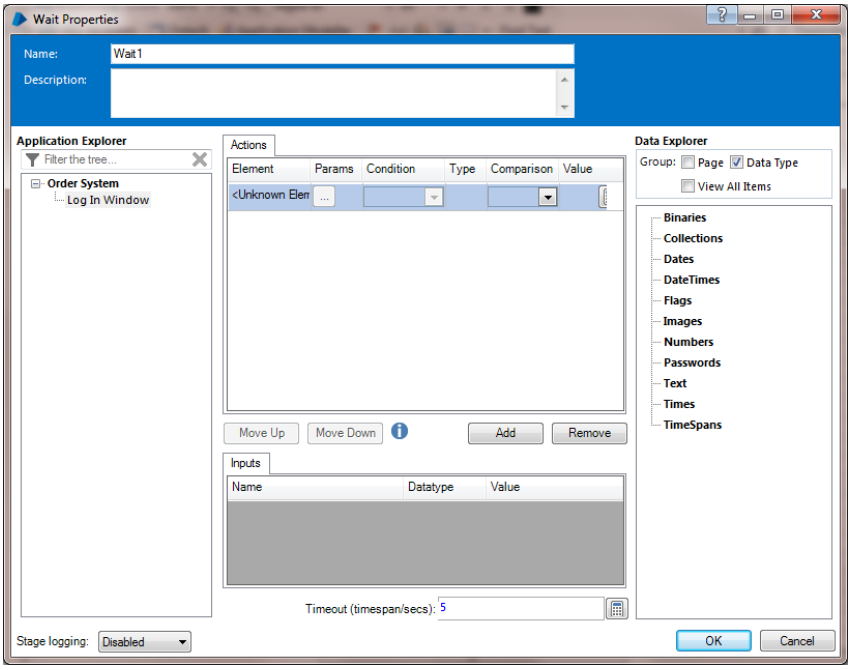


Figure 69: Incomplete Wait Properties

- Drag the **Log In Window** element from the left-hand Application Explorer list and drop it into the Element column of the Conditions list.
- Select the “Check Exists” option from the Condition column and leave the Comparison and Value fields with their default values of “Equals” and “True”.

- Name the stage as you see fit (the name of the element, e.g., **Log In Window** can be used) and if necessary resize it to make things more readable.
- Press "OK" to close the properties window.

We now have a Wait stage capable of waiting for the Log In window to exist, but how should we position it on the page? Note that a new circular node has appeared in the middle of the Wait stage.

Exercise 6.7.2 Wait Conditions

- Link up the Wait stage as shown in the diagram. Notice that you can drag your Wait stage around to make it horizontal if you prefer to have the links vertical.
- Close the Order System application, set the speed to maximum, press "Reset", and then press "Go".

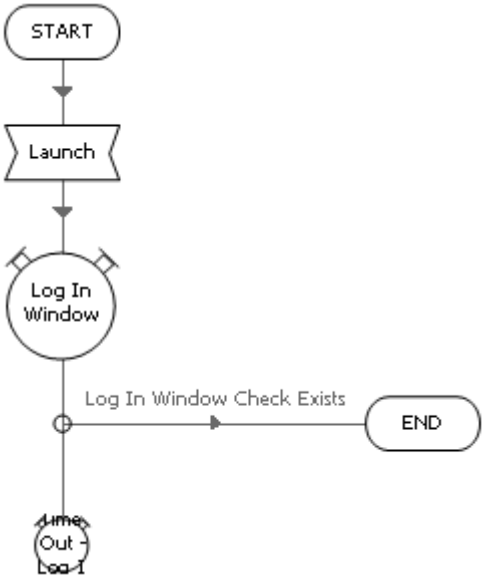


Figure 70: The Check Exists Wait Condition

Although this configuration may seem trivial, it is central to the function of Object Studio and, without the Wait stage, we would have great difficulty creating a reliable application interface. Using the Wait stage, we give the Launch action a degree of flexibility in dealing with Order System.

Key Point

- The Wait stage is absolutely vital to the creation of Business Objects.

Now let us imagine that Order System will not launch for some reason. Rather than let your Business Object wait for something that will never arrive, we will enable it to stop waiting.

Exercise 6.7.3 Wait Stage Timeout

- Open the Wait stage properties and look at the Timeout field at the bottom of the window. Change the 5 to **0.1** and press "OK".
- Add an Exception and link to it from the Timeout end of the Wait stage as shown in the diagram.
- Double click on the Exception and enter the Name and Exception Type as **System Exception**. For Exception Detail, enter the expression**"Log in screen has not appeared"** (remember to include the quotation marks). NB: we shall cover Exceptions in more detail later in the training.

- Manually close the Order System application, press “Reset”, and then press “Go”.
- Notice that the action flows to the Exception and does not take the original “Check Exists” path.

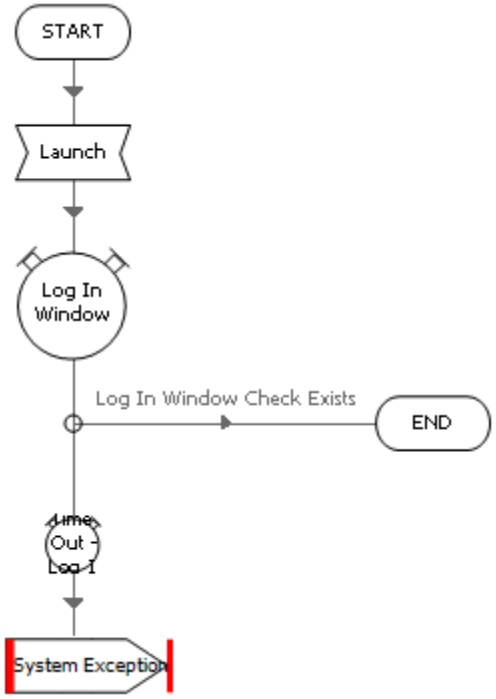


Figure 71: Wait Stage Timeout

A Wait stage has a timeout value which defines the maximum time it is prepared to wait for a condition. If the condition does not occur within the given time, the Wait stage will direct the flow via the end point.

In the above exercise, we simulated a sluggish Order System by making the time out ungenerously small, giving the wait no chance to find the condition.

We will look at how to manage undesirable outcomes like a timeout later on in the course.

Exercise 6.7.4 Multiple Wait Conditions

- Change the timeout value back to **5**.
- Go back and look at the properties of the Wait stage. Use the **Add** button to create a second wait condition and for now just populate it in the same way as you did the first row.
- “OK” the properties window and look at the Wait stage, noticing how a second node has appeared.
- Add a second End stage and link from the new wait condition to it as shown in the diagram. NB: this is an example only for illustration purposes; in reality **the second identical wait condition is redundant**.

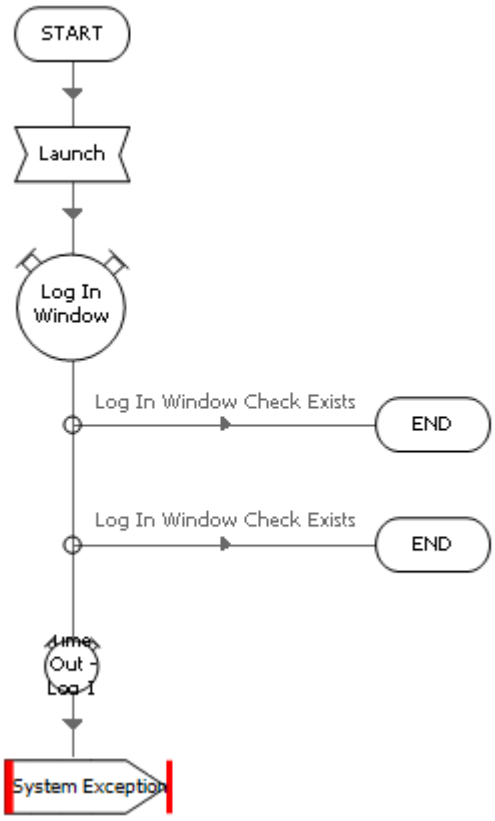


Figure 72: Two Wait Conditions

A Wait stage can be stretched out to accommodate lots of different conditions, providing separate paths to take according to each condition.

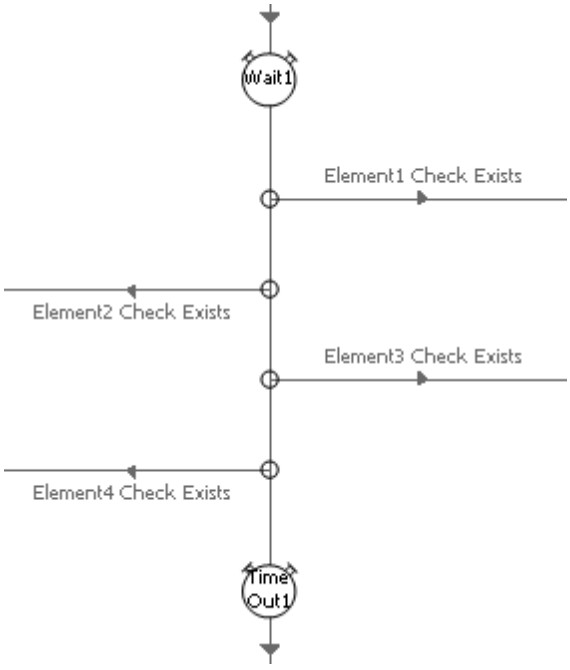


Figure 73: Multiple Wait Conditions

When multiple conditions are used, the Wait stage will check each condition in top-down order and opt for the first "true" condition it finds. If none are true by the time the specified time out period elapses, the end path will be taken.

Exercise 6.7.5 Using a Timeout as a Pause

Timeouts are not necessarily a negative event and can be useful for introducing pauses into a Business Object.

- Add a new Wait stage to your diagram and give it a one-second timeout.
- Link the Wait stage into your diagram (it doesn't matter where) and re-run the page.
- Notice how the page pauses on the Wait stage before continuing.

> *Tip: Run the page at maximum speed to see the effect more clearly.*

By using a Wait stage without any condition, we can use the timeout as a positive outcome that simply delays the flow slightly. Pauses like this are often used to afford some "breathing space" to the target application, particularly if the application is better suited to a more "human-like" operating tempo.

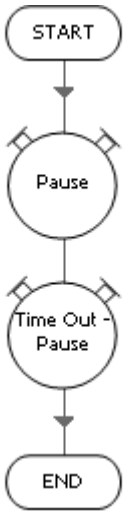


Figure 74: Timeout Used as a Pause

Key Points

- A Wait stage can wait for a single condition or several conditions.
- A Wait stage will take the path of the first True condition or, if the timeout elapses, the action can proceed to carry out further actions or as noted previously to throw an Exception as something has not happened as expected.
- An exception at the Timeout stage will alert the Process to an error that has happened.
- A Wait stage can exist without any condition and the timeout can be used as a pause.

6.8. Timeouts

In the Wait stage used after launching Order System, we kept the default five-second timeout value. In most practical situations, it does not matter if you set this value much higher (e.g., to 30 seconds), because Blue Prism responds immediately once the expected condition is met – the timeout is merely the ***maximum*** amount of time that you are prepared to wait.

On the other hand, if you were to set the timeout to an unsuitably small value (e.g., 1 second), then you may experience unpredictable results. It is therefore better to overestimate rather than underestimate the length of time you need.

Best Practice

- Be generous with timeouts - overestimate rather than underestimate.
- Recall the timeout of five seconds which you applied to the wait stage above. In most practical situations, it does not matter if you set this value much higher (e.g., to 30 seconds), because Blue Prism responds immediately once the expected condition is met – the timeout is merely the maximum amount of time that you are prepared to wait.
- On the other hand, if you were to set the timeout to an unsuitably small value (such as 1 second), then you would experience unpredictable results. It is therefore better to overestimate rather than underestimate the length of time you need.
- Use Data Items to store timeout values. You should use Data Items to store timeout values so that you can modify multiple Wait stages with one easy change to a Data Item.
- Business Objects should have global timeout period data items that are used in all wait stages in the object.
- Using global data items to store timeout periods makes it easy to re-configure how long an application should wait for different types of system activity.

Exercise 6.8.1 Global Timeout Data Items

- Visit the Initialize page of your Business Object. By convention, this is where global Data Items tend to be created. Create some global number Data Items like those below.

Name	Initial Value
Global Timeout – S	5
Global Timeout – M	10
Global Timeout – L	30
Global Timeout – XL	60
Global Timeout – XS	1

The initial values represent a number of seconds and we will use these Data Items instead of a fixed value when creating Wait stages.

- Modify your Wait stage on the Launch page to use an appropriate timeout Data Item.

Typically a larger timeout would be used when waiting for a system to launch, navigate to a new page, update a record, or anything that may take considerable time to execute. Smaller values are more likely to be used for the more rapid movements of an application, such as waiting for a field to become enabled or a validation message to appear.

Exercise 6.8.2 Throttling

As mentioned above, a Wait stage without any wait condition can be used to create pause, and a Data Item can be used to control the length of the timeout.

Using this arrangement it is possible to create a “throttle” by placing Wait stages controlled by a global Data Item at key positions in a Business Object. Typically this could simply be done by starting every page with a “throttled” Wait stage.

The “Throttle” global Data Item can be used to manipulate the pace of a Business Object. To test its effect on a target application, a Business Object can be tested with a “tight” throttle that restricts it to the same pace as a human user. As testing progresses, the throttle can be released until an ideal speed is found. If the throttle is set to zero, the timeout will be instant and there will be no pause.

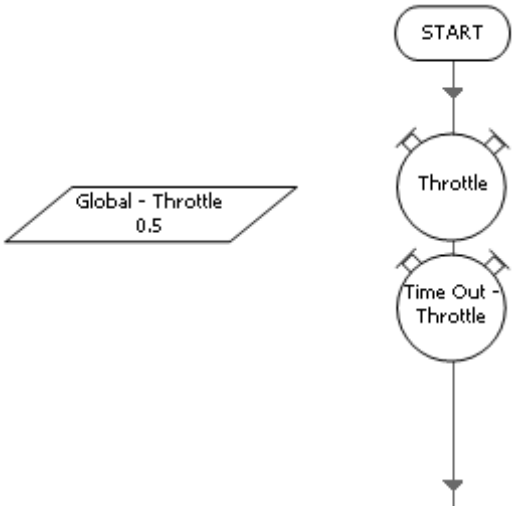


Figure 75: Throttling

6.9. Terminate

A Navigate stage can also be used to terminate or kill an application. As the name suggests, a termination is rather a severe way to close an application and often the application requires that the user (i.e., Blue Prism) should log out properly beforehand. Fortunately, Order System is robust enough to cope with being terminated.

Exercise 6.9.1 Terminating Order System

In this exercise, we’ll improve our Business Object so it can close down Order System.

- Add a new page named **Exit** to the Order System Business Object.
- Go back to the Launch page and copy the Navigate stage.
- Return to the new page and paste in between the Start and End.
- Edit the properties of the new Navigate stage so that it terminates Order System instead of launching it.

> *Tip: You cannot use a wait to check if Order System has closed because a wait can only work when the target application is running. Once the Navigate stage has terminated, the application any subsequent attempt to use an element from Application Modeler will cause an error.*

Key Point

- Using Terminate is similar to killing an application using Task Manager.
- Although some applications are not adversely affected by being closed in this way, it is important to remember that some are and that a gentler, less severe Close method is required.

6.10. Write

Another Object Studio stage, named the Write stage, is used to send values to an application. The result of an expression is used to populate an application element defined in Application Modeler.

Exercise 6.10.1 Spying New Elements

- Open Application Modeler and press Launch Application if Order System isn't running.
- Create a new element in the left-hand list and then spy the **Staff Number** field.
    - *Tip: The Add Element and Add Child buttons will both add a new item to the list. They can be used to arrange elements in a tree structure.*
    - *Tip: List items can also be dragged to new positions and the mouse menu also has options to cut, copy, paste, and delete.*

Key Point

- Using Terminate is similar to killing an application using Task Manager.
- You can organize your elements as you wish, although usually it makes sense to use a hierarchy that reflects the structure of the application.

- Press **Highlight Element** to check everything is OK. What do you think is happening?

Application Modeler has found two elements and cannot distinguish one from the other. Each element must be uniquely identified by its attribute selection in order for Application Modeler to find it, but in this case a conflict has been found.

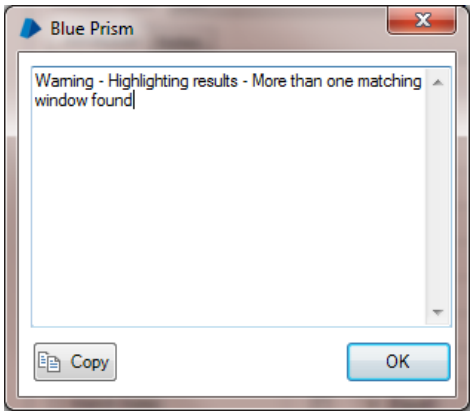


Figure 76: Highlight Element Warning

This is a deliberate error designed to illustrate what can happen in the development of a real-life Business Object, i.e., the selection of attributes suggested by Application Modeler does not uniquely identify the element and you need to make some adjustments.

Key Points

- Application Modeler suggests a selection of attributes when the element is spied.
- The selection must uniquely identify the element and may require manual alteration.

Exercise 6.10.2 Finding Unique Attributes (Part I)

- Look at the Attributes list to see which attributes are being used for Staff Number.
- Find the **Ordinal** attribute and make a mental note of its value.

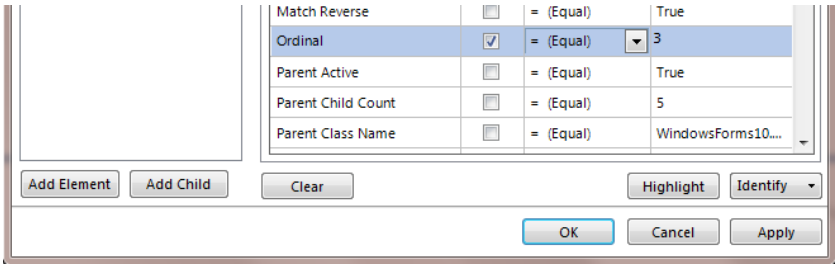


Figure 77: Ordinal Attribute

- Check the **Match** column (ignore the **Match Type** column for now).
- Press "Highlight Element" to check that the field has been uniquely identified.
- Spy the Password field and adjust the selection of attributes in the same way. Compare the Ordinal value of the Password element with that of the Staff Number element.

The attribute selections initially suggested by Application Modeler did not work initially because the "fingerprints" of both elements were identical. By introducing Ordinal into the selection, we have made each fingerprint unique and enabled Application Modeler to "know" the difference between Staff Number and Password.

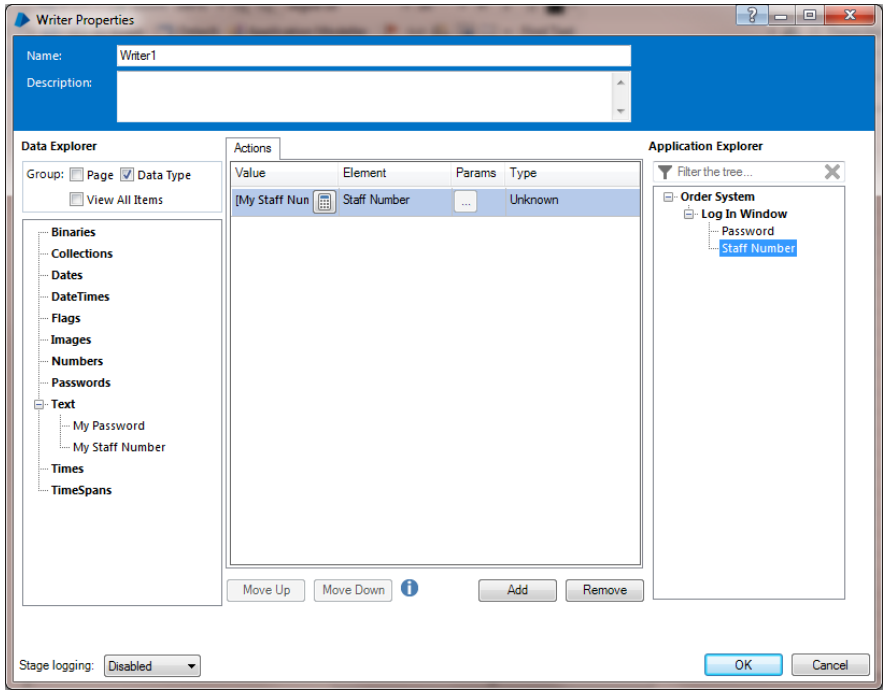
Key Points

- Finding a winning combination of attributes is largely down to trial and error but as your experience grows, you will become more adept at making these adjustments.
- Although Ordinal is a fairly common attribute and can, as in this example, make all the difference, it is not available in all application types.

Exercise 6.10.3 Using the Write Stage

- Once you have successfully identified the two fields, create a new page named **Log In** and add two Data Items.
- Name one Data Item **My Staff Number**, give it the **Text** data type and initial value "bp".
- Name the other Data Item **My Password**, give it the **Password** data type and the initial value **password**.
- Add a **Write Stage** and open the properties.

As the name suggests, the Write stage is used to put values into elements and the properties form is laid out with a list of Data Items on the left and the elements on the right. The middle section is where we indicate which values want we want to go into which elements.



- Find **My Staff Number** in the left-hand list of Data Items and drag it into the Value column.
- Find **Staff Number** in the right-hand list of elements and drag it into the Elements column.
- Name the stage **Input User Credentials** and click "OK".
- Link the up the Write stage and run. See that the value of the **My Staff Number** Data Item has been written into the field identified by the Staff Number element.

Exercise 6.10.4 Finding Unique Attributes (Part II)

So far, we've differentiated between Staff Number and Password and have spied both elements successfully. We also have a Write stage that has passed the value of a Data Item into the Staff Number field.

- Reset and run the Log In page again. You should see an error message like this:

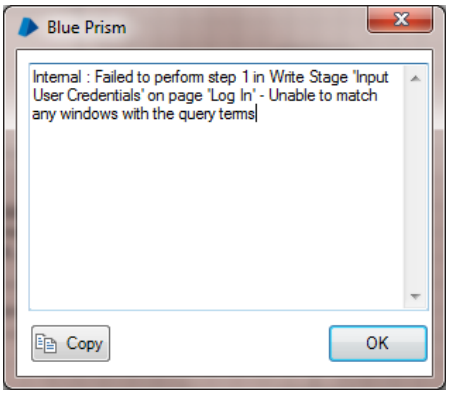


Figure 78: Unable to Match with Query Term

- The "unable to match with query terms" message is Application Modeler's way of telling you it cannot find the element. Essentially it is saying, "I can't see the Staff Number element anymore."

Again, this is a deliberate error designed to illustrate an important point. Elements change state as the application is being used, and sometimes that change in state is enough to disrupt Application Modeler and invalidate our attribute selection.

- We know Staff Number used to work, but now it doesn't. Think about what has changed with the Staff Number field since we first spied it.

The answer is that when we were in Application Modeler the field was blank, but after using the Write stage it is no longer blank. The content of the field is the only thing that has changed, and so this must be the reason Application Modeler is having difficulty.

- Open Application Modeler and scroll to the bottom of the attributes of Staff Number.
- Notice that the **Window Text** attribute is checked. This means it is included as part of the "fingerprint" of the Staff Number element. Notice also that the Value field is blank.

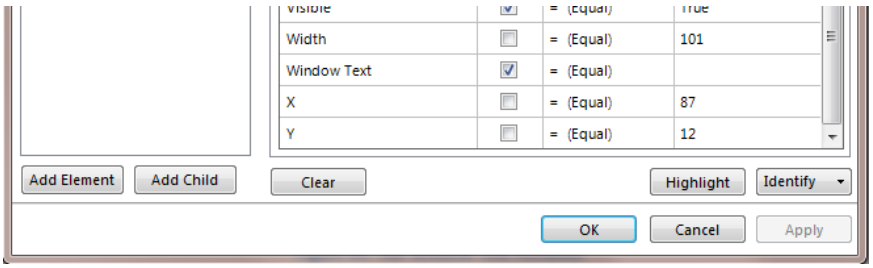


Figure 79: The Window Text Attribute

This effectively means that one of the conditions of the element fingerprint is "Window text = blank", or in other words, "the field must be empty." The Staff Number field used to be empty but it isn't anymore. This is why Application Modeler can no longer "see" the field.

Exercise 6.10.5 Finding Unique Attributes (Part III)

Again, we need to make an adjustment to the choice of attributes we are using for Staff Number. In a similar way to how we added an attribute (Ordinal), we can also remove attributes from the selection.

- Un-check Window Text to exclude the attribute from the Staff Number element definition.
- Press Highlight to test that Application Modeler can "see" Staff Number again.
- Manually remove the text from the Staff Number field and highlight again to confirm Application Modeler can find Staff Number, whether or not it is populated.
- OK Application Modeler and re-run the Log In page.

Key Points

- Using the application may invalidate elements in the application model.
- The "unable to match with query terms" message is Application Modeler saying, "I can't find that thing anymore."
- The selection of attributes can be reduced in order to make an element "correct".

Exercise 6.10.6 Finding Unique Attributes (Part IV)

- Modify the Log In page to write in the value of **My Password** as well.
- Apply the same adjustments to make the page work, regardless of whether the fields are blank.

- *Tip: You can either create a second Write stage or modify the original Write stage by adding an extra row to the list of Actions.*

- *Tip: In case you were wondering why the field values are overwritten and not appended, the Write stage doesn't "type" into fields but "gives" the application a value to put into the field. This is also why the field can be populated even if Order System window is covered by another window.*

6.11. Press

A Navigate stage can also be used to do things like press a button. The options made available in the Navigate properties form depend on the type of element being used.

Exercise 6.11.1 Pressing a Button

- Open Application Modeler and press "Launch Application" if Order System isn't running.
- Create a new element in the left-hand list and then spy the Sign In button.
- Use Highlight Element to check the button can be uniquely identified.

Key Point

- You can use the mouse as normal when spying an application; holding the CTRL key prevents the mouse click from actually pressing the button.

- Add a Navigate stage and link it between the Write stage and the End stage. Edit the navigate properties to press the button. Press "Reset" and "Go".

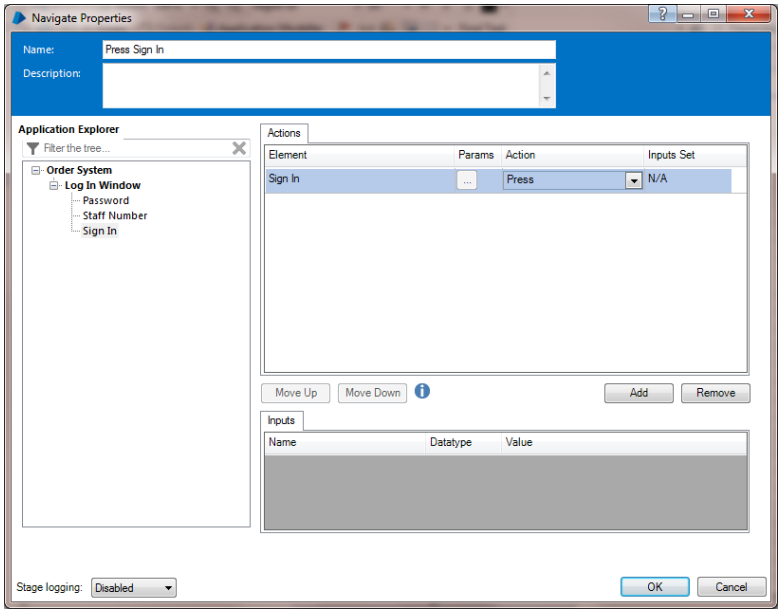


Figure 80: Navigate Properties

Exercise 6.11.2 Waiting for Order System

Think back to the way in which we confirmed that the Launch page was successful and consider if we should add anything to the Log In page. Think of the state Order System should be in once we are logged in - which window will be displayed?

- The answer is that after pressing the button, Order System should display its "Options" window.

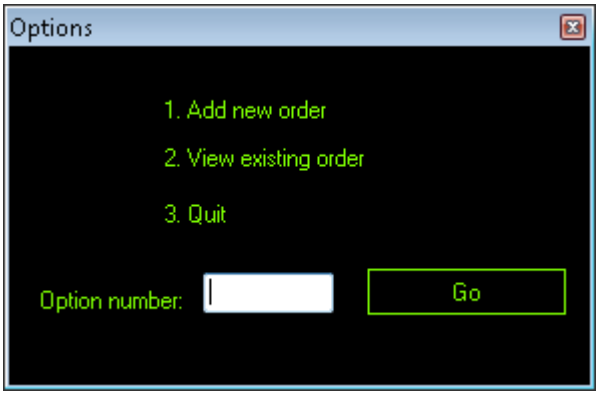
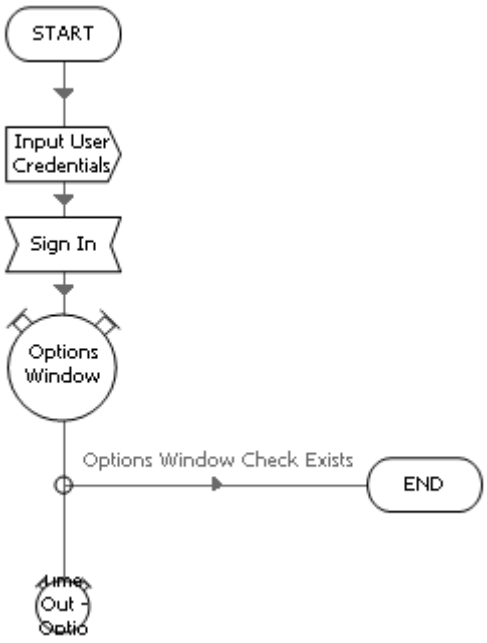


Figure 81: Order System Options Window

- Once you have successfully logged in, spy the Options window of Order System and add a Wait stage after the Navigate stage. Your Log In page should look something like this:



- Close Order System yourself and then run the Launch and Log In pages in succession.
- Repeat until you are happy your Business Object is working correctly.

6.12. Attach and Detach

In some scenarios, the target application may already be running and we want to make use of that instance, rather than try to start another. Similarly, some systems require that the user first logs into a "parent" application that will then start the target application as a "child".

In either case, we want don't want Blue Prism to launch the target application, we want it to work with an instance that is ***already running***.

We have already used a Navigate stage to launch Order System, and a Navigate stage can also be used to attach to or detach from a running application.

Exercise 6.12.1 Detaching from an Application

- Start Order System by running your Launch page.

- Press “Reset” and Press the **Detach** button on the toolbar.

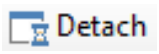


Figure 82: The Detach Button

- Then press “Reset”and run the Log In page. You should see the following error message:

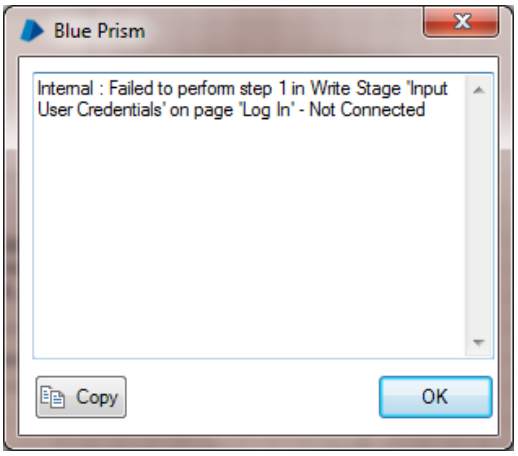


Figure 83: Not Connected

The Detach button has “disconnected” the Business Object from Order System and as a result, the Write stage no longer works. Even though Order System is still there, the Business Object’s communication channel to the application has been severed and the Business Object is no longer “aware” that Order System is running.

Exercise 6.12.2 Attaching to an Application

- Make a new page named **Attach**.
- Go back to the Launch page and copy the Navigate stage.
- Return to the new page and paste in between the Start and End.
- Edit the properties of the new Navigate stage so that it attaches to Order System, instead of launching it. Before pressing “OK”, look at the list of input parameters at the bottom of the screen.

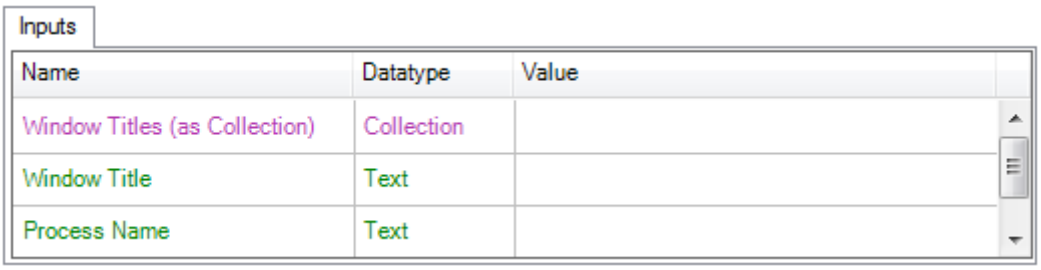


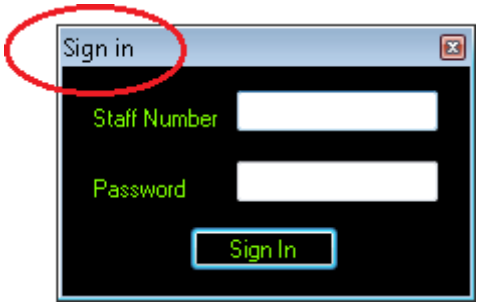
Figure 84: Attach Input Parameters

- Press the Information button to see a description of Attach and its inputs.

Figure 85: The Information Button

- For this exercise, we'll use the **Window Title** parameter to indicate which window on the desktop belongs to Order System.

  * *Tip: Look at the text at the top left of the Order System window.*



- Run the Attach page, switch to the Log In page, then press "Reset" and "Go". You should see that the Log In page now works as before.

The Business Object became detached (because we detached it) and was no longer able to "see" Order System, even though Order System was still running. Attach restored the connection to make the Business Object work again.

Exercise 6.12.3 Attach Inputs

In this exercise we'll make the Attach page identify Order System by its ***Process name***, rather than its window title. By "Process" we are not referring to a Blue Prism diagram, but the name of the ***executable*** that the target system is running.

- Close Order System and run the launch page again.
- Open Windows Task Manager (right-click on the taskbar or press CTRL+SHIFT+ESC).
- Look at the list on the Applications tab of Task Manager and find the "Sign In" item.

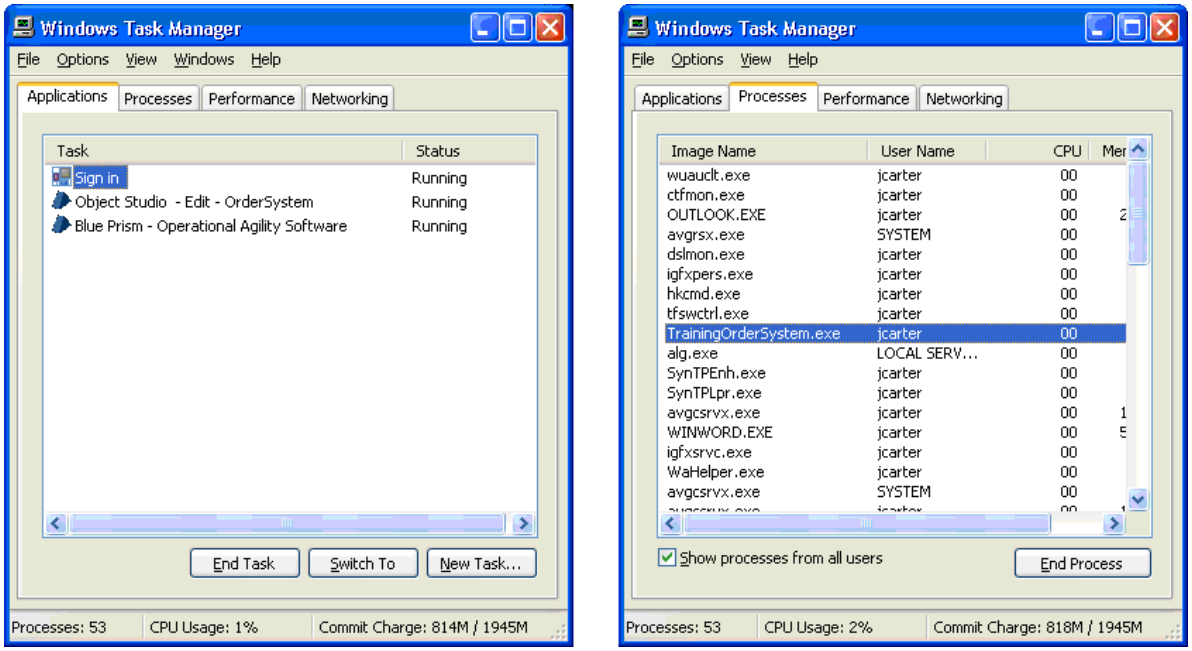


Figure 86: Task Manager

- Right-click on “Sign In” and select the “Go to Process” option. Task Manager will then indicate the name of the Order System executable Process, i.e., **TrainingOrderSystem.exe**.
- Go back to the Attach page.
- Open the Navigate properties and take out the “Window Title” input value.
- Enter the expression **“TrainingOrderSystem”** on the “Process Name” input below.

  * *Tip: Remember the path to the executable file we first used in the Application Modeler wizard.*

  * *Tip: Do not include the “.exe” in the input value.*

Exercise 6.12.4 Application Not Found

Recall that the purpose of Attach is to connect a Business Object to an application that is ***already running***. If for some reason the application is not running, the Attach will fail.

- Close Order System and run the Attach page again.
- The Navigate stage should fail with the following message:

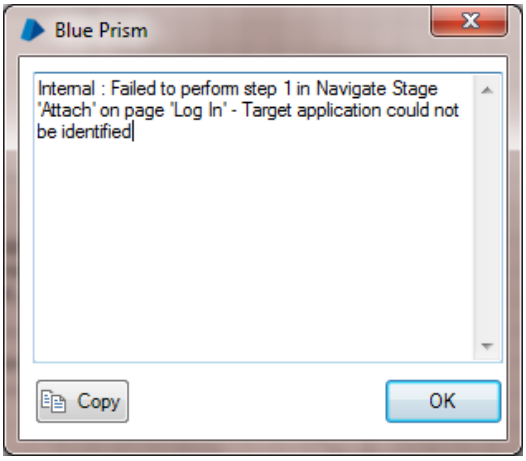


Figure 87: Application Not Identified

The Navigate stage has all it needs to attach but unfortunately the application is not running and so the Business Object cannot find it.

Exercise 6.12.5 Already Connected

Once a Business Object is attached to an application and further attempts to Attach will fail. For Attach to work, the Business Object must not be attached to the application already.

- Run the Launch page again to start Order System.
- Press the Detach button to sever the connection between the Business Object and the application.

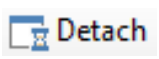


Figure 88: The Detach Button

- Run the Attach page to restore the connection.
- Run the Attach page again to see the following message:

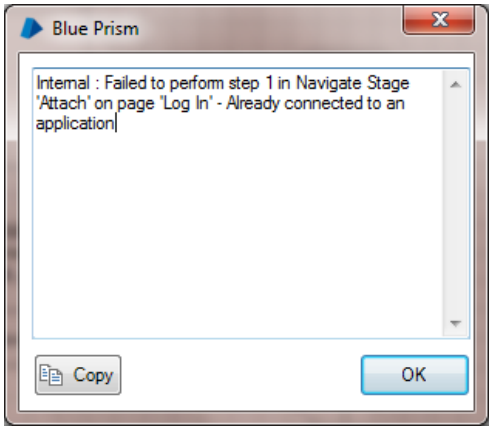


Figure 89: Already Connected Message

Key Points

- Attach is used to connect a Business Object to a running application.
- Detach is used to disconnect a Business Object from an application.
- Attach needs an input value to help it identify the application.
- Attach will fail if the Business Object is already attached.
- When a Business Object launches an application, it is attached automatically.

6.13. Read

As the name suggests, the Read stage is the opposite of the Write stage; it is used to read data from an application. A Write stage takes data (from the result of an expression) and puts it into an Application Modeler element, whereas the Read stage takes data from an element and stores them in a Data Item.

Exercise 6.13.1 Reading from the Application

The most common use of the Read stage is to read data from fields, but we can also read from the application itself. In this example, we will use a Read and a Decision to check if the Business Object is attached to Order System.

- Launch Order System from your Business Object and then look at the Attach page.
- Recall that if we were to run the Attach page, we would get the “already connected” error message (because launching attaches the Business Object automatically).
- Create a Data Item named **Connected** with data type **Flag** and default value **False**.
- Add a Read stage in between the Start and the Navigate (you may need to make some space).

Figure 90: Read Stage Button

Figure 91: Read Stage

- Open the Read stage properties and see that it is similar to the properties of a Write stage, but now the list of elements is on the left and the Data Items are on the right.

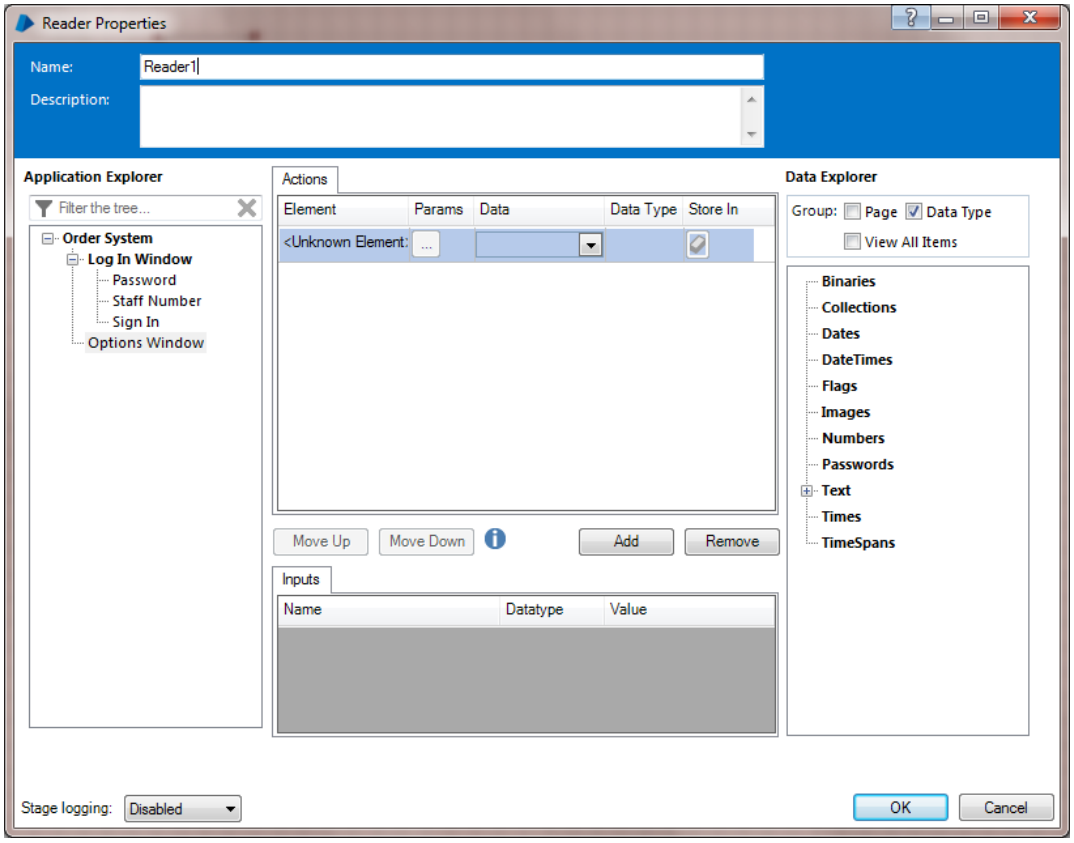


Figure 92: Read Properties

- Drag the very top (or root) element from the left-hand list into the Element field near the center of the screen.
- Select **Is Connected** from the Data drop-down list.
- Drag the Data Item named **Connected** from the right into the Store In field (remember how a Calculation stage has a "Store In" field too).

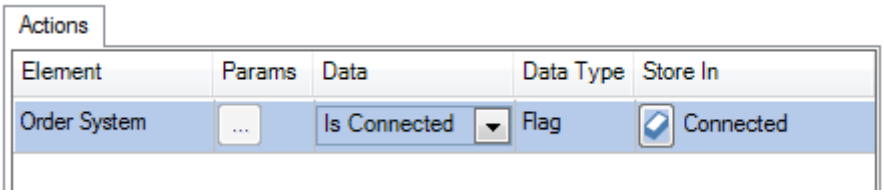


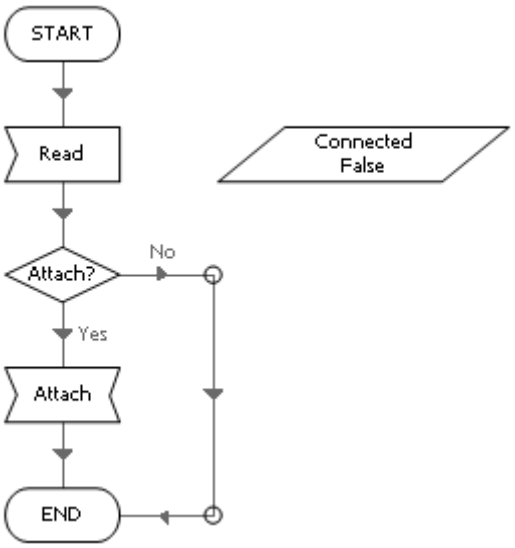
Figure 93: Reading "Is Connected"

- "OK" the properties window.

We now have enough information to decide whether to attach or not. If the Business Object is not connected, we can route the diagram to the Navigate stage to make the attachment. Otherwise, we should avoid the Navigate stage to avoid causing the "already attached" error.

- Below the Read stage add in a Decision to check if the value of the flag Data Item. Use the expression **[Connected]=False**.

- Link the “Yes” branch of the Decision to the Navigate stage and the “No” branch around it. Your Attach page will look something like this:



The Attach page is now able to detect whether it needs to make the attachment or not. This means we can now run the Attach page without worrying if we will cause the “already attached” error.

- Experiment with the Attach page by running it after detaching from Order System.
- Try closing Order System and then running the Attach page.

We will look at reading from fields later on in the course.

6.14. Actions

We now have a basic Business Object that can launch, log in, and close the Order System application. Remember a Business Object can only be used via a Process, so we need to return to Process Studio to continue testing.

An Action stage allows us to use a Business Object in a Process.

Exercise 6.14.1 Using the Business Object

- Save and close Object Studio and close Order System if it is running.
- Create a new Process named **Order System Test Rig** and add a new Action stage.

Figure 94: Action Toolbar Button

- Open the Action properties window and select **Order System** in the first drop-down.
- Now open the second drop-down list but see that it is empty.

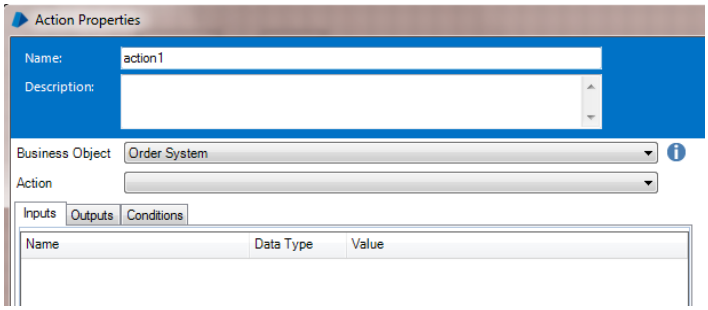


Figure 95: Action Stage Properties

- We know the Order System Business Object has pages but for some reason they are not listed. Think back to when we first used Control Room but could not see our Circular Path Exercises Process initially.

Like a Process, Business Object pages must be published before they are exposed for use. This allows us to control what can and cannot be used.

Exercise 6.14.2 Publishing Business Object Pages

- Reopen the Order System Business Object. You can do this by right-clicking on the Action stage and selecting **View action in Object Studio**.

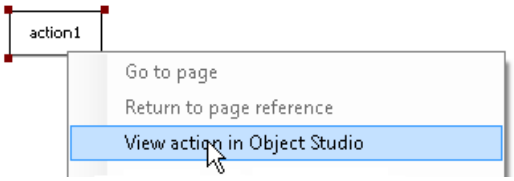


Figure 96: Viewing Action in Object Studio

- Publish the Business Object pages by right-clicking on the page tabs. Notice the icon on the page tab denoting a published page.

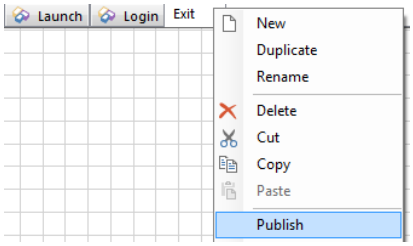


Figure 97: Publishing Business Object Pages

- Save and close the Business Object.
- Return to Process Studio and press the Refresh button to load the latest Business Object definition from the database.

Figure 98: Refresh Button

- Look again at the Action stage properties to see that the second drop-down list has now changed.

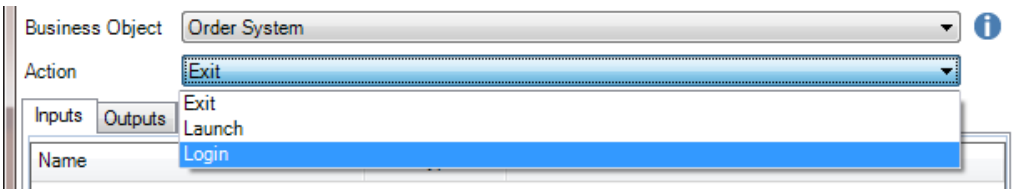
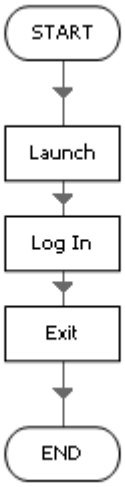


Figure 99: Published Pages in the Action List

- Add two more Action stages to create a Process to launch, log in, and close Order System. It should look something like this:



- Run the Process to check everything works as expected. Remember that a Process must be saved at least once before it can run.

Publish, save, and close the Process, and then run it in Control Room. Notice how the Process runs much quicker in Control Room.

6.15. Action Inputs and Outputs

Earlier we looked at using inputs and outputs with pages and sub-Processes. In fact the most common usage for inputs and outputs is with Action stages.

Just as a Business Object provides a Process with the functionality, it needs to manipulate an application; it is the inputs and outputs of the Business Object's pages that move data back and forth between the Business Object and the Process.

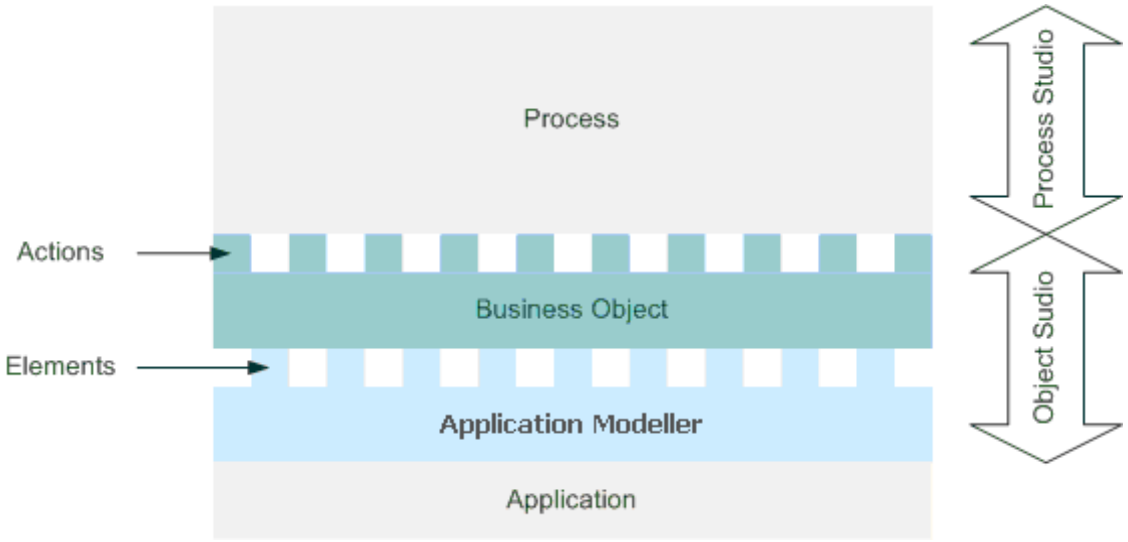


Figure 100: Published Pages in the Action List

Exercise 6.15.1 Adding Log In Inputs

Currently, our Order System Business Object will log in using the same staff number and password every time. This isn't ideal, as in reality we're likely to want to be able to log in using a variety of different user credentials.

- In the Order System Business Object, add **Staff Number** and **Password** input parameters to the Start stage of the Log In page.

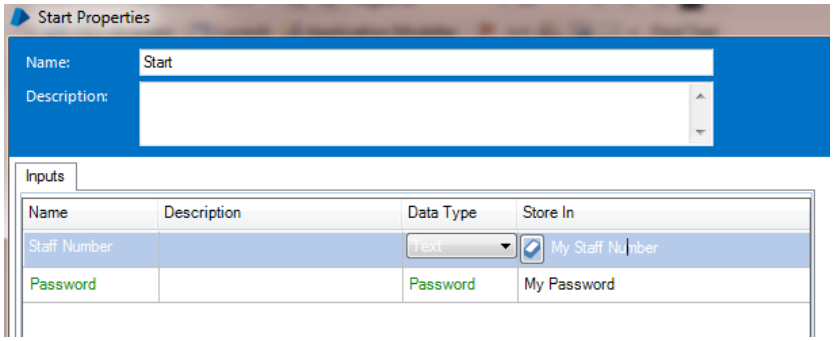


Figure 101: Input Parameters

- Note down a note of the initial values of the two Data Items before changing them to blank.
- Save the changes, open the **Order System Test Rig** Process, and look at the properties of the Log In action.

Previously, when the Log In page of the Business Object didn't have any inputs, the properties of the Action stage in the "Test Rig" Process did not have any input either. Now, though, there are inputs and the Business Object is effectively ***asking*** the process to provide it with data. Essentially, the Business Object is saying to the Process, "What staff number and password do you want me to log into Order System with?"

Key Points

- Inputs to a page in Object Studio provide the means for a Process to supply data to the Business Object.
- In Process Studio these inputs are shown in the properties of the Action stage.

- Refer back to your note and populate the new inputs, then test that they work. For now, don't worry that you can see the password value in plain text.

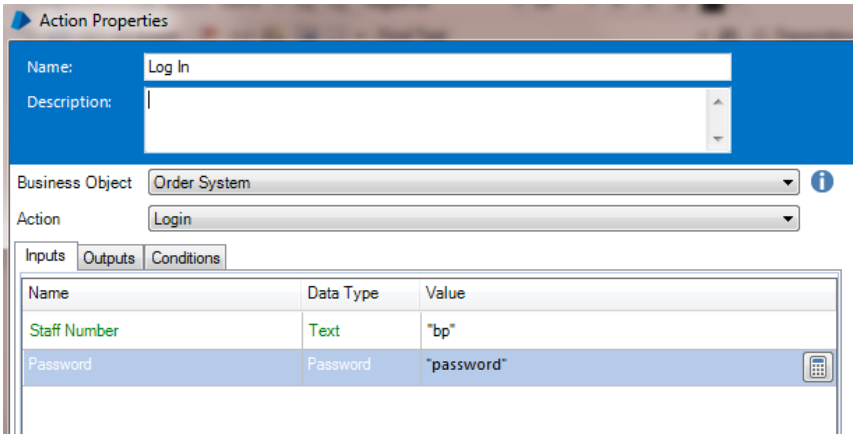


Figure 102: Input Values from Text Expressions

The Business Object no longer possesses staff number and password values because we have made its Data Items blank. The Process is now ***telling*** the Business Object what values to use via the inputs in the Action stage.

The inputs are used to set the current values of the Data Items on the Log In page of the Business Object, and in turn they are used by the Write stage to populate the fields on the application window.

6.16. Data Items as Inputs

Input values are supplied as the result of expressions. These expressions can be made complex if necessary but in reality they are usually made using the names of Data Items.

Exercise 6.16.1 Data Items as Inputs

The Order System Test Rig Process is now supplying the Order System business object with staff number and password values. The Business Object doesn't contain any user credentials anymore and so cannot log in on its own – it must be provided with inputs.

The Process is currently supplying those inputs with the expressions **"bp"** and **"password"**. Let's change those expressions to use Data Items rather than plain text.

- Create a new Data Item named **SN** with data type **Text** and initial value **bp**.

Key Point

- The initial and current values of a Data Item are not expressions, and as such text values do not need quotation marks.

- Create a new Data Item named **PW** with data type **Password** and initial value **password**.
- Amend the inputs in the Action stage properties so that the expressions use the new Data Items instead of text values.

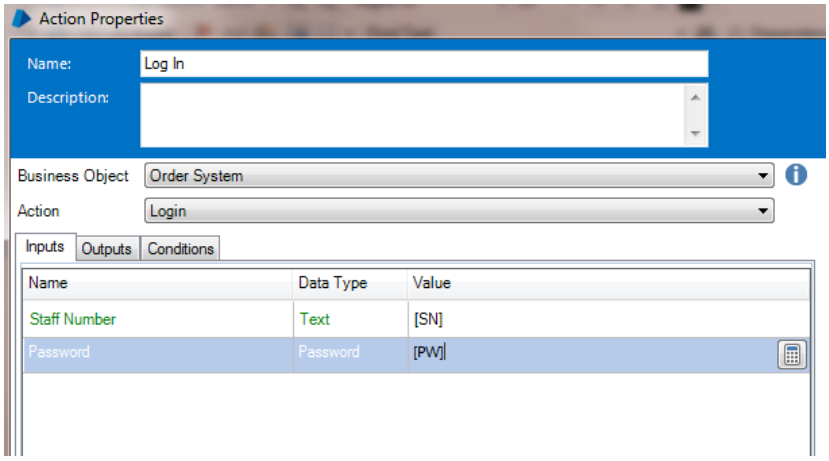


Figure 103: Inputs

- Run the Process again to check it works correctly. You should now see that the user credentials held in the Data Items in Process Studio are passed down into Object Studio and used to log into Order System.

Critically, we now don't need to modify the Business Object to have it log in using different user credentials – we simply provide it with different input values.

6.17. Review

- Application Modeler is used to spy application elements and capture their details as attributes.
- A selection of attributes is used to uniquely identify an element.
- Some adjustment of the attributes may be required to identify an element correctly.
- A Navigate stage is used to perform an action on an element, such as launching the application or pressing a button.
- A Business Object does not have to launch;business objects can also attach to applications that are already running.
- A Write stage is used to send values to the application.
- A Read stage is used to get values from the application.
- A Wait stage is used to pause and wait for the application.
- An Action stage is used to call a Business Object page from a Process.
- Inputs transfer data from a Process to a Business Object.

Blue Prism Keywords

Application Modeler, Spy, Highlight, Element, Attribute, Navigate, Launch, Wait, Read, Write.

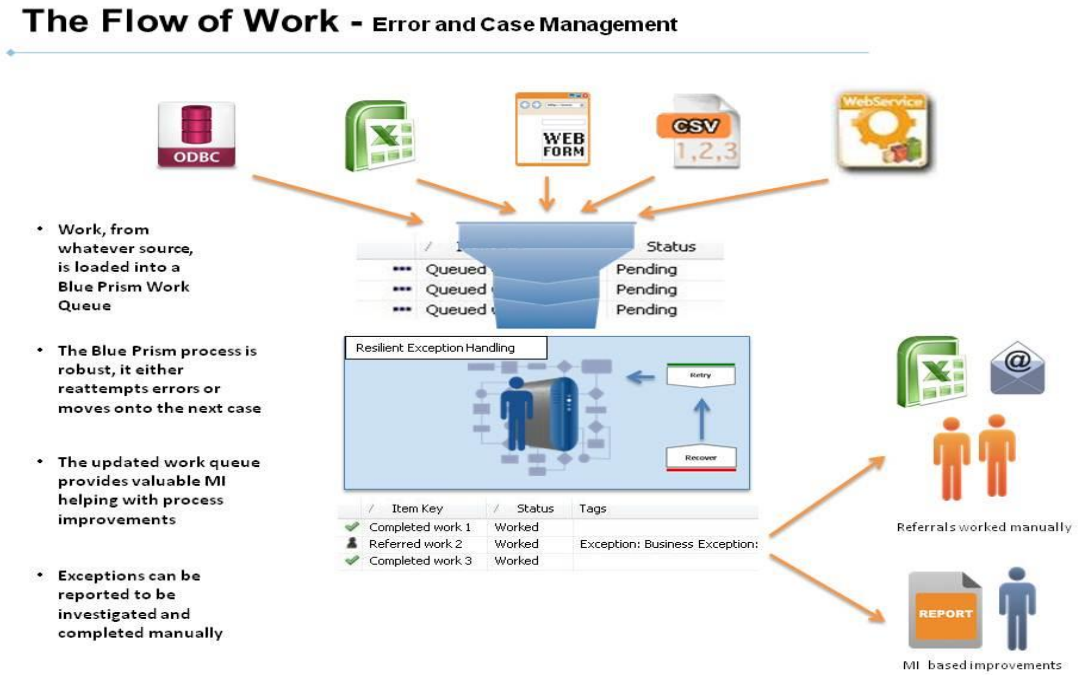
7. Overview of Error and Case Management

As the next two chapters, Error Management and Case Management, are related, it is useful to look briefly at the theory behind them. Error Management looks at the way Blue Prism can deal with problems it may encounter when running a Process and Case Management discusses how Blue Prism manages its workload.

Without Error and Case Management the following issues would exist:

- Unexpected errors within Processes and Business Objects would be unhandled, causing Processes to terminate whenever anything unexpected occurred.
- Every potential problem would need to be identified and catered for within the Process or Business Object. This is effectively impossible as it requires planning for the unknown.
- Processes would need to be continually monitored and restarted if there were any problems.
- Monitoring workloads and work rates would be difficult, with no view of what was happening to individual cases.
- There would be no MI (Management Information) available for reporting or to assist process improvement.

These issues would lead to automated solutions that were difficult to control, maintain, and analyze. Together, Error Management and Case Management enable the creation of robust Blue Prism solutions that are reliable, easily monitored and have standard MI reporting.



8. Error Management

There are two main features of Error Management:

First, the occurrence of unexpected events in a Process or Business Object can be identified (or to use a programming analogy, ***caught***). This gives the developer the opportunity to build additional robustness into the logical flow in an attempt to ***handle*** the unexpected. For example, by taking the flow along a different route or by repeating the sequence that failed.

Second, cases that could not be worked can be identified. If, for whatever reason, a case cannot be completed, rather than stopping, the case can simply be marked as an ***exception*** and the process can move on to the next case. As you may have realized, this feature is naturally related to Case Management.

In our previous examples and exercises, we have only looked at ideal scenarios and have not really considered what to do when things do not go to plan. In the real world, problems will inevitably occur and we should aim to create a robust solution capable of dealing with them. Thorough testing should reveal the most common problems and we can then make adjustments for them.

We have seen how to use a Wait stage to allow a Business Object to pause for an application, but as yet we have not considered what to do if our Wait stage times out. Maybe a simple retry is enough to resolve an issue, or perhaps there is something unusual with the data we are working with (e.g., a suspended account) and retrying is of no use. It would be better to make a note of the problem and move on.

Simply moving to the next job might be enough, or it could be that we need to make a fresh start and restart the target application(s). More seriously, there could be a problem that is beyond our control (e.g., an application is offline) and we have no choice but to stop completely.

These kinds of problems and errors are known as ***exceptions***, and the logic used to cater for them is known as ***exception handling***.

8.1. Exception Handling

Rather than failing in the event of an exception, Processes and Business Objects can be designed to handle the exception and try to keep going.

Although not every type of exception is recoverable, and sometimes it makes no sense to attempt a recovery, the ability to handle exceptions at least provides us with the opportunity to decide what to do when something goes wrong and the logical flow has strayed from the ideal path.

We have seen that a solution is formed by Processes and Business Objects. Exceptions can potentially occur at any point and we shall see how exceptions are handled in such an arrangement.

Exception handling is done with a set of stages we have not used before.

8.2. Recover and Resume

As the names suggest, Recover and Resume are stages used to attempt to salvage and move on from an exception. The use of these kinds of stages also introduces a new concept to the design of a diagram.

Exercise 8.2.1 Recovering from an Exception

- Create a new Business Object named **Exception Exercises Object**.
- Add two number Data Items named **X** and **Y** with initial values of **1** and **0**.
- Add a Calculation stage that divides **X** by **Y**. Yes, that's right, divide by zero!

- Save the Business Object and run the page. An exception should occur and the following message should appear:

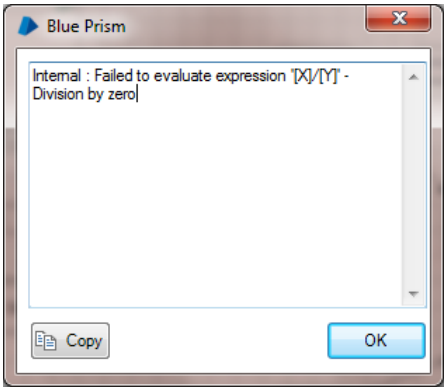


Figure 104: Divide by Zero Message

- Now add a **Recover** stage, a **Resume** stage, and another End stage and link them up as shown below. Run again to see the effect.

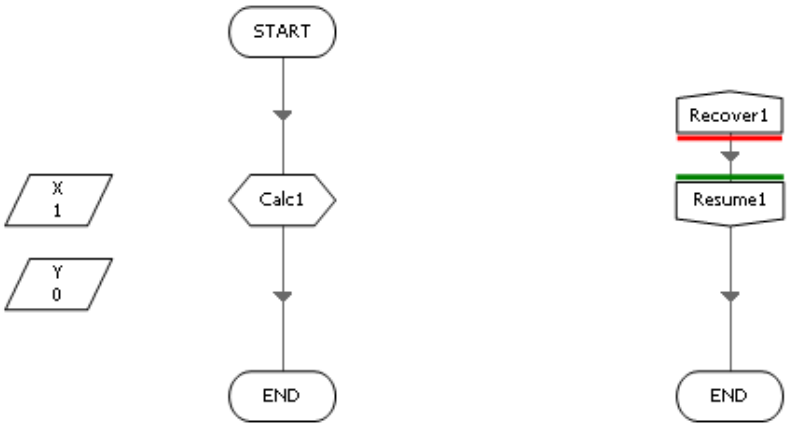


Figure 105: Recover and Resume Stages

Rather than fail at the Calculation stage, the exception has been ***recovered***. Critically, the flow of the page has jumped between two stages (Calc1 and Recover1) that are not connected by a physical link.

The Recover stage attracts, or ***catches*** the exception, giving the designer (i.e., you) the opportunity to create some sort of recovery sequence. In this basic example, the recovery sequence is simply to take the diagram to a different end point.

When an exception is caught, the Business Object (or Process) is said to be in ***Recovery Mode***, meaning the exception is "live". Passing through the Resume stage diffuses the exception and enables the diagram to come out of Recovery Mode and continue a normal flow. Importantly, the Resume stage does not fix anything - that is the responsibility of the designer.

Key Points

- A Recover stage acts like a magnet – exceptions are drawn to it and the diagram flow will ***jump*** from the exception point to the Recover stage.
- When Blue Prism is on the path between a Recover and a Resume it is said to be in ***Recovery Mode***.
- A Resume stage neutralizes the exception but ***does not fix the problem***.
- Once past the Resume stage, the exception is effectively "gone" and the diagram can continue its normal flow.
- By default a page only uses one Recover stage and any others will be redundant.

As well as handling unexpected exceptions like this, we can also create them on purpose. This is known as ***throwing*** an exception.

8.3. Throwing Exceptions

In some circumstances it can be to our advantage to generate exceptions deliberately. If a "divide by zero" error was the reason a real-life case could not be worked, we might prefer to mark the case with a friendlier, more humane message to assist any manual rework that may be required.

Exercise 8.3.1 Throwing an Exception

- Add a Decision stage above the Calculation and set its expression to check if **Y** is zero.
- Connect the negative link to the Calculation and the positive link to a new Exception stage.

Figure 106: Exception Toolbar Button

Figure 107: Exception Stage

- Rerun and see how the Exception stage is used to **throw** an exception

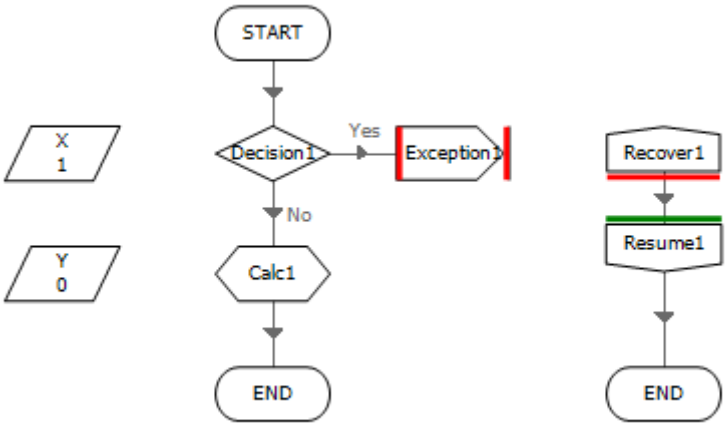


Figure 108: Throwing an Exception

Exercise 8.3.2 Exception Type

When we throw an exception, we can give the exception meaning by embedding information with it.

- Open the properties of the Exception stage and populate the **Exception Type** and **Exception Detail** fields with something like “Invalid data found” and “Y is zero”. (You can type in a new exception type if there are none available in the drop-down list.)

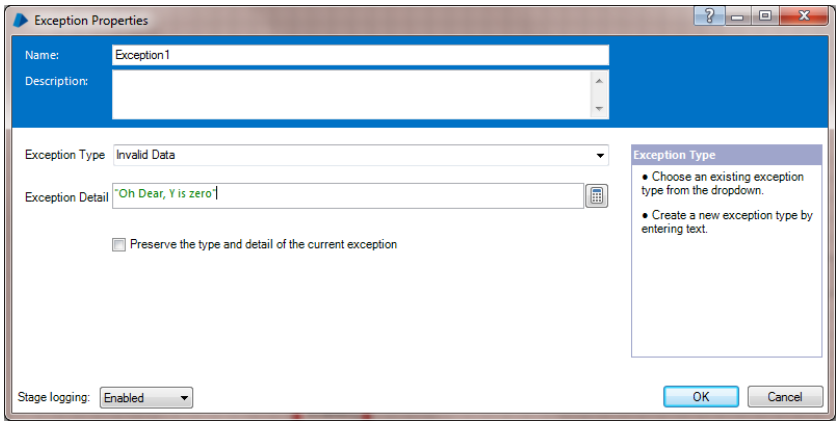


Figure 109: Exception Properties

Key Points

- The Exception Type value in an Exception Stage is not an expression and does not need quotes.
- The Exception Detail value is an expression and does need quotes.

- Add a Calculation stage between the Recover and the Resume. Step through and stop on the new Calculation.
- Open the Calculation properties and look at the “Exception” functions in the bottom left corner.

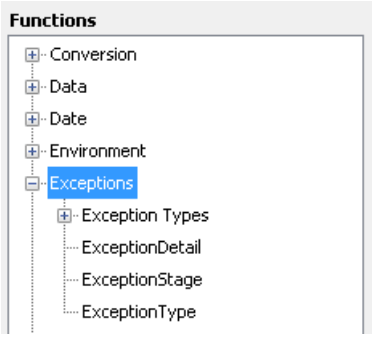
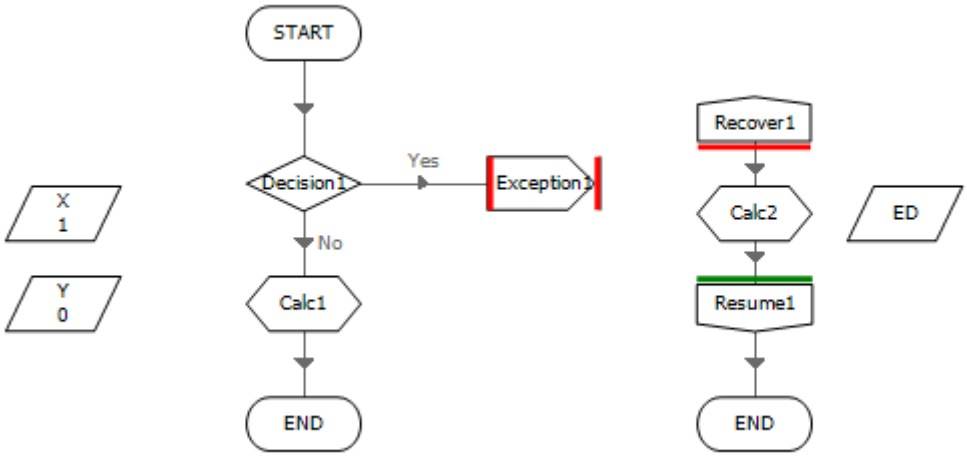


Figure 110: Exception Functions

- Drag **ExceptionDetail** up into the expression editor to create the function ExceptionDetail().
- Type **ED** into the “Store In” field and press the auto-generate button to create a new Data Item.

Figure 111: The Auto-generate Button

- Your diagram should look something like this:



- Run your page to see that the exception detail is captured in the Data Item.

At this point, the significance of this feature may not be apparent but we will see how the source of the exception and the recovery logic can exist in ***separate*** diagrams.

Key Point

- Exception functions like ExceptionDetail() and ExceptionType() cannot be used anywhere other than in between a Recover and a Resume, i.e., in ***Recovery Mode***.

Exercise 8.3.3 An Unhandled Exception from an Action

- Cut the recovery logic – Recover1, Resume1, Calc2, ED, and the second End.
- Publish the page, then save and close the Business Object.
- Create a new Process named **Exception Exercises Process** and add an Action stage that uses the Business Object we have just saved.
- Save the Process once to commit it into the Blue Prism database.
- Run the process and you should see a familiar exception message. What do you think has happened?

The answer is that the exception has traveled upwards from the Business Object and into the Process.

Exercise 8.3.4 Handling an Exception from an Action

- Paste the recovery logic cut from the Business Object into the Process so it looks like this:

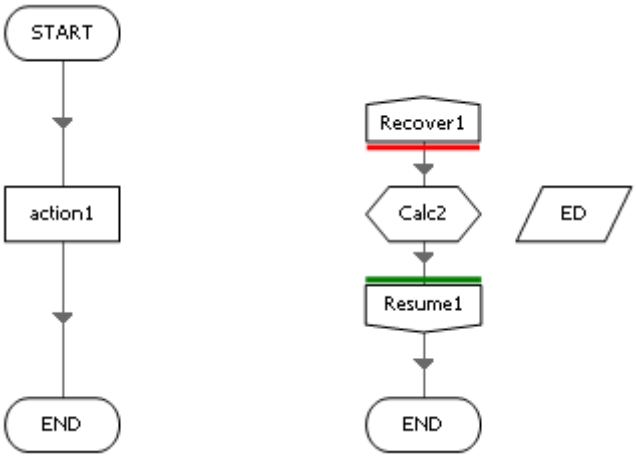


Figure 112: Recovering from an Action

- Reset and run the page again. What is happening now?

We have deliberately chosen not to deal with the exception in the Business Object and in doing so we have allowed the exception to escape upwards into the Process. Fortunately, the Process contains some recovery logic to ***handle*** the exception. Admittedly the handling here is minimal, but the intention is to illustrate the concept.

Exercise 8.3.5 An Unhandled Exception from a Page

- Add a new page (keeping the default name) to the Process.
- Cut everything from the Main page and paste it onto a new page.
- Go back to Main Page and add a Page Reference where Action1 used to be.

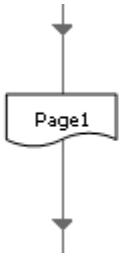


Figure 113: Page Reference Stage

- Add a new End stage and link it with the Start and Page Reference stage.
- Reset and run the Process. What has happened?

As before, the Business Object did not handle the exception and it escaped upwards into the Process, where it was handled on Page1.

Exercise 8.3.6 Handling an Exception from a Page

- Move the recovery logic back on to Main Page and rerun the Process. What has happened now?

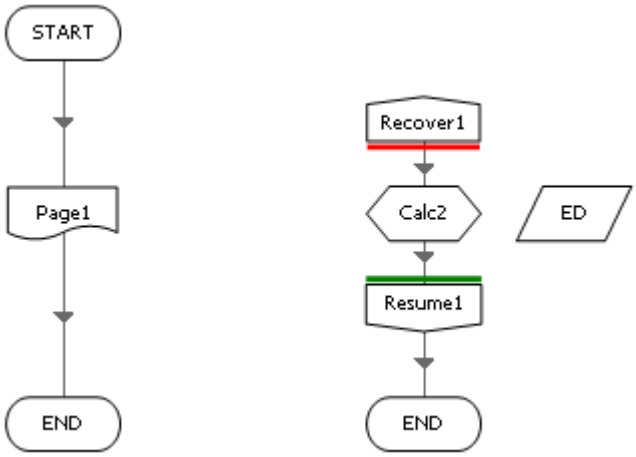


Figure 114: Recovering from a Page

Again, the exception has gone unhandled in the Business Object and moved upwards to Page1 in the Process. However, since we have now moved the recovery stages, Page1 also leaves the exception unhandled. The exception moves upwards from Page1 on to Main Page, where it is eventually handled.

Exercise 8.3.7 Re-throwing an Exception

- Copy the recovery logic on Main Page and paste a copy on to Page1.
- Add a new Decision stage between the Recover and the Calculation.
- Link the Decision with a new Exception stage so the page looks like this:

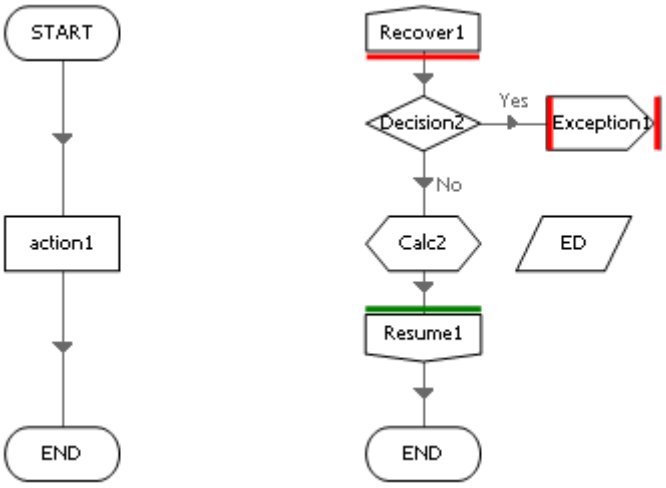


Figure 115: Re-throwing

- Make the Decision expression check whether the recovered exception is a “divide by zero” exception. There are a number of ways to do this, and the expression InStr(ExceptionDetail(), “Y is zero”)>0 is an example.
- Open the Exception stage properties and check the **Preserve the type and details of the current exception** checkbox.

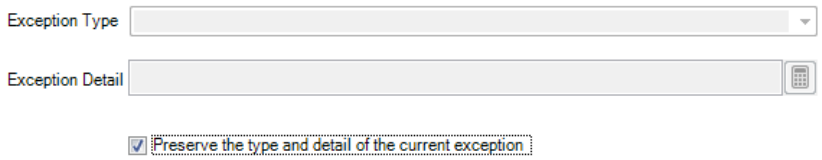


Figure 116: Preserve the Type and Detail of the Current Exception

- Re-run the Process. What has happened?

As before, the exception has gone unhandled in the Business Object and moved up to Page1 in the Process. It is caught on Page1 but, importantly, it is ***re-thrown*** so that it remains unhandled and continues to move upwards.

Effectively Page1 has decided to exempt itself from dealing with any "divide by zero" problems and simply allows that type of exception to keep moving upwards. Fortunately, in this scenario, the Main page is able to handle the exception.

8.4. Preserving the Current Exception

When using the Exception stage, it is important to understand the "Preserve" checkbox and when to use it.

When the checkbox is checked, you will have noticed the Exception Type and Exception Details fields are disabled. Checking the checkbox indicates that the current exception is to be re-released or thrown again and no new details are necessary.

For this reason, the "Preserve" check box must only be used in Recovery Mode, i.e., somewhere between a Recover stage and a Resume stage. Using the checkbox outside Recovery mode will in itself generate an exception.

The following diagram illustrates correct use of the "Preserve" checkbox. The right-hand exception is thrown in Recovery Mode, so the checkbox is checked. The other exception is a new exception thrown outside Recovery Mode, so the checkbox is not (and must not be) checked and exception details are supplied.

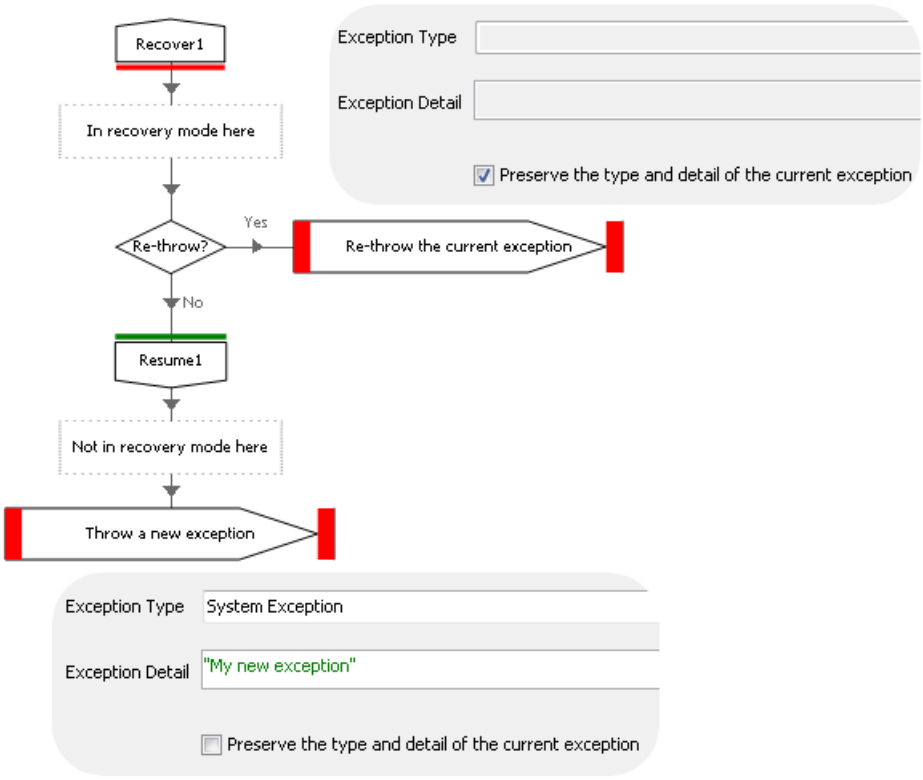


Figure 117: Exceptions and the Correct Use of the "Preserve" Checkbox

Misuse of the “Preserve” check box is illustrated below. The right-hand Exception stage will generate a new exception in addition to the current exception. Any subsequent exception handling will deal with the first exception but will not handle the other, potentially resulting in undesired effects.

The other Exception stage will itself generate an error because it is wrongly using the checkbox – there is no current exception to preserve once past the Resume stage.

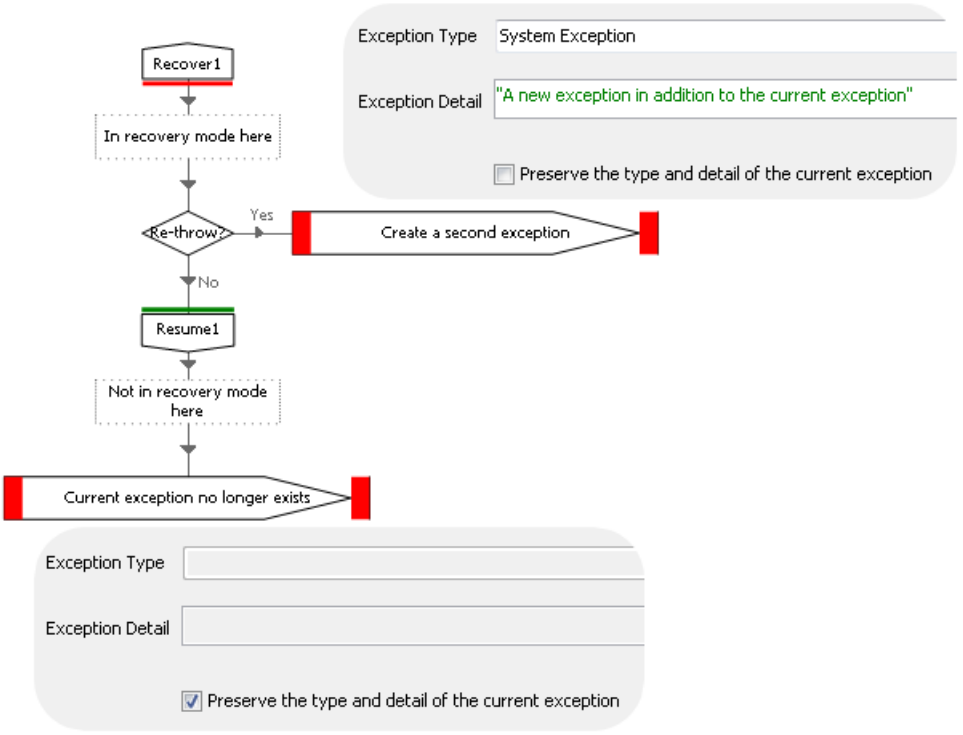


Figure 118: Exceptions and the Incorrect Use of the "Preserve" Checkbox

Key Points

- The “Preserve” checkbox is only applicable when in Recovery Mode.
- Re-throwing an exception will generate a ***second*** exception unless the “Preserve” checkbox is checked.
- To throw a new exception, make sure it is thrown ***after*** a Resume stage, thereby neutralizing the current exception before the new one is created.

Exercise 8.4.1 An Unhandled Exception

- Delete the recovery logic on the Main page and see what happens when you run the Process again.

The exception has gone unhandled all the way up from the Business Object up through the Process pages. The Main page was unable to recover and so the Process has failed.

8.5. Exception Bubbling

The way in which an exception moves upward through the layers of a solution is known as ***bubbling***. An exception will bubble upwards until it is handled, and if it is not handled, it will eventually bubble up to the Main page of the Process and cause it to fail.

We can use bubbling to our advantage, as we do not have to catch an exception straight away; we can let it bubble up and handle it at a higher level in the hierarchy.

Even when we do catch an exception, we have the option to throw it upwards again. Deciding where and when to handle exceptions is a key part of solution design that we shall return to.

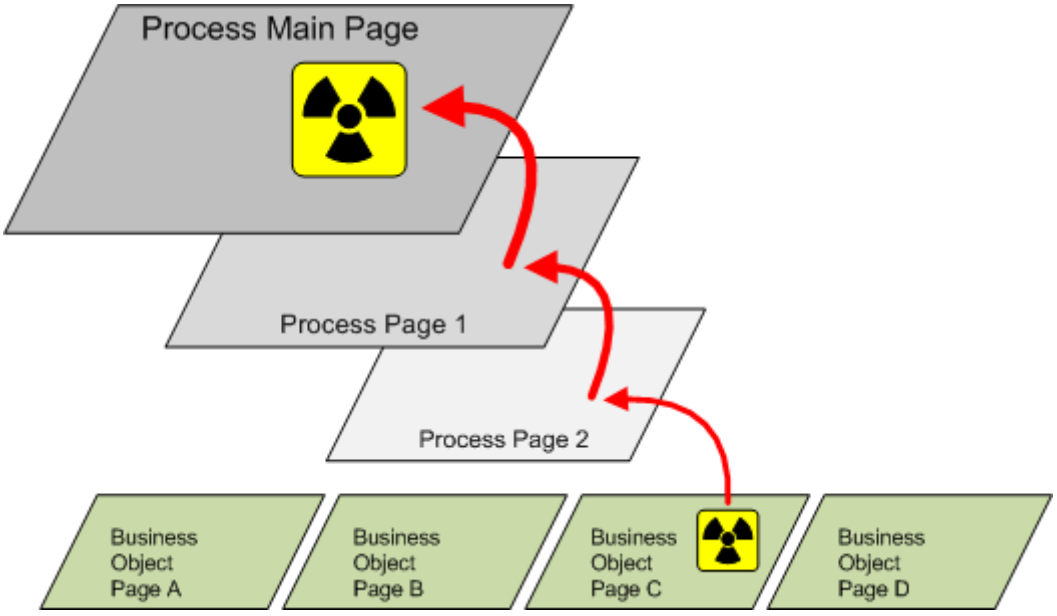


Figure 119: Exception Bubbling

Key Points

- Exceptions travel upwards towards the Main page of the Process.
- Exception handling can be on the same layer as the source of the exception or above it.
- Unless the exception is intercepted it will terminate the process.

8.6. Exception Blocks

We can add a greater degree of control to exception handling with the use of Exception Blocks. A Block is used to isolate an area of a diagram that a Recover stage is responsible for.

Without a Block, a Recover stage will handle any exception on that page. When a Recover stage sits inside a Block it will ***only*** catch exceptions in that Block and ***ignore*** all others.

Blocks enable us to use more than one Recover stage on the same page.

Figure 120: Block Toolbar Button

Figure 121: Block Element

Exercise 8.6.1 Using Blocks (Part I)

- Close the Process and reopen **Exception Exercises Object**.
- Add new Recover, Resume, and End stages to Action1, placing them some distance to the right of the diagram.

- Draw a Block around the Exception stage to make the diagram something like this:

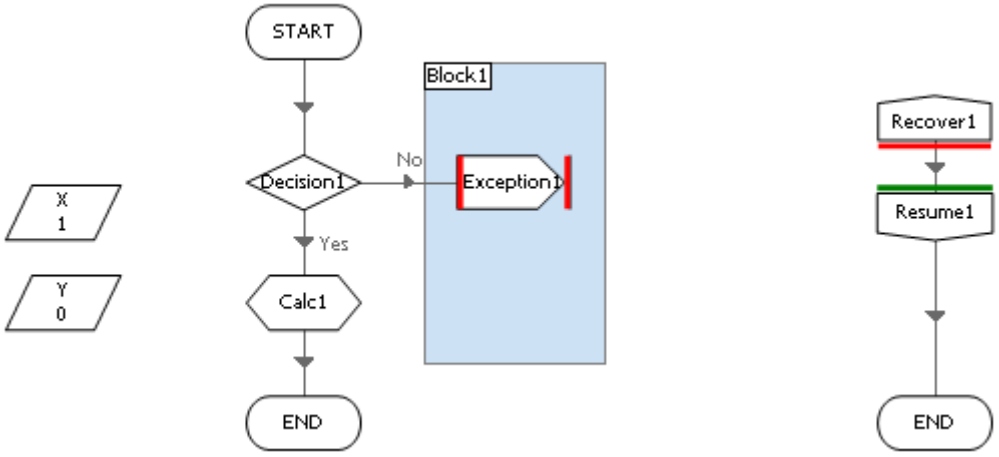


Figure 122: Recover Stage outside a Block

- Run to see the exception handled as normal; the Block has made no difference.

Blocks only work if they contain a Recover stage - without a Recover, a Block has no effect.

Exercise 8.6.2 Using Blocks (Part II)

- Now stretch the Block to the right and add another Recover, Resume, and End as shown below.

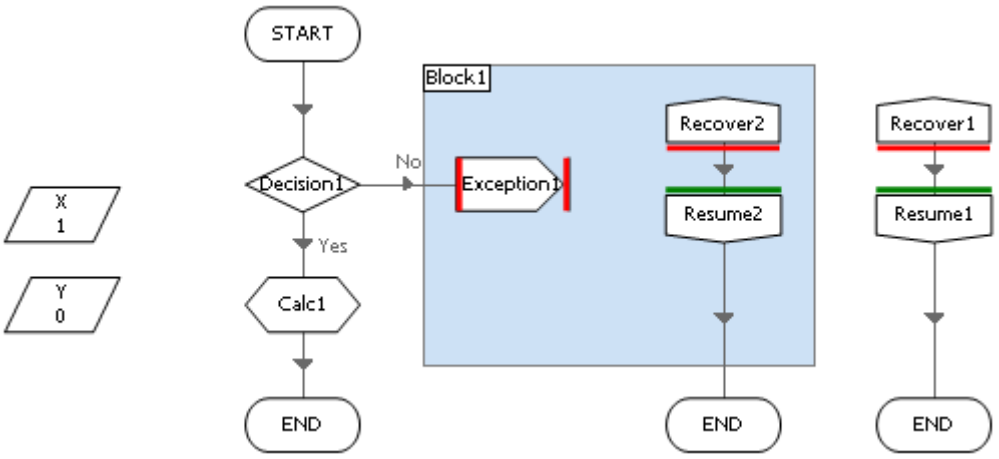


Figure 123: Recover Stage inside a Block

- Rerun to see how the exception is now handled by the new recovery section.

Because Recover2 is ***inside*** the Block, it assumes responsibility for any exceptions occurring in that Block.

Exercise 8.6.3 Using Blocks (Part III)

- Resize the Block again so that the Exception stage is no longer contained and the diagram looks like this:

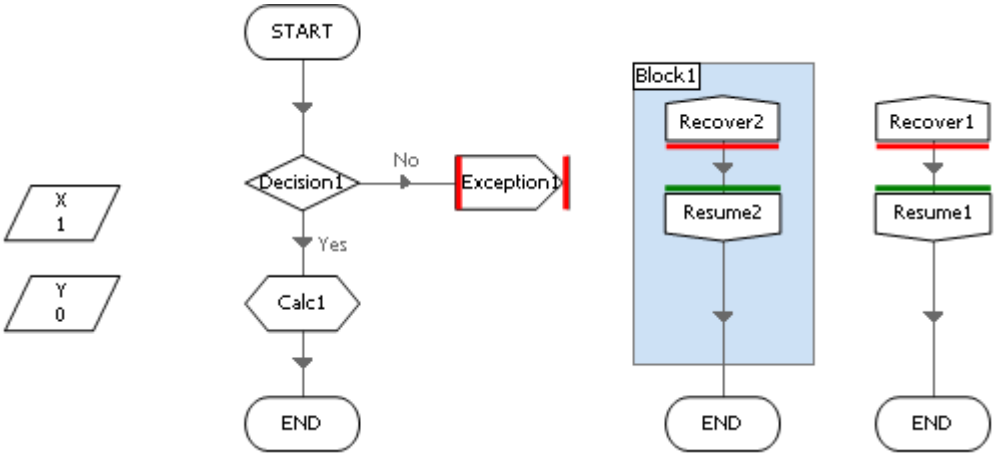


Figure 124: Exception Stage outside a Block

- Rerun to see that the exception is again handled by the first recovery section and the second recovery section has become redundant.

Key Points

- One Recover stage will handle all exceptions on a page.
- Without Blocks, any extra Recover stages on a page will be superfluous.
- Blocks are a way of using multiple Recover stages on the same page.
- An exception generated inside a Block will be handled by the Recover inside that Block.
- If a Block does not contain a Recover stage, the Block will have no effect.
- One Recover stage will handle all exceptions generated inside its Block.
- Any extra Recover stages in a Block will be superfluous.
- Blocks should not overlap and Blocks cannot be nested (i.e., Blocks within Blocks).

8.7. Exception Handling in Practice

We've seen how Blue Prism can detect and handle exceptions but how is this used in practice. Let's look at an example, consider the following ***Main Page*** of a process:

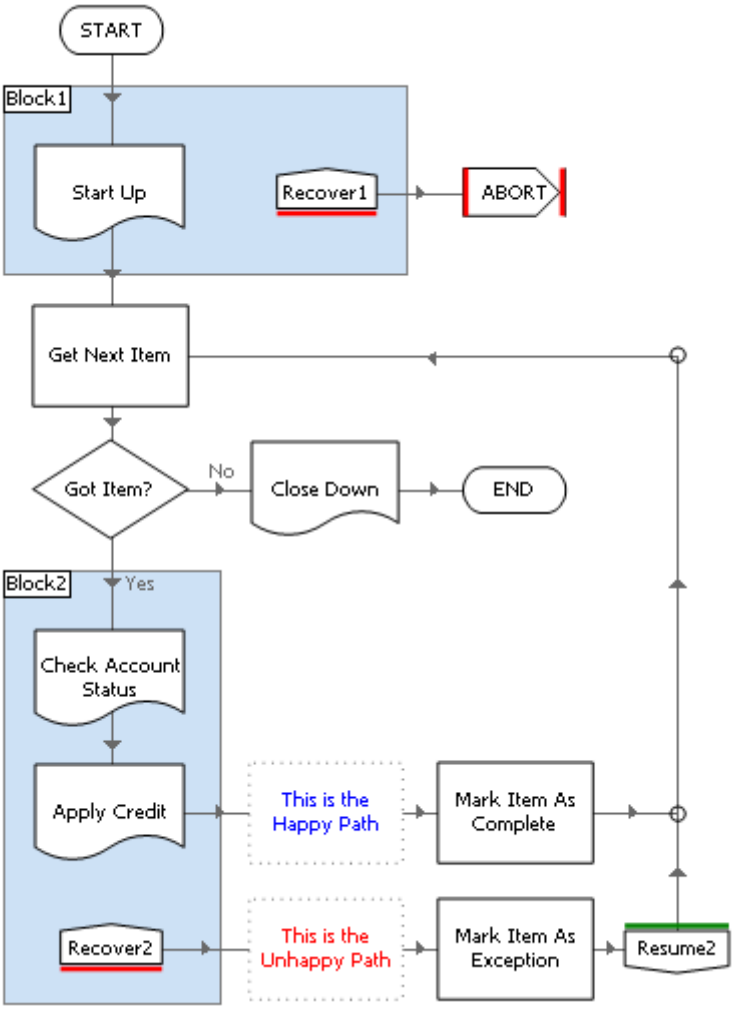


Figure 125: Exception Handling Example - Main Page

In the example above, we can see two main types of exception handling, one where we will allow the process to terminate because we cannot continue, and one where we will handle an exception and resume processing.

Consider the first block, ***Block1***. Any exception bubbled up from the ***Start Up*** page will be caught by the ***Recover1*** stage. But, because we've been unable to start the target application, the process cannot continue and we will simply release the exception which, because we are throwing it on the Main Page, will cause the process to terminate.

The ***Start Up*** page logic is as shown below:

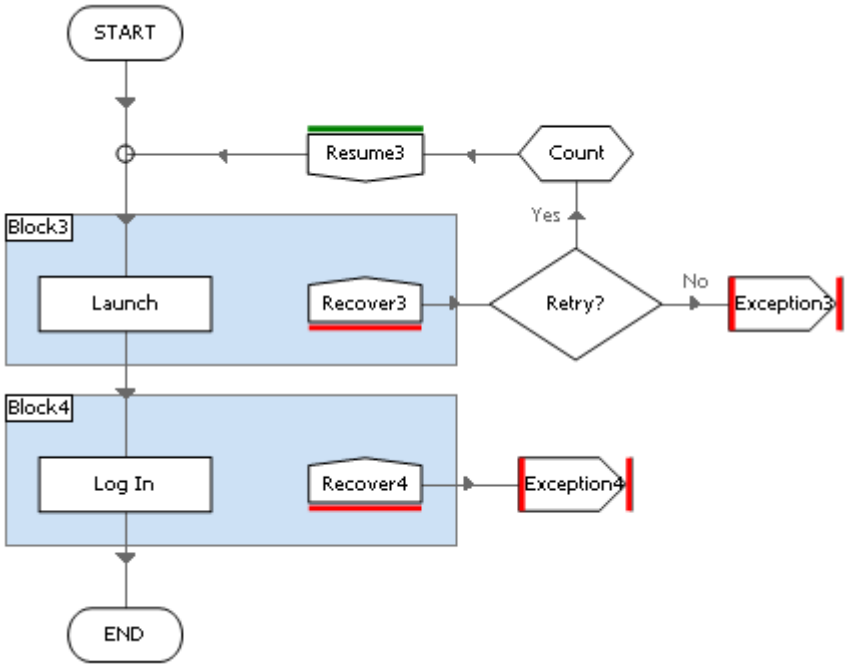


Figure 126: Exception Handling - Start-up Page

The exception handling in ***Block3*** enables multiple attempts to launch the application, and a count is kept to limit the number of attempts. Once the maximum number of attempts is reached, the exception is not handled and allowed to bubble up to the parent page. And, because Main Page will not handle any exceptions coming from the Start Up page, the process will terminate.

The exception in ***Block4*** is more basic – only one attempt to log in is possible, and if an exception occurs (say because the password has expired), the exception is simply allowed to bubble up and kill the process.

Returning to Main Page, consider the logic around ***Block2*** below.

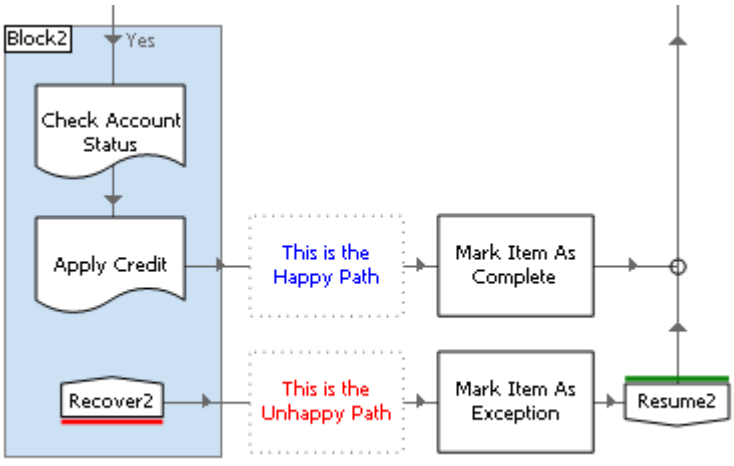


Figure 127: Exception Handling - Block2

Ideally every work queue item will be worked to completion and the process will keep to the ***happy path***. However, the process makes provision for the possibility an exception bubbling up from either the ***Check Account Status*** and

Apply Credit pages with ***Recover2***. This ***unhappy path*** logic enables the process to record the exception detail in the queue item and, critically, to ***resume*** the normal flow, ready to get the next item from the queue.

8.8. Review

- Problems are inevitable in a real-world environment.
- Exceptions can be handled to make a robust solution.
- The nature of an exception can be interpreted to determine how or if the situation can be resolved.
- An exception will bubble up towards the Main Page of a Process until it is handled.
- Exceptions can be handled anywhere in Business Objects and Processes.
- An exception handling policy is part of the solution design.
- Recover, Resume, and Exception stages are used to create exception handling logic.
- Like a Start stage, the Recover stage cannot receive inbound links from other stages; instead the flow jumps to it when an exception occurs.
- Similarly, an Exception stage does not link to another stage because the flow jumps from it.
- A page can only use one Recover stage unless Blocks are used.
- A Block is used to isolate an area of a page and provide it with its own recovery logic.
- A page can have multiple Blocks, as long as they do not overlap.

Blue Prism Keywords

Exception, Recover, Resume, Throw, Bubble, Block.

9. Case Management

Almost every Blue Prism Process will make use of a ***work queue***, which is essentially a list of jobs, or cases. New cases can be fed into the queue, and the queue can be updated with results as each case is worked.

A work queue provides the following features:

- Multiple machines can work from the same queue at the same time, each retrieving different cases to work.
- An individual case can be marked as "complete" if it has been worked satisfactorily, or marked as an "exception", if it could not be completed.
- Queued work can be monitored and maintained from Control Room.
- MI such as volumes, performance levels, and exception details can be extracted from queue data.

A Blue Prism Process is intended to obey instruction to do a repetitive task. That instruction could be some kind of list feeding the Process with work, for example a spreadsheet. Alternatively, a Process might be designed to wait for work to appear on an ad hoc basis, for example files appearing in a folder.

Realistically we should expect there will be problems and accept that some cases will succeed and some will not. We may also instruct Blue Prism to purposely disregard certain cases. Whatever happens, we will need to record results.

If there are too many cases for one instance of Blue Prism to cope with, we may want to run the same Process on multiple machines to share out the workload.

The Work Queue feature provides this functionality to store, manage, share, and report on Process work.

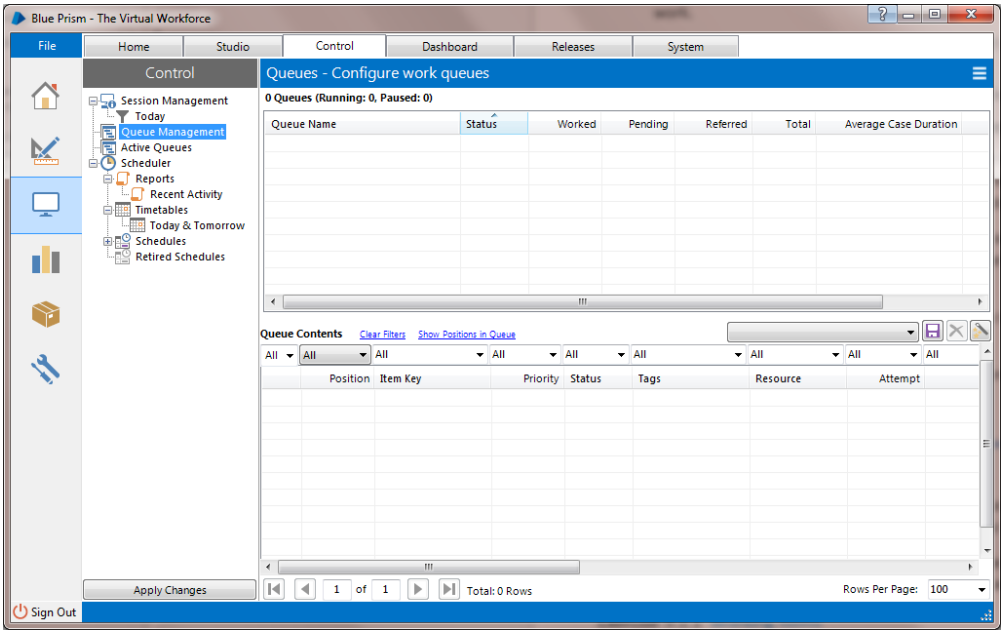


Figure 128: Queue Management

9.1. Review

A work queue is an internal configurable list that enables a Process to manage its workload. A Process can use different work queues and a work queue can be shared by multiple Processes if required.

The Queue Management tab in Control Room provides the operational user interface for work queues. The tab is divided in two; a list of queues is shown in the upper half and the lower half shows a list of the items in a queue.

Exercise 9.1.1 Working Items

- Create a Process named **Queue Exercises 1** and add a new Action stage.
- Open the Action properties and select **Internal - Work Queues** from the list of Business Objects.

Key Point

- Blue Prism has some Business Objects that come as part of the software. These "internal" Business Objects are not diagrams and cannot be seen or changed, only used.

- Select **Get Next Item** from the list of actions and enter **"Queue1"** (remember the quotes!) as the **Queue Name** input parameter value.
- Select the Outputs tab to see the output values. Use the auto-generate buttons in the Store In column to create the output Data Items.

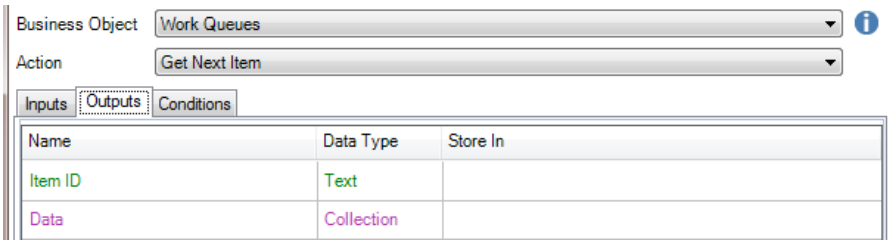


Figure 129: The Auto-generate Button in the Store In Column

> *Tip: Data Items created like this are automatically given the same name and type as the output.*

- "OK" the properties and rearrange the new Data Items on the page. Right-click on the End stage and select Breakpoint from the mouse menu.
- The diagram should now look something like this:

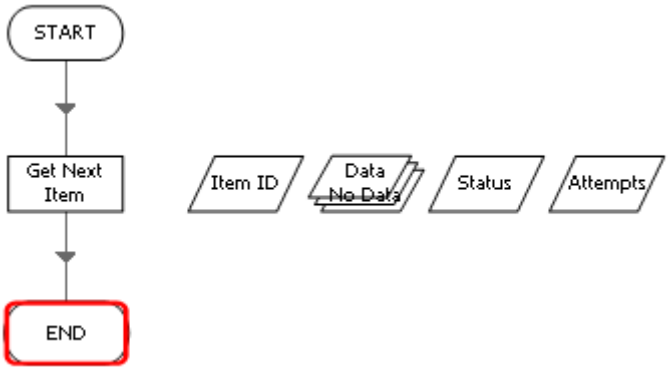


Figure 130: Get Next Item

- Save the Process and run the page.
- Close the Breakpoint message when it appears but do not press the Reset button.

- Open the properties of the Data Items to see their current values.

Exercise 9.1.2 Completing Items

- Leave the Process at the Breakpoint and bring Control Room to the front.
- Refresh the screen, either by pressing the Refresh button, on the drop down menu , or the F5 key. Notice how the first item in Queue1 is now marked with a yellow lock icon, .
- Return to the diagram and pull the End stage down to make room for a new Action stage.
- Select the **Internal - Work Queues** Business Object as before, but now choose **Mark Completed** from the second drop-down list.
- Drag in one of your existing Data Items for the input value and press "OK".

> *Tip: Every item is given a unique ID consisting of a series of characters and numbers. This kind of ID is known as a Globally Unique Identifier or GUID.*

- Your diagram should now look something like this:

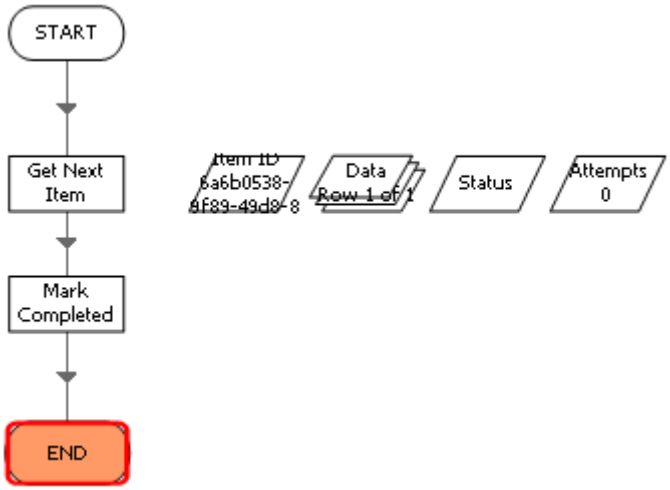


Figure 131: Mark Completed

- Remove the Breakpoint from the End stage and right-click on the Mark Completed action.
- Select Set Next Stage from the mouse menu and then step through to the end of the Process.
- Go back to Control Room and refresh the Queue Management screen.
- See that the first item is now marked with a green check mark ( ).

Exercise 9.1.3 Exception Items

- Return to the Process and change the second Action stage to use **Mark Exception** instead of Mark Completed. Choose your own value for the Exception Reason input and enter False as the Retry input.

> *Tip: Remember an input takes its value from the result of an expression, and in an expression text values must be written in quotes, **"like this"**. The value of a flag does not need to be written in quotes - it is either **True** or **False**.*

- Run the Process again and then see how the next item in the queue has been marked with an exception icon, .
- Scroll to the right to see the Exception Reason column where you should see your input value.

Key Points

- In Control Room an unworked or *pending* item will be marked with three blue dots, like this: •••.
- Locked items are marked with a padlock, like this: .
- Completed items are marked with a green check mark, like this: .
- Exception items are marked with a purple flag, like this: .

Exercise 9.1.4 Locking Items

- Save the Process and then use Save As to create a copy named **Queue Exercises 1a**.
- Open the original process too so that you have two Process Studios open.
- Step through both Processes to show that when one Process has got an item, the other Process cannot get the same item because the queue has locked it.

Key Points

- The special "Work Queue – Internal" Business Object is used to interact with a work queue.
- When an item is being worked, it is "locked" so no-one else can work it.
- When an item is marked as "complete" or "exception", the lock is released.
- An item is referred to by its unique Blue Prism ID.
- Each item has its own Collection to hold data.

9.2. Queue Items

A queue needs to be filled with items before it can be used and the work queue Business Object provides a way to add new items using a Collection. Here we will be using a collection as an input to an action, rather than as an output as we did previously.

Exercise 9.2.1 Adding New Items

- Start a new Process named "**Queue Exercises 2**".
- Create a new Collection with columns matching the output Collection in the first Process. Add two fields named **Account ID** and **Note Text** so your collection properties look as follows:

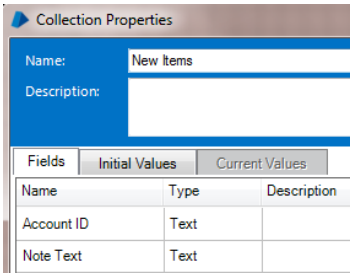


Figure 132: Collection Fields

- Name the Collection **New Items** and complete a handful of new rows on the **Initial Values** tab. Type in what you like as the values.

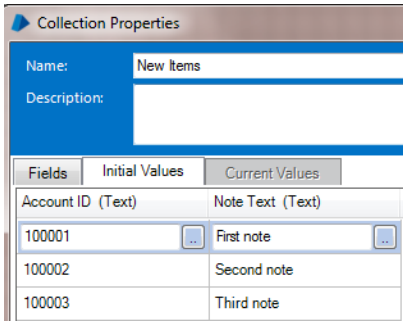


Figure 133: Collection Initial Values

Key Point

- Like in a Data Item, the initial and current values of a Collection are not expressions and text values do not need quotes.

- Create a new Action stage that uses the **Add to Queue** action from the **Internal - Work Queues** Business Object.
- Enter **"Queue1"** (remember the quotes!) as the Queue Name input parameter and drag in your Collection as the Data input parameter. Leave the other input parameters blank for now.
- Link up the new Action stage and save the Process. It should look something like this:

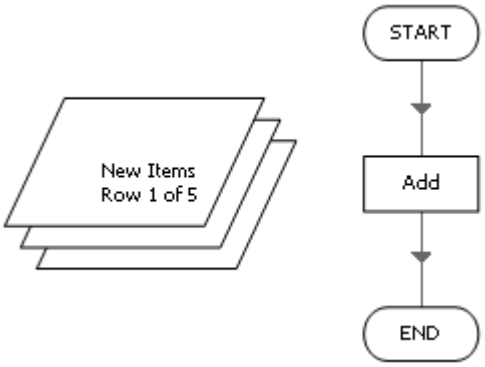


Figure 134: Add to Queue

- Run the Process then go back to Control Room and see your new items added to the queue.

> ✱ *Tip: Inputs and outputs can sometimes be optional and left blank.*

As items get marked as "complete" or "exception", the queue will eventually have nothing left available to work. The signal to a Process that the queue has nothing available is that the outputs from Get Next Item are returned empty.

Exercise 9.2.2 Working All Items

- Go back to the **Queue Exercises 1a** Process and add a decision after the **Get Next Item**.
- The easiest way to know if Get Next Item has got anything is to check the **Item ID** output. Use the expression **[Item ID]<>""** in your Decision expression.
- Link the Yes branch of the Decision to the Action stage.

- Link the No branch to a new Anchor stage. The Anchor stage is an inert stage that has no effect on diagram logic. Its purpose is to join links together, either to redirect a link or create a long link.

Figure 135: Anchor Toolbar Button

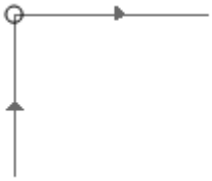
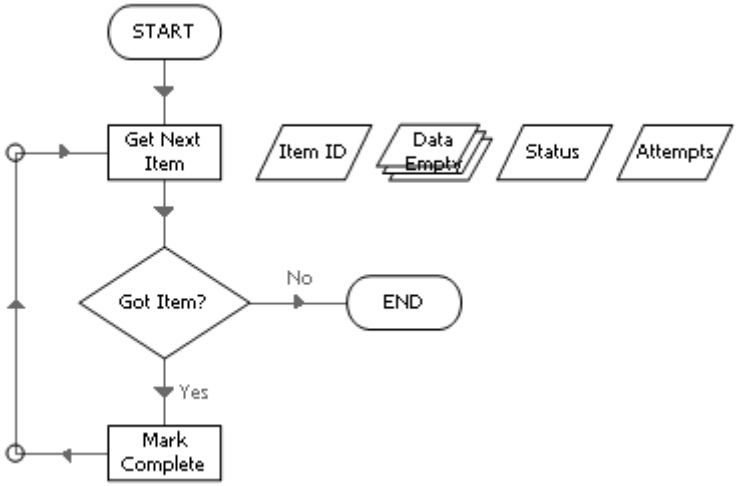


Figure 136: Anchor Stage

- Link from the Anchor stage to the End stage so that your diagram looks something like this:



As you can see, the process contains a linked loop that returns the flow back up to Get Next Item after Mark Complete. To prevent the process going around this cycle forever (an infinite loop), we have used a Decision to stop when the queue has no more items available.

The Decision expression **[Item ID]<>""** says in essence, "Have we got an item ID?" and if the answer is "Yes", we know we have an item to work with. When the answer is "No", we know there are no more items left to work and we redirect the flow to the end point.

- Arrange the desktop so you can see Process Studio and Queue Management.
- Run the process slowly and watch how it flows around the loop. Use the Refresh button, , on Queue Management to see the item statuses change.

> ✱ *Tip: The Queue Management tree item in Control Room does not update automatically – refresh to see the latest picture.*

Best Practice

- A ***Get Next Item*** Action should almost invariably be followed by some sort of ***Got Item?*** Decision.
- An expression checking if the ID is blank is the best way to test if you have got an item.

9.3. Work Queue Configuration

Work queues are created in System Manager, the administrative area of Blue Prism.

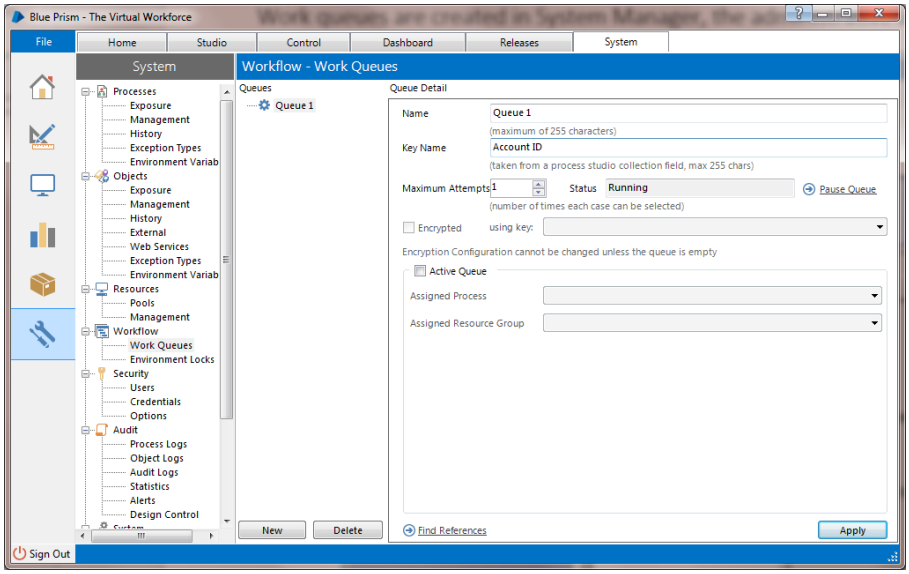


Figure 137: Work Queue Configuration

Exercise 9.3.1 Creating a Queue

- Go to System Manager and look at the Work Queue Configuration tab in the Workflow section.
- Create a new queue named **Queue2** and name the Key Name **Staff ID**. Return to Control Room to see the new queue.

  - *Tip: Remember to use the Apply button to confirm your changes.*
  - *Note: The Encrypted option is available to ensure any sensitive data held within the queue is encrypted in the Blue Prism database. Leave this option un-checked for this exercise.*

Exercise 9.3.2 Populating a Queue

Modify the Queue Exercises 2 Process so that it can populate the new queue with five new items.

- Reset the Queue Exercises 2 Process and then inspect the properties of the **New Items** collection.
- Remove all the existing fields from the collection and then add the following fields. (Take care to spell "Staff ID" exactly as you did in the previous exercise.)

Name	Data type
Staff ID	Number
Surname	Text
Forename	Text

DOB	Date
Full Time	Flag

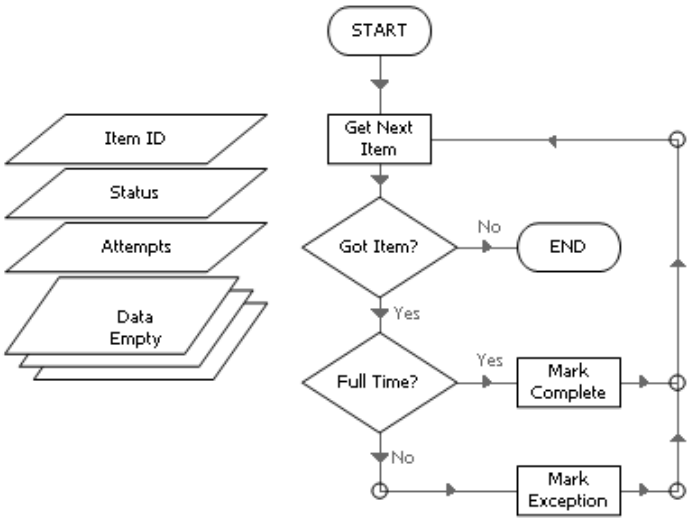
- Add five items to the initial values.
- Run the Process and check that your new queue has been populated. Note that the Staff ID values are in the Item Key column.

Exercise 9.3.3 Working a Queue

- Modify the **Queue Exercises 1** to use the new queue. Mark the item as complete if the employee is full-time and mark part-timers as exceptions.

  * *Tip: Remember to use dot notation for Collections in expressions e.g., **[Data.Full Time]**.*

- Your diagram might look something like this:



Exercise 9.3.4 Creating a Narrative

We shall repeat the previous exercise but record some additional detail as we work each case.

- Open **Queue Exercises 2** and add a new field to the Collection, name **Narrative**, data type **Text**. Leave the Narrative values blank.
- Run the process put some new items into Queue2, then save and close.
- Add Note stages to represent some imaginary steps into your Process – e.g.,"Locate Staff Member", "Open Staff Record", and "Read Working Hours".

Figure 138: Note Stage Toolbar Button

Like the Anchor, a Note is an inert stage that has no effect on a Process or Business Object. Its purpose is to annotate diagrams.

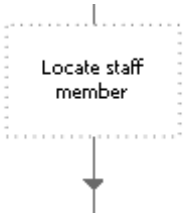


Figure 139: Note Stage

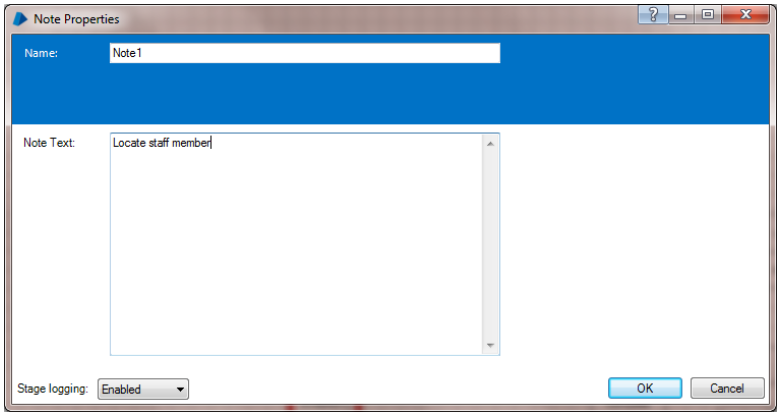


Figure 140: Note Stage Properties

- After each Note, use a Calculation stage to modify the **Narrative** field in the Data collection.

  * *Tip: Think carefully about your Calculation stage. Would you prefer to replace the narrative or append to it?*

- After each Calculation add an Action using the **Set Data** from the **Internal - Work Queues** Business Object to commit the change back to the queue.

  * *Tip: Think about how collecting this information could save your Process some time and think about who else might use the information and how.*

Key Points

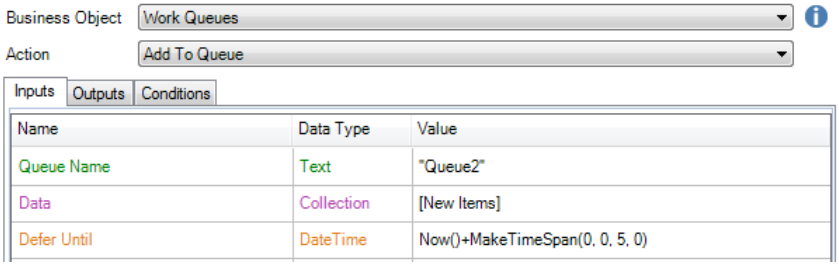
- The item data Collection can be updated to save information back to the queue.
- Providing empty fields (like Narrative) when the item is created could provide storage for data collected while the item is being worked.

9.4. Defer

New items can be deferred to prevent them from being worked too soon. If a deferral date is specified when items are created, the queue will hold on to them until that date. Effectively the items are temporarily ***frozen***.

Exercise 9.4.1 Deferring New Items

- Open **Queue Exercises 2** and add new items to **Queue2** but this time set the **Defer Until** input to a future date.



- Return to **Queue Exercises 1** and notice how Get Next Item returns nothing, as if the queue was empty. The queue will not release these items until the deferral date arrives.

Key Point

- If Get Next Item fails to get an item, it does not necessarily mean there are no unworked items in the queue; it could be that there are ***deferred*** items the queue is yet to release.

Exercise 9.4.2 Manually Overriding the Defer Date

It is possible to change an item's deferral date from Control Room.

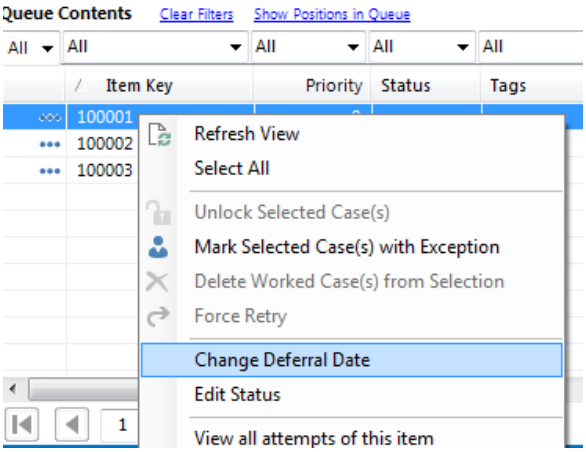
- Go to Queue Management in Control Room and select one of your deferred items.
- Right-click, select Change Deferral Date, and change the date to a few seconds into the future (you can't change it to the past).

Exercise 9.4.3 Deferring a Working Item

Defer can also be used to release an item got from the queue without applying a "complete" or "exception" result. This feature is useful for part-working an item and then "freezing" it for later - perhaps in a Process that spans two days where items are "half" worked today and then deferred until tomorrow.

- Return to **Queue Exercises 1** and change the properties of the "Mark Complete" Action stage to "Defer". Use a similar expression as before for the **Until** input parameter.
- Step though the Process and go back to Queue Management to see how after "getting" an item it can be released again.
- Once you are satisfied with how Defer works, undo your changes to return the Action stage to "Mark Complete" again.

> ✱ *Tip: Using **Now()** as the expression will make the item instantly available.*



- Return to **Queue Exercises 1** and confirm that items are available to be worked.

9.5. Attempts

Items can be worked more than once if necessary. The ***Max Attempts*** field on the Work Queue Configuration section of System Manager is set to 1 by default but this can be set to a higher value if necessary.

When Max Attempts is greater than 1 the work queue will revive any exception cases by ***cloning*** a new item and inserting it into the queue. The following series of tables will help to demonstrate the effect.

Exercise 9.5.1 Understanding Queue Attempts

A queue has been set up with Max Attempts=2 and there are four items in the queue. The first item has been completed but ABC-002 has just been marked as an exception.

Status	Item Key
▪▪▪	ABC-004
▪▪▪	ABC-003
⚑	ABC-002
✔	ABC-001

The work queue sees that ABC-002 has attempts remaining and inserts a new cloned item into the queue ***before*** ABC-003. This cloned item will be the next available item, not ABC-003.

Status	Item Key
▪▪▪	ABC-004
▪▪▪	ABC-003
▪▪▪	ABC-002
⚑	ABC-002
✔	ABC-001

ABC-002 is worked again and marked as an exception again. The queue sees that no more attempts at ABC-002 are allowed (Max Attempts=2) and does not create a new item. The Process moves on to the next item, ABC-003.

Status	Item Key
	ABC-004
	ABC-003
	ABC-002
	ABC-002
	ABC-001

Overriding Max Attempts

The ***Retry*** input parameter of the Mark Exception action in the ***Internal - Work Queues*** Business Object can then be used to ***override*** Max Attempts.

When Retry is **True**, the queue will generate another attempt if the limit of Max Attempts has not been reached. But when Retry is **False**, it will override Max Attempts and prevent any further attempts regardless of the number of retries available.

The ID of the new item is supplied as an output from Mark Exception.

Keeping hold of the new item

The ***Keep Locked*** input parameter works in conjunction with the Retry input. When Keep Locked is **True**, the new cloned item will become instantly locked. This provides the opportunity to carry on working on the new item. When Keep Locked **False**, any new item created is freely available as the "next" item.

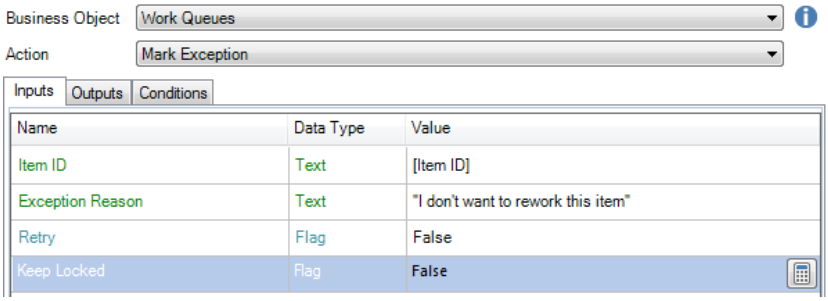


Figure 141: Retrying an Exception Item

Exercise 9.5.2 Retrying Items

- Modify Max Attempts in the **Queue2** definition and experiment with retrying a queue item.

Key Point

- A queue can be set up to enable multiple attempts on an item.
- Attempts only apply to exceptions – items marked as complete cannot be reworked.
- The creation of a retry item can be cancelled from the diagram.

9.6. Pause and Resume

A work queue can be temporarily paused so that it will not release items at all. The ***Resume*** function reactivates a paused queue.

From the perspective of a Process, a paused queue will appear devoid of available items. This feature can be a useful way of making a Process "think" it has no work to do.

Exercise 9.6.1 Pausing a Queue

- Go to the list of queues in Queue Management and right-click on a queue. Experiment with Pause and Resume to see the effect.

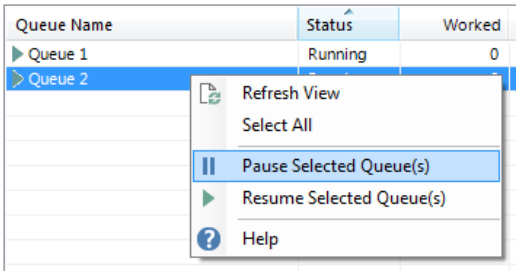


Figure 142: Pausing a Queue

9.7. Filters

The Queue Contents list can be filtered to display a subset of the queue. Drop-down lists above each column are used to apply a filter, and a filtered view can be saved and reapplied as required.

9.8. Reports

The Queue Contents list can be exported as a basic report.

Exercise 9.8.1 Reporting on a Queue

- Right-click on a queue item in the lower half of Queue Management.
- Select **Export current view as a report** and follow the subsequent steps to create a report file.
- Experiment by first adjusting the drop-down filters before exporting to create different reports.
    - *These extracts are intentionally "raw"; there are other ways to create reports that you will learn about as part of a mentoring program after this course.*

9.9. Review

- A queue is used by a Process to manage a workload.
- Work is stored as a list of individual items.
- A special "internal" Business Object is provided to work with queue items.
- Each item contains a Collection of data to be used by the Process.
- When an item is being worked, it is locked to ensure it cannot be worked elsewhere.
- After being worked, an item is either marked as complete ( ) or as an exception ( ).
- A queue can be configured so that exception items can be worked again.
- A queue can be temporarily deactivated and reactivated.
- A report of queue items can be created.

Blue Prism Keywords

Work Queue, Queue Item, System Manager, Key, Pause, Resume, Defer, Retry, Attempts.

10. Additional Features

This section deals with some of the aspects of Blue Prism we have not yet looked at.

10.1. Safe Stop

When a process is running in Control Room you can stop the process by selecting the process session and clicking **Stop Selection.** This will stop the process immediately. The case currently being worked will not be completed and Blue Prism will mark it as an Exception at Clean Up.

Figure 143: Control Room Session Controls

If you right click a running process session you will see the following stop options; Immediate Stop and Request Stop.

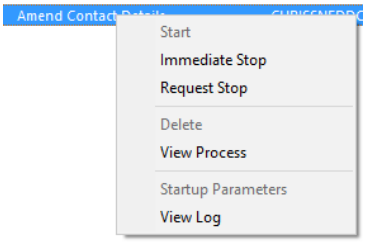


Figure 144: Context Menu for Running Processes

Immediate Stop is the equivalent of Stop Selection. Request Stop will ask the process to stop where there is a configured safe stop with the process.

You can configure a safe stop within your process by using the inbuilt Environment function IsStopRequested() within a Decision stage.

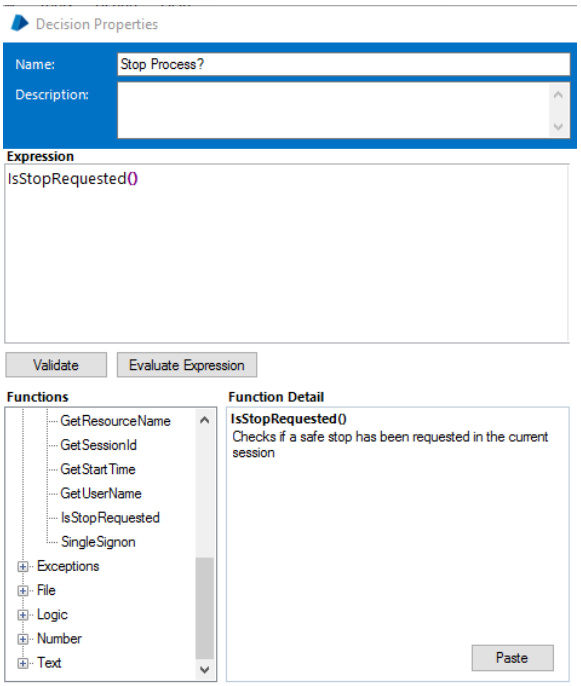


Figure 145: Decision for Safe Stop Request

Typically you would place this Decision stage prior to getting the next case from a Work Queue. This will allow the process to complete the current case before detecting a safe stop request and processing accordingly.

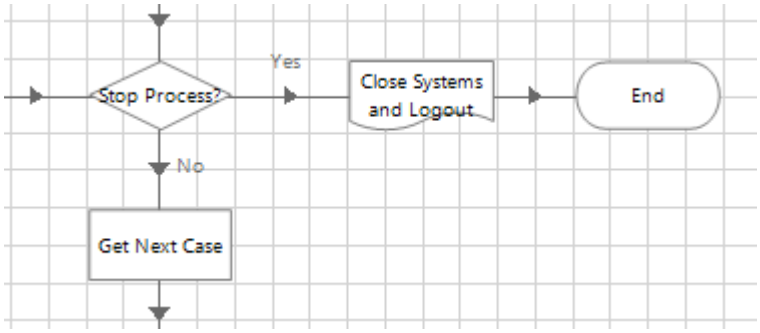


Figure 146: Process Checking for a Safe Stop Request

10.2. Collection Actions

We have seen how to create a Collection and how to use a loop to access each row. The special ***Internal - Collections*** Business Object has other actions that we have not used yet, most of which are self-explanatory.

- Add Row.
- Remove Row.
- Count Rows.
- Count Columns.
- Remove All Rows.
- Copy Rows. Copy Rows is used to generate a new Collection from a subset from another Collection

Key Point

- The inputs of the ***Internal – Work Queues*** actions ask for the ***name*** of the collection, rather than the collection itself. The data type of the "Collection Name" input is Text, not Collection.

10.3. Choice Stage

Like the Decision stage, the choice stage is based on expressions that result in either **True** or **False**. The difference is that a choice stage can evaluate a series of expressions rather than just one.

The choice stage looks a little like a Wait stage and works in a similar way, where each expression creates a separate branch. Expressions are evaluated in top-down order, and the first one found to be **True** takes the flow down that branch. The bottom Otherwise branch is taken if none of the expressions are **True**, much like the timeout end of a Wait stage.

Figure 147: Choice Stage Toolbar Button

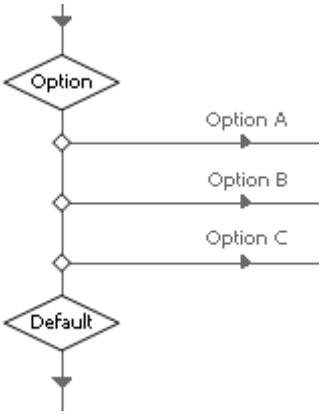


Figure 148: Choice Stage

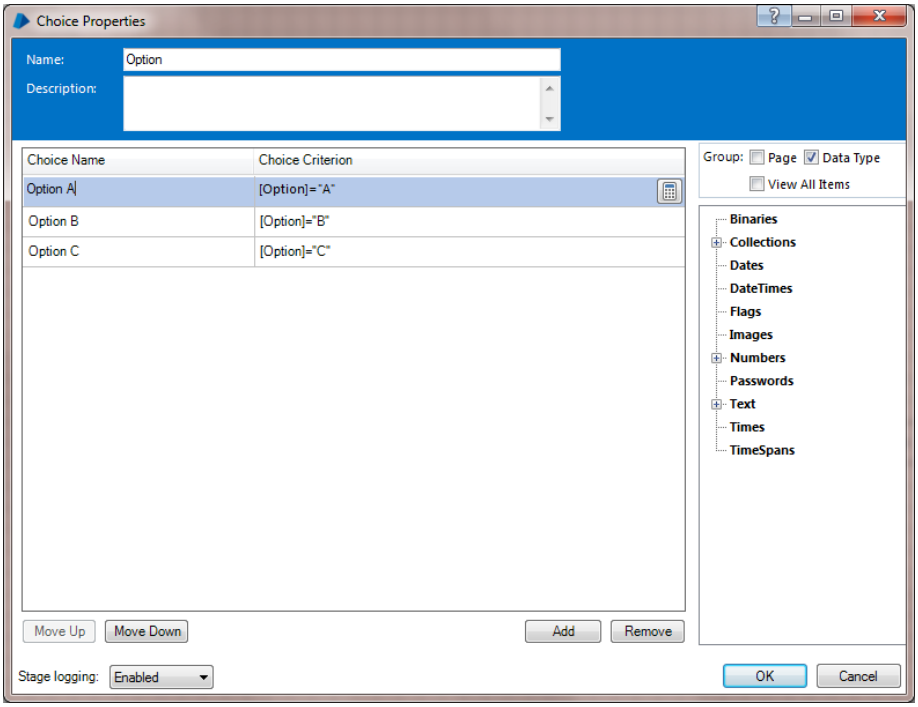


Figure 149: Choice Stage Properties

10.4. Logging

Blue Prism can log the details of stages used in Processes and Business Objects. By default, stages used in Process Studio are logged and those in Object Studio are not. This can easily be changed by modifying the properties of a stage, selection of stages or all stages.

Nearly all stage properties windows have a combo box with the stage logging options; Enabled, Disabled and Errors Only. Errors only will log the stage only if an error is thrown when the stage runs.

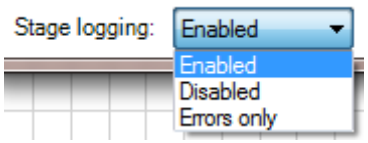


Figure 150: Stage Logging Options

Alternatively, the menu commands ***Edit > Selected Stages >*** and ***Edit > All Stages*** can be used to modify the stage logging of more than one stage at a time.

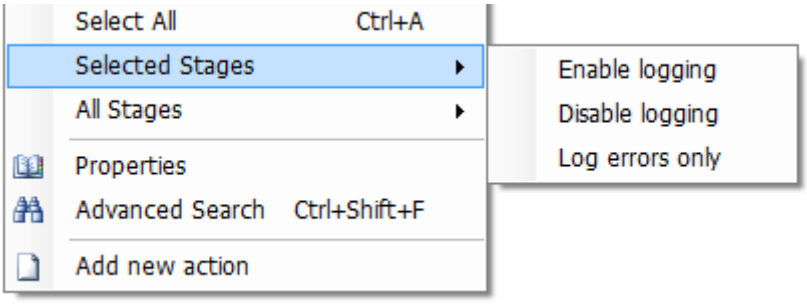


Figure 151: Logging Menu

Page information stages are not logged and Data Items are only logged when their values are used as inputs and outputs.

Key Points

- The decision as to which stages are logged and which are not should not be overlooked.
- Logs are particularly useful during a test phase as a means to investigate bugs.
- However a live Process running all day can put a vast amount of logs into the database, the maintenance and back up of which should be considered.
- The security or legal implications of storing sensitive data are also something to think about.

10.5. Log Viewer

The Log Viewer is used to inspect the log of a session, either as it is running or after it has finished. This is especially handy for reviewing the workings of a progress and tracking down problems.

The Log Viewer has a search function enabling you to look through the (often numerous) rows of a log and the visibility of columns can be configured to suit.

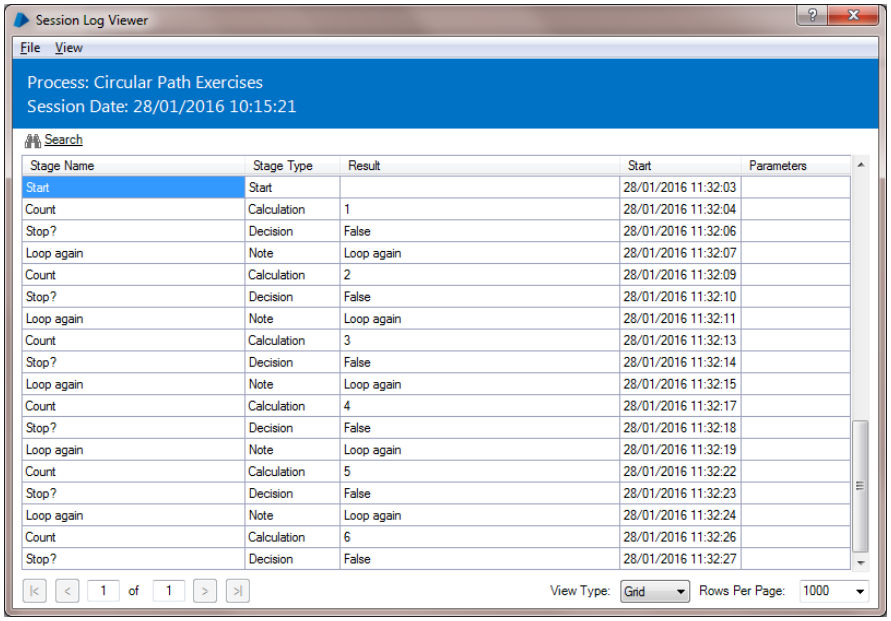


Figure 152: Log Viewer

10.6. System Manager

As the name suggests, System Manager is where Blue Prism's own settings are kept. Some of the tasks that can be performed are as follows:

- Create and maintain user accounts, roles, and permissions.
- Retire resources, Processes, and Business Objects.
- View Process or Business Object change history.
- Manage web services.
- Manage work queues.

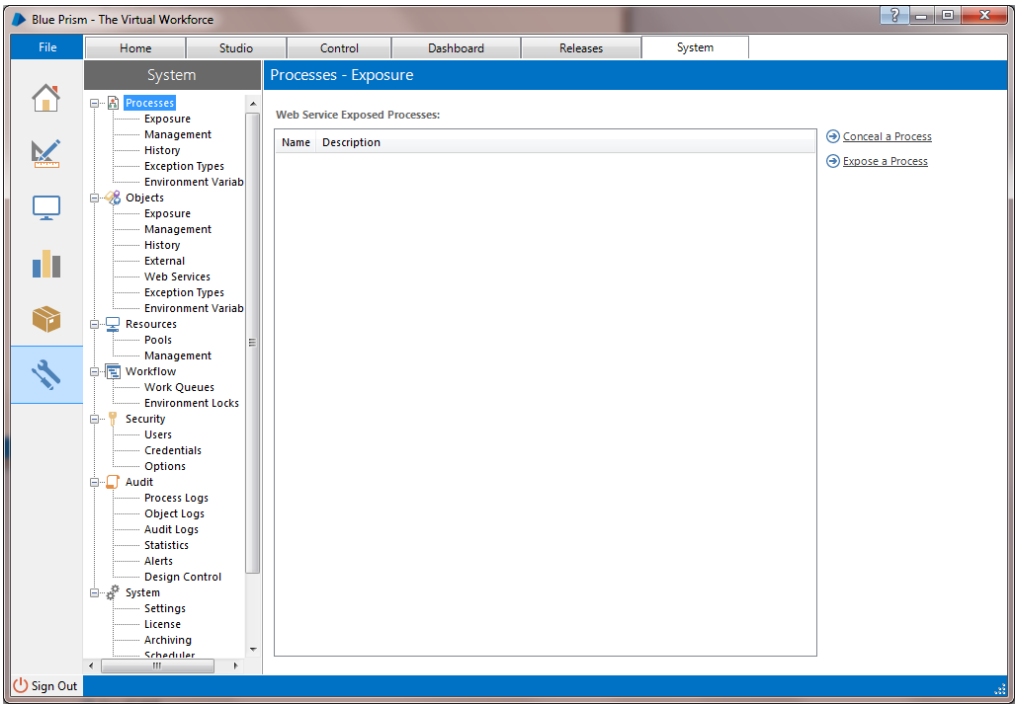


Figure 153: System Manager

10.7. Process/Business Object Grouping

Processes and Business Objects can be arranged into groups in order to simplify and tidy the lists used in the Blue Prism application. Groupings have no effect on the Processes or Objects themselves. They are moved to and from groups by using "drag and drop" in System Manager. You can create groups by right clicking on either Processes or Objects in the Studio tab.

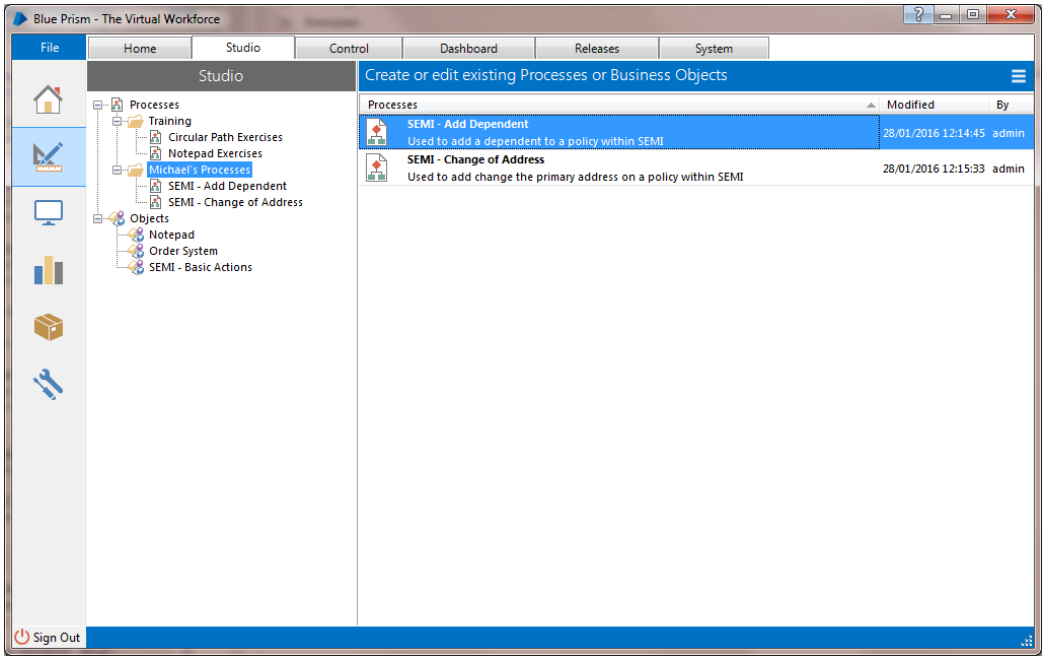


Figure 154: Process Grouping

10.8. Process and Object References

You can view what Processes and Business Object are using a particular Process or Business Object by finding references. To find references of a select Process or Business Object in Studio and click the hamburger icon at the top right and select **Find References**

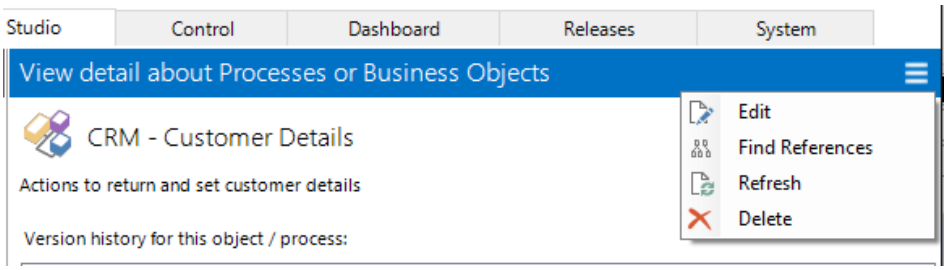


Figure 155: Use the hamburger menu to Find References

This will return a list of where the Process/Business Object is used.

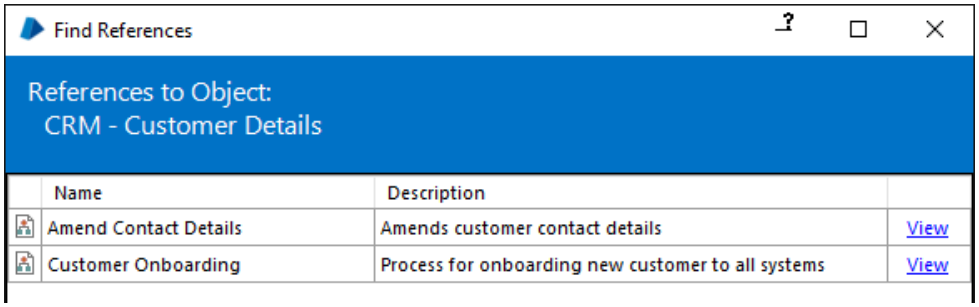


Figure 156: List of found references

10.9. Export and Import

A Process or Business Object can be exported as an XML file, and similarly an exported file can be imported into Blue Prism. This can be useful for backing up and moving work between databases.

A wizard is provided to guide the user through each step of the procedure.

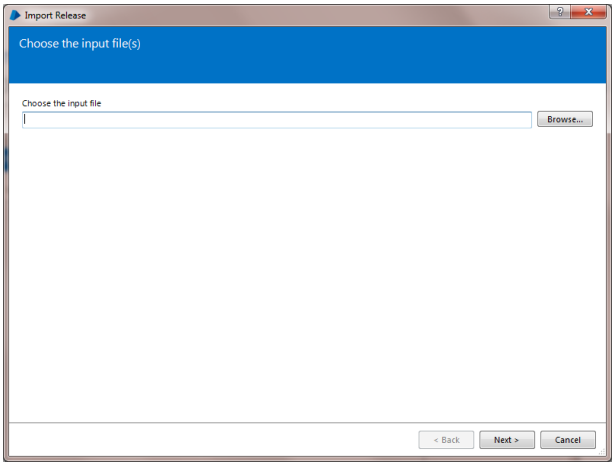


Figure 157: Import Wizard

Exercise 10.7.1 Exporting and Importing

- From the main Blue Prism application window, select **File > Export** from the main menu.
- Experiment with exporting a Process as an XML file.
- Switch to a different database, if you like. Alternatively, you may simply imagine that you have done so. Choose **File > Import** and locate an XML file.

> ✱ *Tip: Processes and Business Object names must be unique.*

10.10. Release Manager – Packages and Releases

The above XML import/export is useful for one-off ad hoc migrations of specific Business Objects and Processes, but in the real world a Process might have several dependencies. For example, a Process might employ a range other Processes, Business Objects, and Work Queues.

To migrate all of these parts individually would be an error-prone task requiring careful attention to detail and conscientious use of checklists. For this reason, Blue Prism provides the ***Release Manager*** which allows users to create checklists known as ***Packages***.

At various stages during the development cycle, a ***Release*** can be exported using the checklist provided in the Package. Two different releases produced on two different occasions could contain different versions of the Processes/Business Objects/work queues, etc., but because they have come from the same Package (i.e., the same checklist) they will contain the same items.

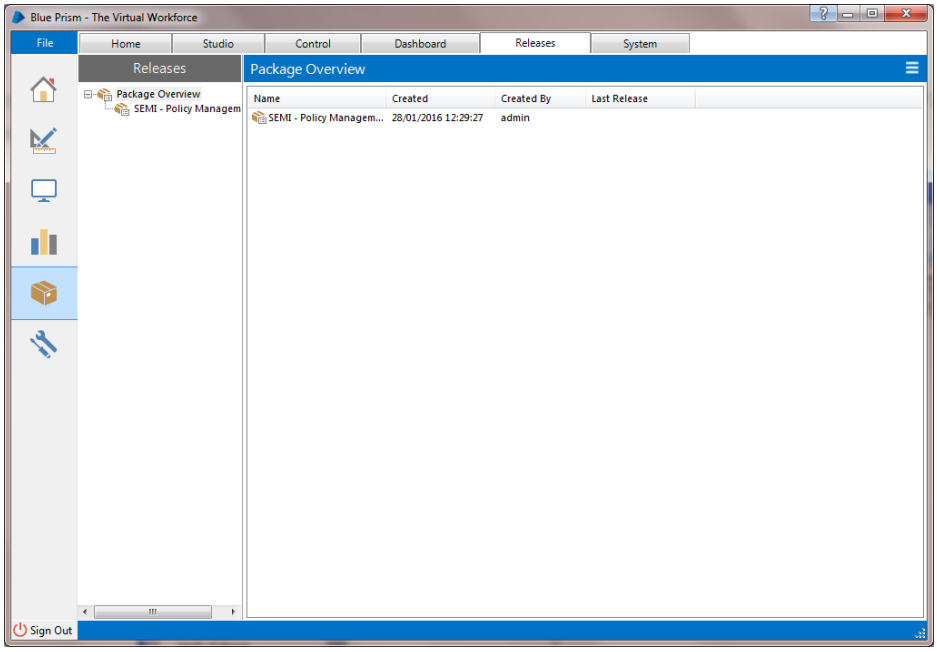


Figure 158: The Blue Prism Release Manager

Exercise 10.8.1 Creating a Package

- Go to Release Manager and click **New Package**.
- Complete the wizard, choosing a combination of items to include in the package. For this exercise, it is not important what items you pick.

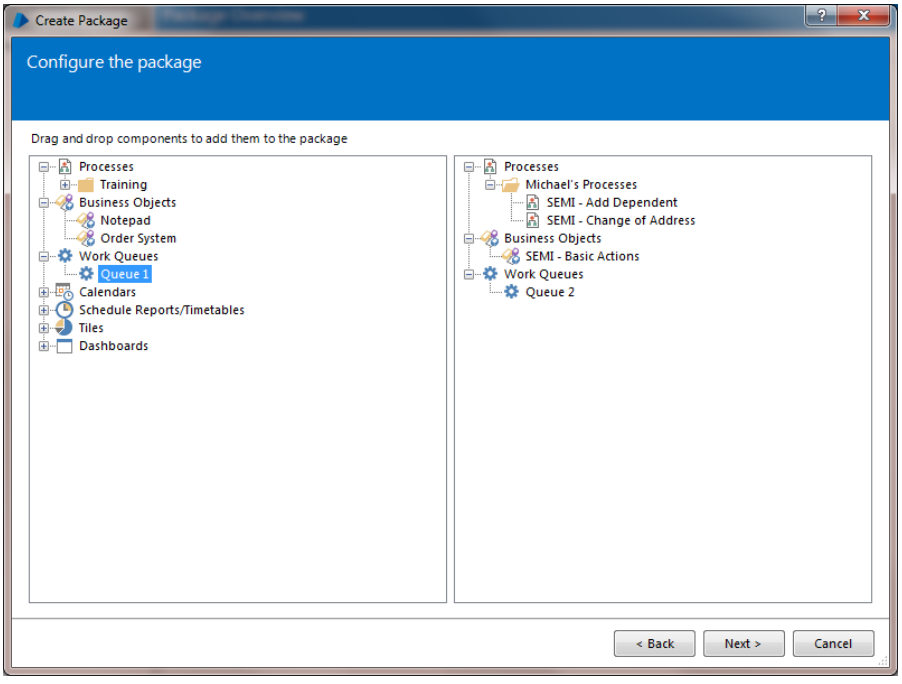


Figure 159: Create a Package Wizard

Exercise 10.8.2 Exporting a Release

Now that a Package has been created, we can create a ***Release***. The Release will be a snapshot of the state of each of the items in the Package at a particular moment in time.

- Select the Package you just created and click **New Release**.
- Choose a name for the release and provide some release notes (e.g., “Development complete. Process now ready for UAT.”).
- Choose a location to save the file; the file extension will be **.bprelease**

The Release file contains the details of all the items in the package and could be used to import the items into another database. Typically this would mean a new environment, e.g., exporting a Release from the Development environment and importing into a UAT environment.

Key Points

- Blue Prism creates different environments by using separate databases. Typically this would be three databases - Development, Test, and Live.
- Moving work from one database to another is done by exporting and importing.
- An exported file can also be attached to email and used as a basic back up.

11. Consolidation Exercise

Now we have learnt the basics of Blue Prism configuration, let's put this into practice by building and running a Process end-to-end.

- To reach a standard recognized by Blue Prism that will allow you to progress to the next stage of your accreditation, you must complete the exercises in this chapter to Blue Prism's satisfaction. For those that are self-learning, at the end of the chapter you will be provided with details of how to notify Blue Prism that your completed process is ready for review.

11.1. Order System Process

The following exercises will guide you towards creating a Process that can place orders in the Order System application. The exercises are deliberately less explanatory to force you to think about what you have learnt so far.

The basic steps the Process will take are as follows:

- Read a CSV file containing order information.
- Load the information into a queue.
- Start Order System.
- Work through the queue items creating orders.
- Close Order System.

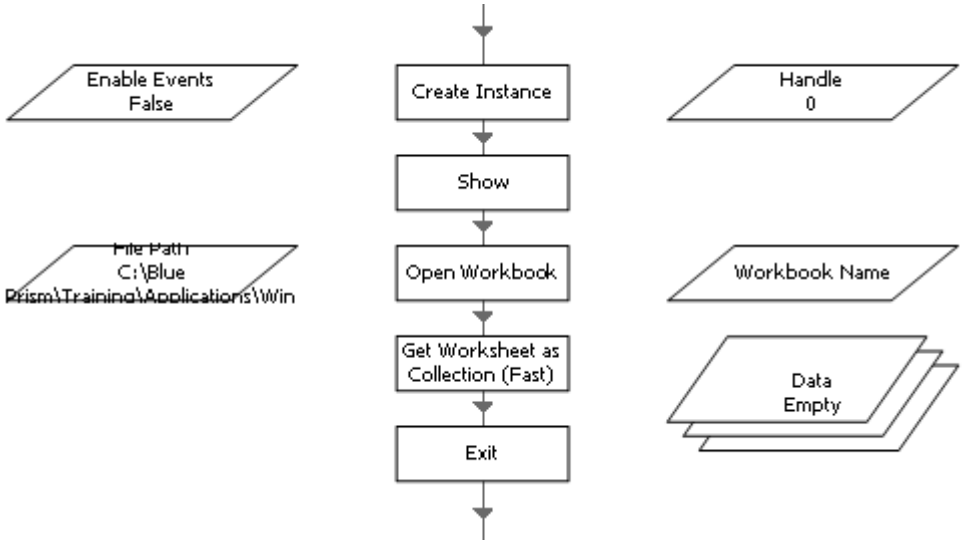
Exercise 11.1.1 Reading from Excel

There is an Excel Business Object available that you can use to read a CSV file, and there is a ready-made file you can use too.

Blue Prism ships a number of standard business objects that can be imported into your object library. These objects cover a range of interfaces and utilities and can be found in C:\Program Files\Blue Prism Limited\Blue Prism Automate\VBO.

To install the Excel interface object, click File – Import and then select the BPA Object - MS Excel.xml file from the above VBO directory.

- The Business Object is named **MS Excel VBO**.
- The file is named Orders.csv and is in C:\BluePrism\Training\Applications\Windows.
- The sequence of Actions is as follows:
- **Create Instance** to start a session of Excel.
- **Show** to make it visible (it's invisible by default).
- **Open Workbook** to open the file.
- Get Worksheet as Collection (Fast) to read the data.
- **Exit** to close Excel.
- Begin a new Process named **Create Orders.** It should look something like this:



- Test that the process can open the CSV file and retrieve the data. If successful, the Data collection will no longer be empty.

Exercise 11.1.2 Loading the Queue

Continue working on your ***Create Orders*** Process so that it will load the CSV data into a queue.

- Make a new Work Queue.
- Use the **Internal – Work Queues** Business Object to put the CSV data into the queue.
- Test your process. Is the data being written to the work queue? Remember the Work Queue can be viewed in Control Room.

Exercise 11.1.3 Working the Items

Continue working on your ***Create Orders*** Process so that it will work through the queue. For now we are simply pretending to work by simulating how the work queue functionality.

- Have your process get the next Pending item from the queue (you'll need to create some data items for the outputs).
- Check to see if a case has been returned
- Mark a returned item in the queue as complete (see Working All Items).
- Loop back to get the next Pending item and repeat until no item is returned and the process can end.

> * *Tip: The file only has five orders in it so don't worry about importing the file and "processing" the orders more than once.*

Exercise 11.1.4 Launching Order System

Continue working on your ***Create Orders*** Process so that it will launch and close the Order System.

- Launch and Log into the Order System soon after the start of the Process. Perform this on the Main Page.
- Close the Order System just before the end of the Process i.e., when no more cases are returned from the work queue.
- Exception Handling. It is important that if we can't successfully start the Order System we DON'T proceed with the process. Create a block around the order system launch and login sequence and use a recover

stage to catch any exception thrown. If we then re-throw this exception, the process will fail as we are throwing an exception on the Main Page.

- *Tip: It would be inefficient to launch and close for every item.*
- *Tip: Remember the purpose of the "Preserve" check box in the Exception stage properties.*

Exercise 11.1.5 Going Further into Order System

Let's now look how a user would submit an order before we automate the steps:

- Log into Order System using your Business Object.
- Familiarize yourself with the application by **manually** inputting a few orders.
  - *Tip: You can use whichever data you like as long as all fields are completed.*

Exercise 11.1.6 Updating the Order System Business Object

Once you understand the steps required to put an order into Order System, update your Business Object. We will be adding three new actions (pages) to your object.

Below is a summary of the pages your Business Object will need:

Action	Inputs	Outputs	Precondition	Post Condition
Navigate Menu	Option number		Options Menu	
Input Order	Product Code Number Required Price Cost Center		New Order Page	Order Confirmation screen has opened
Order Confirmation		Order Reference Number	Order Confirmation screen	Order Confirmation screen has closed

When creating elements in Application Modeler, remember to use the tree structure and group elements by the screen or section they are in. This will make locating elements in a busy Application Modeler much easier.

When building and testing your solution, you may find that some elements can no longer be found by Application Modeler.

- Remember when spying new elements to use Highlight Element and adjust the element attributes accordingly to make the element unique.
- Check the Window Text attribute. Take care matching on this attribute as the value may change as the Business Object runs.

Exercise 11.1.7 Action – Navigate Menu

- Add a new page to navigate the option menu named "Navigate Menu".

- Set the input parameter.
- Our pre-condition is that we are on the Options Menu. When the action starts wait for an element that will be available on the Options Menu. Set a timeout period and throw a System Exception if the timeout expires.
- Use the input parameter to set the option and click "GO".
- Your action will need to be **published** if it is to be visible to a Process.

Exercise 11.1.8 Action – Input Order

- Add a new page named "Input Order".
- Set the input parameters.
- Our pre-condition is that we are on the Order Page. When the action starts, wait for an element that will be available on the Order Page. Set a timeout period and throw a System Exception if the timeout expires.
- Use the input parameters to create the order and click "Submit Order".
    - *Tip: Remember there is no data validation in Order System. When building your page, use fictitious data as current values in your Data Items to test your solution.*
- Our post-condition is that we are on the Order Confirmation screen. Once the order has been submitted, wait for an element that will be available on the Order Confirmation screen. Set a timeout period and throw a System Exception if the timeout expires. You may need a long timeout here to absorb any system latency. We can throw a System Exception here because we know the training system allows any data. In reality, we would probably throw a Business Exception and determine the exception reason from the application error message e.g., invalid product code, product out of stock, etc.
- Your action will need to be **published** if it is to be visible to a Process.

Exercise 11.1.9 Action – Order Confirmation

- Add a new page to navigate the option menu named "Order Confirmation".
- Set the output parameter.
- Our pre-condition is that we are on the Order Confirmation screen. When the action starts, wait for an element that will be available on the Order Confirmation screen. Set a timeout period and throw a System Exception if the timeout expires. This is unlikely as we are waiting for the screen in the previous action. However, it is best practice to wait at the start of the action and confirm the screen you are on.
- Retrieve the order number from the screen and click "Continue".
    - *Tip: Use the Text functions in a Calculation stage to extract the number from the confirmation text.*
- The post-condition is that the Order Confirmation screen has closed. Add a wait stage to wait for the screen to close.
    - *Tip: We waited at the start of the action for the Order Confirmation screen. Simply reverse this logic. i.e., check exists = False.*
- Your action will need to be **published** if it is to be visible to a Process.

Exercise 11.1.10 Updating the Create Orders Process

Return to your Create Orders Process and make use of the new actions in the Business Object. Enable your Process to perform the following steps:

1. Add the data in the CSV file to the work queue
2. Log into Order System
3. Get the next item from the queue. We must work the items by retrieving them one at a time from the queue. We DO NOT work the items by referencing the collection retrieved from the CSV.
4. Navigate to the New Order screen
5. Submit the order
6. Get the Order Reference Number
7. Mark the queue item as complete
8. Repeat steps 3-7 for all items
9. Close Order System

Exercise 11.1.11 Case Exception Handling

We have already added some exception handling to the process to catch any exception where the process failed to launch and login to the Order System. In this instance, we throw an exception from the Main Page to fail the process as there is no point in continuing.

Now, we want to handle case exceptions. If for any reason an exception occurs when processing a case, we must set that case to an exception and move onto the next.

- Add a block around the following steps:
- Navigate to the New Order screen.
- Submit the order.
- Get the Order Reference Number.
- Have a Recover Stage within the block catch the exception.
- From the Recover Stage make the case an exception before continuing with the next case.
- When making a case an exception always set the exception reason. In this instance we want the reason to be the exception detail bubbled up from the action.
    - *Remember to use a Resume Stage before we continue with the next case.*

Exercise 11.1.12 Testing the Process

- Step through the Process in Process Studio and Object Studio to satisfy yourself that the Process is working correctly.
- Practice using Step, Step Over, and Step Out.
- Use the Go button the play the Process at different speeds.
- Experiment with Breakpoints.

Exercise 11.1.13 Improving Resilience

The Order System requires that all data fields are completed when inputting an order. What happens if you omit a field?

- Improve the resilience of your Process by ensuring that such cases are marked as exceptions **without** the Process failing.

- After you retrieve the next case, validate that the required data are present before navigating to the Create Order Page.
- If data are missing, make the case an exception and carry on.

Exercise 11.1.14 Running in Control Room

- Run the Process in Control Room to see it run end-to-end.
- You must do this five times without the process failing. If the process fails, investigate why and apply a fix.

> *Tip: Remember the Process will need to be published before it can run in Control Room.*

Exercise 11.1.15 Analyze the Session Logs

- Recall that we created a page in the Business Object to identify the order reference number and pass it back to the Process as an **output parameter**.
- Although we didn't use this value in the Process, you should be able to see it in the session logs.

Exercise 11.1.16 Package the Process

- Create a package for the Process and its dependencies ready for deployment to an alternative environment.

11.2. Consolidation Exercise Checklist

Before submitting your solution make sure you have completed the following:

Create Orders Process

- Exception handling is in place for application start-up and login. ☐
- Exception handling is in place for case processing. ☐
- Process checks for required fields prior to entry into the system. ☐
- Process makes the case an exception where required data are missing ☐

Order System Object

- Application Modeler logically laid out. ☐
- Navigate Menu action created. ☐
- Wait stage at start and exception thrown on timeout. ☐
- Input Order action created. ☐
- Wait stage at start and exception thrown on timeout ☐
- Order Confirmation action created. ☐
- Wait stage at start and exception thrown on timeout. ☐
- Action extracts and returns the order reference number. ☐
- Wait stage at end to confirm Order Confirmation screen has closed. ☐

Control Room

- Process has been run in Control Room a minimum of five consecutive times without failure. ☐
- The order reference number is visible in the session logs. ☐

Release Manager

- Package and release have been created. ☐

11.3. Submitting Your Completed Solution

Online Training

If you are taking the Foundation Training on one of the Blue Prism remote training environments via your browser, please email support@BluePrism.com with the name of your Order System process. A member of Blue Prism will access your training environment to assess your work.

Distributed Training

If your Foundation Training was setup by your own organization and you are *not* taking the training via an internet browser, please consult your local Blue Prism development lead.

While your work is being reviewed, please continue with the rest of the course to learn about the more advanced features of Blue Prism.

12. Advanced Features

12.1. Undefined Collections

Until recently, the Collections we have used have been ***defined***, meaning that their column names and data types have been set up by us in advance.

When using a Collection to receive an output parameter, it is possible to leave the Collection ***undefined***, i.e., with no columns set up at all. You may have noticed that we have already done this in ***Exercise 9.1.1***.

Exercise 12.1.1 Queue Item Data Collection

- Go back to the **Queue Exercises 1** Process and step through to the Get Next Item action. Look at the properties of the Collection used for the output parameter. Is there anything odd about it?

  ✱ *Tip: Look at the Initial Values tab and then at the Current Values tab.*

The Collection is undefined, i.e., it has no columns to begin with. It only receives its definition when the Process runs the Get Next Item action.

Key Point

- Collections are the only kind of Data Item that can behave in this way. All other Data Items must be predefined with a data type.

12.2. Data Item Initialization

We have looked at how a Data Item can be given an initial value, and that this default value will also be used as the current value when running starts. By default a Data Item is only visible from its own page, but un-checking the Visibility check box on the properties form will make the Data Item global.

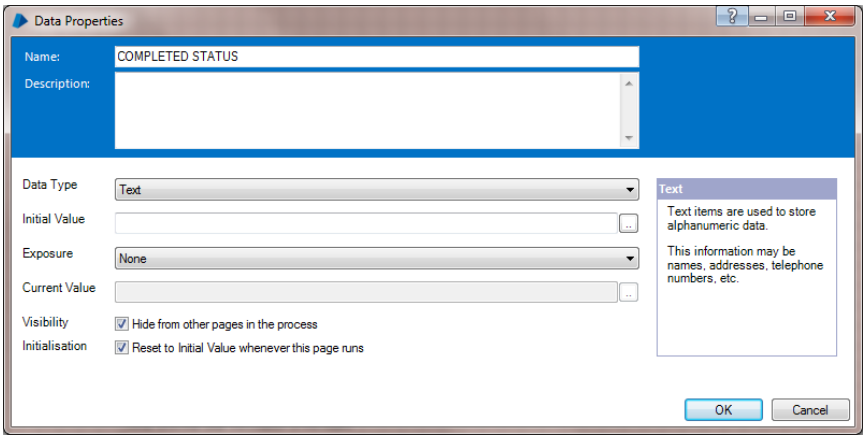


Figure 160: Data Item Properties

Just below the Visibility checkbox is another checkbox named “Initialization”, and it controls how the current value of a Data Item is affected when its page is run.

By default, the current value is reset to the initial value when the start stage of the page is reached. When the checkbox is un-checked, the current value is maintained and not reset. Both settings have their advantages, depending on the circumstances.

For example, global Data Items can be placed on a page that simply links the start stage to an End stage. Although the page will appear to do nothing, it can be used to ensure that the global Data Items are put back to their initial values simply by running the “Reset” page.

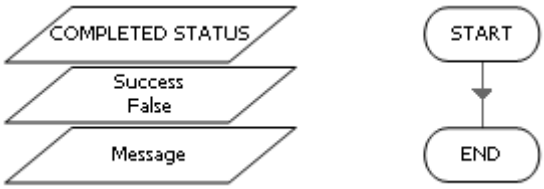


Figure 161: Resetting Global Data Items

12.3. Data Item Exposure

Data Items can have their values controlled from ***outside*** the diagram - this is done by ***exposing*** a Data Item. By default, Data Items are not exposed but we will look at how that can be changed.

Environment Variables

Exercise 12.3.1 Using Environment Variables

An Environment Variable is a value that is made available to all Processes and Business Objects, i.e., across the environment.

In order for the following exercise to work, we will need to import a Business Object that allows the process to use a "Sleep" action. Following the action we carried out in the earlier exercise to import the Excel Object, we will need to install the BPA Object - Utility - General.

To install the Utility - General object, go to Blue Prism and click File – Import and then select the BPA Object - Utility - General.xml file from the VBO directory.

- The Business Object is named Utility – General.
- The object is in C:\Program Files\Blue Prism Limited\Blue Prism Automate\VBO.
- Go to System Manager and look at the Environment Variables tab in the Processes section.
- Add a new Environment Variable named **Stopping Time** with data type **Time** and a value of three or four minutes from now.
    - *Tip: Remember to click the Apply button.*
- Create a new Process named **Stopping Test**.
- Create a Data Item named **Stopping Time**, and use the Exposure field to link this to the Environment Variable, as illustrated.

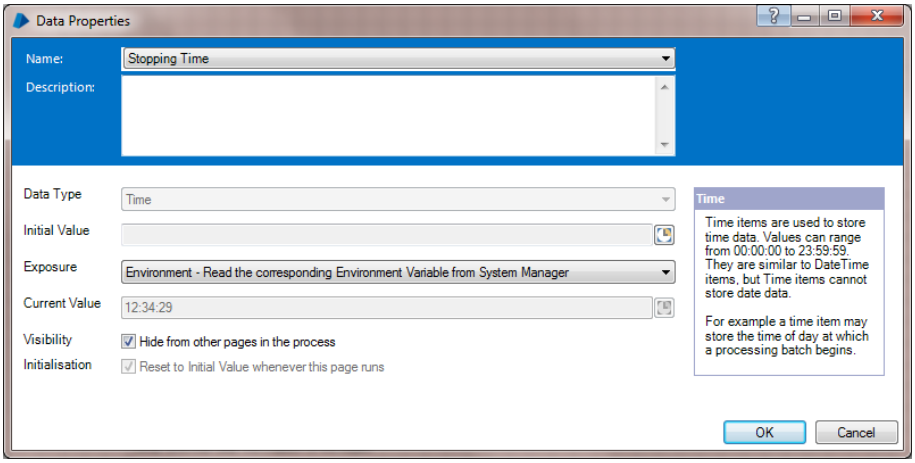


Figure 162: Configuring Data Item Exposure

- We also want to use a “Sleep” action if the “Stopping Time” is not reached. This means the process can “Sleep” or “Wait for a time defined”.
- To use a “Sleep” action, from the previously imported Utility – General Object, place an action on the page and select the Utility – General VBO and use the action “Sleep”.
- You will see there is an “input” for “Sleep Time” set this to 10 seconds. When the process is running if the Sleeping Time is not met, the process can “Sleep” for 10 seconds before carrying on and checking.

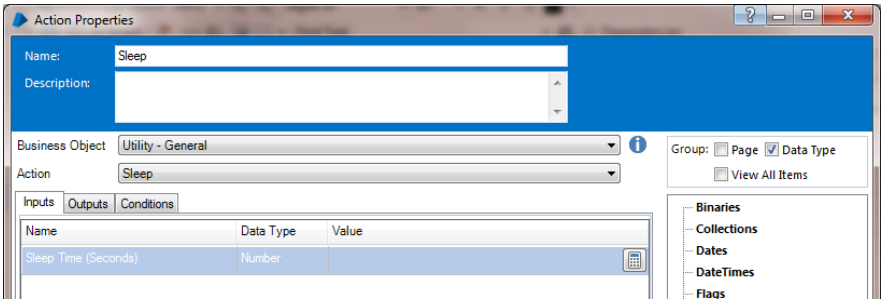
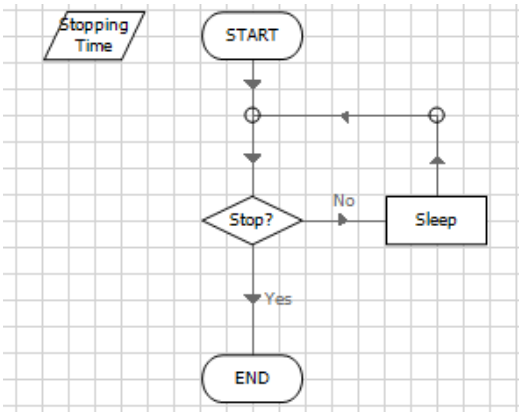


Figure 163: Setting up the Sleep Action

- Configure your Process to spin in a loop until the stopping time is reached.
- IMPORTANT: Switch off logging within the process to prevent the database filling up.

  * *Tip: to disable logging use the combo box at the bottom of the “Stop?” and “Sleep” properties dialog boxes or via the edit menu* **Edit > All Stages.**

- Your process should look something like this:



- Save and publish your Process.
- Amend the System Manager Environment Variable to be one or two minutes into the future, if necessary.
- Run the Process and check it stops at the appointed time.

Exercise 12.3.2 Environment Variable Use Cases

Consider why you might want to centralize the control of such variables into System Manager. Consider also the advantages of several Processes being able to reference the same Environment Variables.

- Create a Calculation stage and attempt to change the value of your environment variable.
- Press the Validate button and observe the error message.

Key Points

- Environment Variables are available to all Processes and Business Objects.
- Data Items exposed as Environment Variables are read-only.
- The name and type of the Data Item must match the Environment Variable.

Session Variables

Like Environment Variables, Session Variables are exposed ***outside*** the diagram, and as the name suggests, are applicable to ***sessions***, i.e., running Processes.

Key Points

- Session Variables are specific to that ***instance*** of the Process. If two instances of the same process are running at the same time, they will both have the same Session Variables but the Session Variables will have different values.
- Session Variables need no setup in System Manager.
- Data Items exposed as Session Variables are writable.
- Session Variables can be viewed and modified from Control Room.

Exercise 12.3.3 Using Session Variables

- Change the Exposure of the **Stopping Test** Data Item to **Session**.
- Set the initial value of the Data Item to be **23:00:00**.
- Run two copies of the Process in Control Room.

- Click Show Session Variables.
- Observe the value of the session variables as the Processes run.
- Modify the stopping time of one of the sessions to be in 10 seconds time. Wait for the Process to stop.
- Notice how the other session was unaffected.

12.4. Casting

Casting is when a value of one data type is automatically transformed to a different data type.

Exercise 12.4.1 Casting Examples

- Create a Calculation with the expression **123**.
- Press the Evaluate Expression button to see the message **Expression Result = 123 (number)**.
- Now change the expression to **"123"** (i.e., with quotes).
- Evaluating the expression now gives the message **Expression Result = 123 (text)**. Why is this?
- The first expression is a number but the second expression is not, it is text containing numerical characters.
- Now evaluate the expression **"True"** (i.e., with quotes).
- And then change the expression to **True** (i.e., without quotes) and evaluate again.

What is happening? The answer is that the expressions are resulting in different ***data types***.

We have mentioned that a Data Item must have a data type, and that data type must be set up in advance and cannot change when the Process runs. However, it is possible to "force" a Data Item to accept a value of a different data type. When faced with a different data type, a Data Item will attempt to interpret the value as its own data type.

Exercise 12.4.2 A Casting Calculation

- Go back to the Calculation and give it the text expression **"123"**. We know the result will be text, but use a number Data Item in the Store In field anyway.
- Step through the Calculation and see that the number Data Item current value is set to **123**.

The text result has been automatically translated into a number so that it can be stored in the number Data Item. This is casting.

- Change the expression to **"abc"** step through again.
- This time, the expression result could not be interpreted as a number and an exception has occurred.

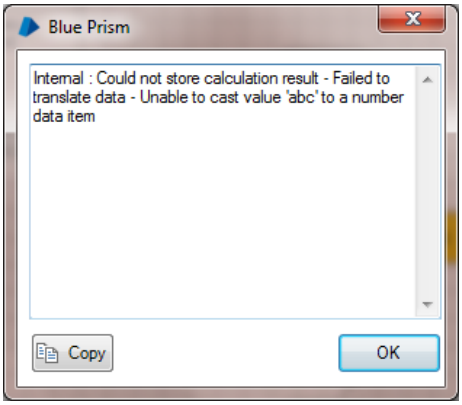


Figure 164: Casting Error Message

It is important to realize that although casting can be very useful, it should be used with care. Casting is not infallible; sometimes the translation cannot be made, causing an exception.

The most common practical use of casting is in the handling of application screen data. Normally these data are read and written as text but it is likely that some of it will need to be cast to another data type at some point.

Casting can be done automatically by changing the element data type in Application Modeler. This will change the data type of the inputs and outputs of read and Write stages that use that element. However, this setting should only be changed with careful consideration, as it depends on the casting operation always succeeding.

For example, suppose an element named Account Balance was given the number data type. A read stage using this element would then have a number output parameter. This would be fine as long as the application always showed a number in that field. If ever the application displayed a non-numerical value (such as “1.23DR”, “£???” or “ERR”), the read stage would create an exception because it failed to do the cast.

These kinds of problems are fairly common and it is recommended that data are read from and written to an application as text. This simplifies moving data to and from the application and allows any necessary casts and checks to be explicitly shown in the diagram.

12.5. Code Stage

The code stage allows Microsoft.Net code scripts to be embedded in a Business Object (but not a Process). A code stage can have inputs and outputs and can be written in VB, C# or J#.

Typically code stages are used to make use of some functionality that Blue Prism does not provide.

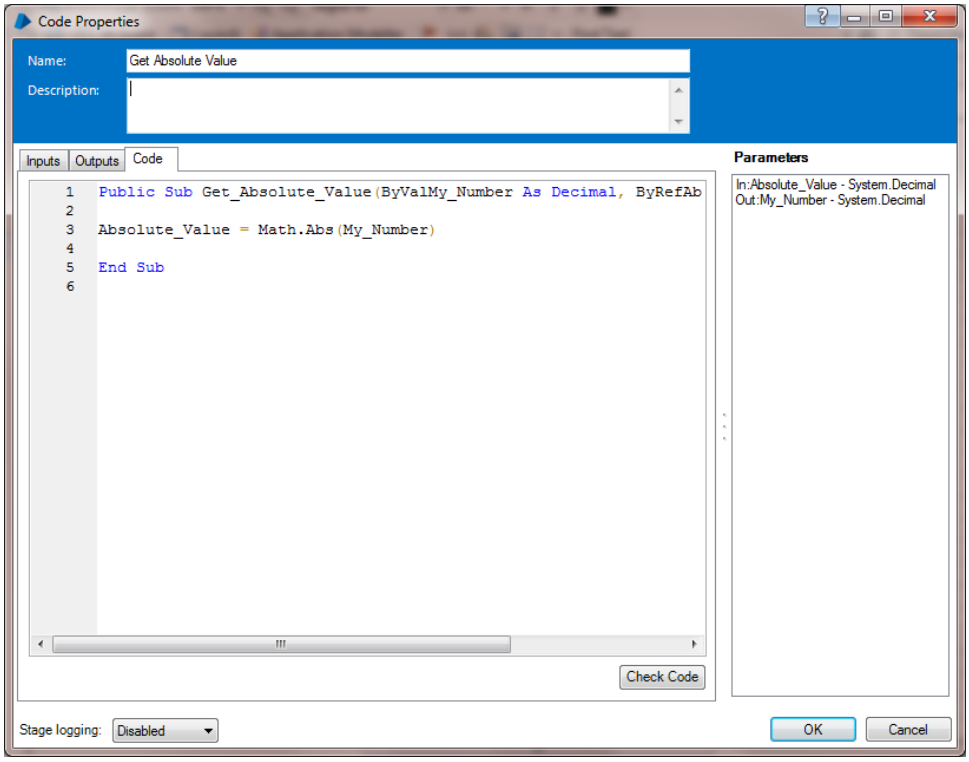


Figure 165: Code Properties

For example, a code stage named ***Get Absolute Value*** with a number input named ***My Number*** and a number output named ***Absolute Value*** would automatically generate the following sub routine outline:

```
Public Sub Get_Absolute_Value(ByValMy_Number As Decimal, ByRefAbsolute_Value As Decimal)

End Sub
```

The user would then be able to complete the body of the sub-routine as required.

```
Public Sub Get_Absolute_Value(ByValMy_Number As Decimal, ByRefAbsolute_Value As Decimal)

Absolute_Value = Math.Abs(My_Number)

End Sub
```

The Business Object information stage on the Initialize page provides options related to code stages such as the following:

- References to DLLs from the .Net framework.
- Name space imports.
- Global code.

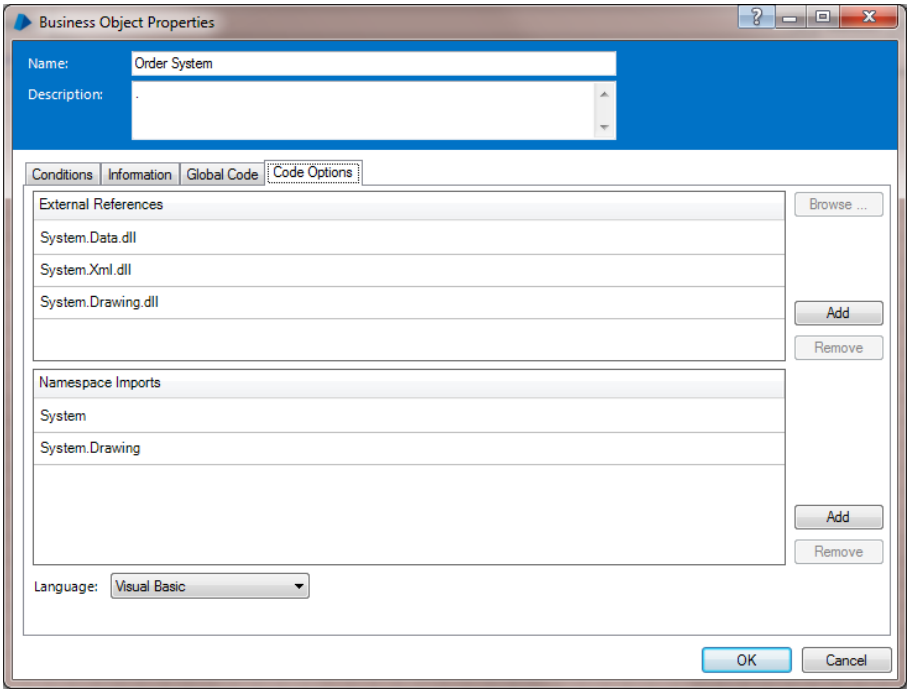


Figure 166: Code Options

12.6. Run Mode

Run mode determines how a Process can be run in relation to other Processes. Run mode is defined in the Business Objects (and therefore by the applications) that a Process uses and is designed to safeguard Processes getting in the way of each other.

Run mode is set on a tab in the properties window of the Business Object information stage on the Initialize page.

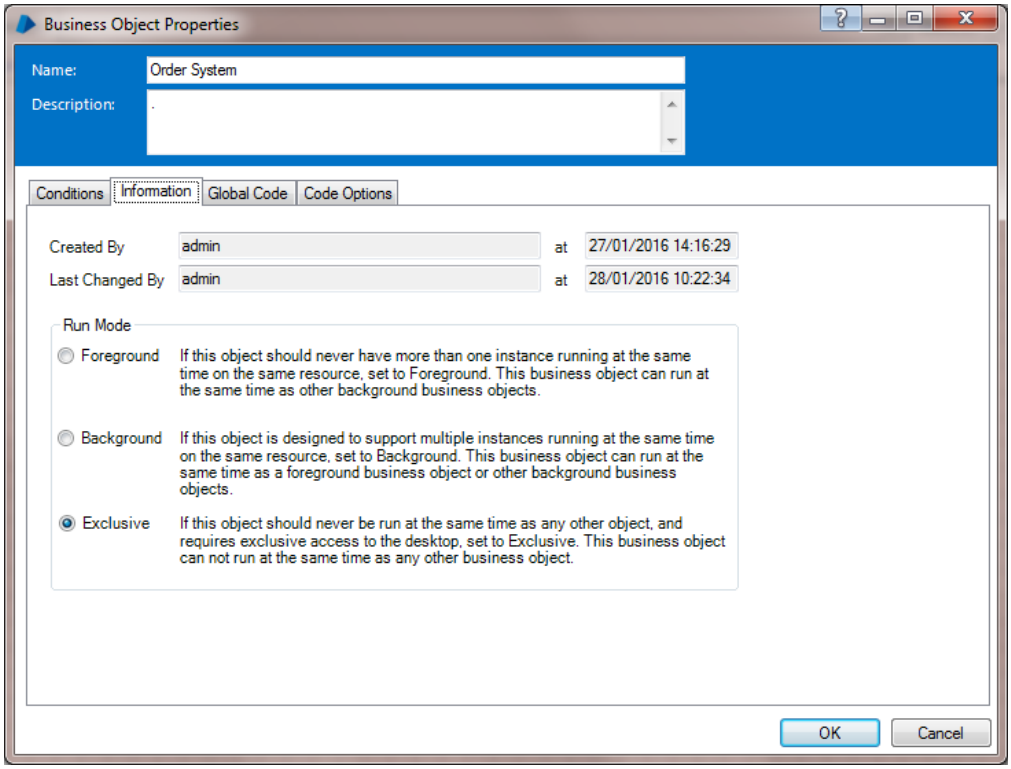


Figure 167: Run Mode

There are three possible run modes:

- Exclusive. An exclusive Process must run on its own and cannot run at the same time as any other Processes.
- Foreground. A foreground Process can only run alongside other background Processes, but only one foreground Process can run at a time.
- Background. A background Process can run alongside other Processes, as long as they are background or foreground Processes.

The run mode order of precedence is Exclusive, Foreground, Background. The implication of this is that the run mode of a Process will come from the “highest” run mode of all its Business Objects. For example, if a Process employs five background Business Objects and one exclusive Business Object, the Process will be exclusive.

Sometimes existing sessions can prevent a new session from being created and a message appears saying the resource is busy. This is due to the run mode of the Business Object(s) used by the Process.

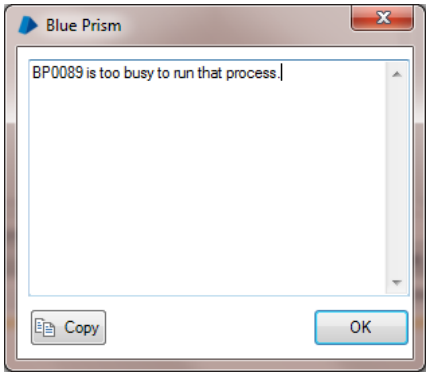


Figure 168: Busy Resource Message

12.7. Initialize and Cleanup

Business objects have two default pages named ***Initialize*** and ***Cleanup***, and these pages cannot be removed or published. They are intended to provide the opportunity to execute logic at the start and end of the life cycle of a Business Object.

The life of a Business Object begins when a Process uses it for the first time. Just before the action is run, the Initialize page is run automatically. By default, the Initialize page simply links the start and End stage, so in effect it does nothing. It is possible to modify the Initialize page to perform some sort of preliminary work.

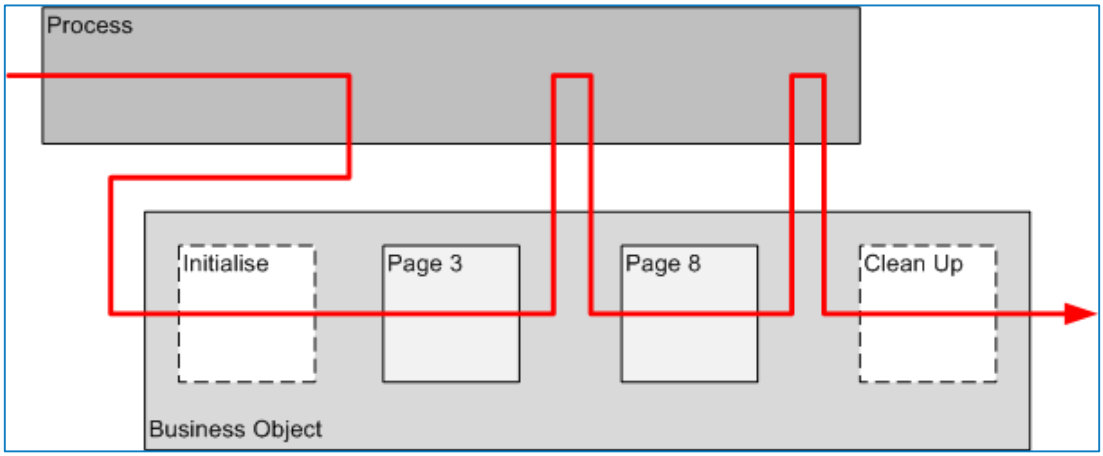


Figure 169: Initialize and Cleanup

The life of a Business Object ends when the Process using it ends. Just before this happens, the Cleanup page is run automatically. Like Initialize, Cleanup does nothing by default but can be modified to perform some penultimate task.

If a Business Object is being used by a Process that in turn is being used by another Process, the life of the Business Object ends when the "master" Process ends.

12.8. Attribute Match Types

Recall that in the Application Modeler, we checked and un-checked the Match column to change our selection of attributes. Next to the match column is another column named ***Match Type***. By default, the Match Type is set to "equals" but there are other options providing a degree of flexibility in how Application Modeler uses the attributes included in the element fingerprint.

- **Wildcards** allow you to perform a fuzzy match on text. **Microsoft Word*** will match "Microsoft Word – Document1" and any other similar name.
- **Numeric Comparisons** – you might be looking for a dialog window which is 300 pixels or fewer in width.
- **Non-Equality** – you might want to match a dialog window which does not have the window title "Error".
- **Dynamic Matches** – a match based on a value which might change all the time, discussed in the next section.

Exercise 12.8.1 Wildcard Matches

- Create a Business Object **Attribute Match Types Test** which automates the Microsoft Paint application (C:\windows\system32\mspaint.exe).
- Spy the main window (whose title is usually **untitled – Paint**) and click Highlight to ensure your match works.
- Change the Window Text attribute match type to **Wildcard** with value ***Paint** and click Highlight again.
- Modify and save the drawing in Microsoft Paint - notice the window title has changed.
- Does your element match still work with a wildcard?
- Change the match type back to Equality. Does it still work?

Exercise 12.8.2 Numeric Matches

- Open a dialog box in Microsoft Paint, such as the **Image Properties** window.
- Spy it and find its width from the Attributes list.
- Experiment ways in which your match can be supported by a comparison of widths.

Exercise 12.8.3 Non-Equality

- Think of your own method of testing using this match type.

12.9. Dynamic Attributes

The Dynamic match type allows us to specify the value of an attribute ***from the diagram*** rather than use the one in Application Modeler. When we mark an attribute as Dynamic we are essentially telling Application Modeler, "Don't use your value for this attribute, use the value the diagram is going to give you."

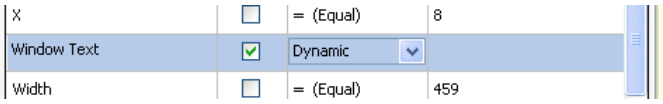


Figure 170: Dynamic Attributes

The most common example is to match against the value of a Data Item; for example an "Account Details" window might contain the customer's name or ID in the window title, i.e., something we are not going to know until ***run-time***.

When an attribute is made dynamic, the matching value needs to be supplied on the fly, each time that element is used, be it in a Read, Write, Navigate, or Wait stage.

A dynamic attribute can be thought of as a mandatory input parameter required by any stage using the element. To continue the example of customer details making up part of a window title, the expression might look like the following:

"Customer System " & [Account Number] & " " & [Account Holder]

The normally disabled parameters button (Params for short) in the stage properties widows becomes enabled whenever an element with dynamic properties is used.

Exercise 12.9.1 Using the Dynamic Match Type

- Open the **Order System** Business Object and go to Application Modeler. Create a new element and spy the **Staff Number** field as before.

Recall that we need to include the ***Ordinal*** attribute to be able to distinguish Staff Number from the Password field.

- Change the match type to Dynamic. Note that the Value column has become grayed out, this is because the value must be supplied from the diagram.
- Close Application Modeler and add a new page to the Business Object.
- Create a Write stage to write "Hello" to the new element. In the Write stage properties, see how the button in the Params column is enabled when you drag in the new element.

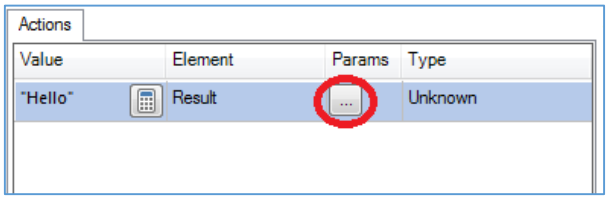


Figure 171: The Params Button in a Write Stage

- Press the button to open the Application Element Parameters window and enter the value "99".

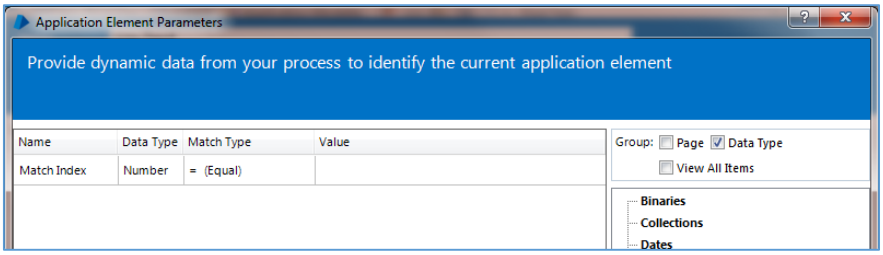


Figure 172: The Element Parameters Window

- Run the page to see if it works.

The write has failed because Application Modeler cannot find any element with matching attributes. The value of Ordinal is being supplied by the Write stage and, in this scenario, is providing a spurious value.

- Change the dynamic attribute to 3 and check that the page runs correctly.
- Now copy the Write stage and paste in another Write beneath the first.
- Edit the new Write stage so that the dynamic attribute has the value 2.
- Go to Order System and manually clear the Staff Number field.
- Run the page again. What has happened?

The second Write stage has written to the Password field. Effectively, the second Write stage has said to Application Modeler, “Find the element using 2 as the value of the Ordinal attribute.” and because we have stipulated Ordinal 2, Application Modeler has found the Password field, not the Staff Number field.

12.10. Active Accessibility

When Application Modeler spies a Windows application, it makes use of a Microsoft API (Application Programming Interface) named ***Win32***. This is how Application Modeler is able to “see” the elements of an application and retrieve information about them. However, in some applications, Win32 does not provide Application Modeler with sufficient information for spying.

As an alternative, Application Modeler can be switched to another Microsoft API named ***Active Accessibility*** (or ***AA*** for short). This API is designed to facilitate applications for people with impaired vision or hearing, such as screen readers, and Application Modeler can also use it for spying.

The spy mode is changed by pressing the “ALT” key. Win32 is the default, and pressing “ALT” repeatedly switches the spy mode.

Key Points

- Blue Prism has ***multiple*** spy mode options.
- After pressing the Identify Element button, use the ALT key cycles through them.

An element spied using AA will have a different set of attributes to a Win32 spied element, but they are used in the same way. Elements found with different spy modes can be used in the same Business Object.

Active Accessibility and Win32 are also available when spying a browser application. Application Modeler is in HTML spy mode by default for a browser application, but “ALT” can be used to switch to AA or Win32 as necessary.

Exercise 12.10.1 Spying with Active Accessibility

- Go back to the **Order System** Business object and create a new element in Application Modeler.
- Press the Identify Element button and move the mouse over the application.
- Press the ALT key to switch to **AA** mode and you should see the red graphics change to blue.
- Spy an element as normal and see how a different set of attributes is shown in Application Modeler.
- Compare the new element with existing elements spied using Win32.

Tips when using Active Accessibility

- The Active Accessibility interface can be much slower than the default Win32 interface. The Match Index and Match Reverse (explained in a later section) can be used to provide a significant performance improvement.
- When using Active Accessibility, multiple matches can be found even though the element seems to be unique. This is because elements may exist even when they are not displayed. Checking the Invisible attribute (which will be set to False) ensures that only elements currently on the screen will be matched.

12.11. Application Manager Mode

Application Manager can run in a variety of different modes, which affect how Blue Prism integrates with the application that is being modeled. For our training exercises, we left the default value.

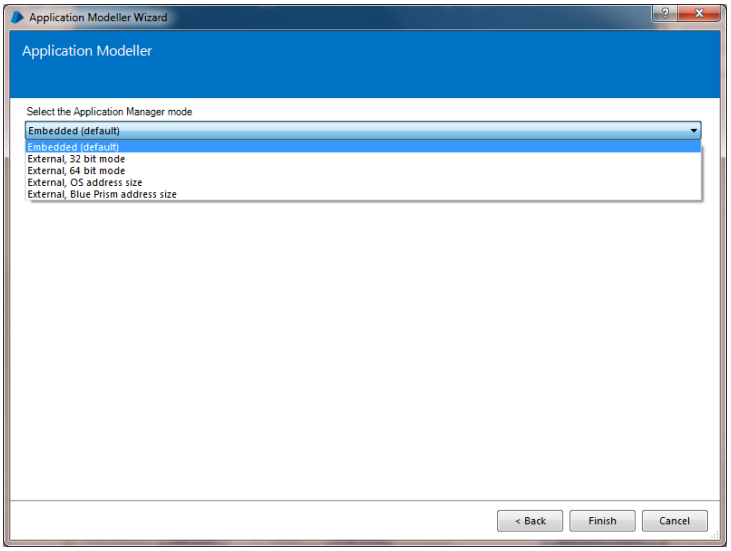


Figure 173: Application Modeler Modes

If Blue Prism is running as a 32-bit process, but the application being modeled is a 64-bit process, then "External, 64-bit mode" should be selected.

Some unreliable applications may impact Blue Prism if they crash. If this is the case, an external mode should be selected so that the unreliable process does not have a detrimental impact on the Blue Prism process.

An External mode is also recommended when modeling Java applications: this is due to limitations of the Java Access Bridge interface.

12.12. Global Clicks and Keys

Some forms of navigation require that the target application is active (i.e., on top of the screen) and some can be performed when the application is inactive or even minimized.

Global Mouse Click and ***Global Send Keys*** are examples of operations that need an active application. If the application is not at the front, these operations will click or type onto whatever window is active, even if it is a different application or just the empty desktop.

For this reason, global clicks and keys should be used as a last resort and handled with care.

12.13. Credentials

Application account details (i.e., usernames and passwords) could simply be kept in Data Items within a Process or Business Object, but ***Credentials*** provide a more secure and centralized storage option.

Credentials are a secure repository for details used to log in to target applications. They are encrypted in the Blue Prism database to make them only accessible to those who should be able to use them. The Credentials Management system determines which Processes, Resources, and Roles have permission to access the information, and a special ***Internal – Credentials*** Business Object provides actions for using credentials.

Credentials are set up in the Security section of System Manager but as an advanced topic they are not covered this course.

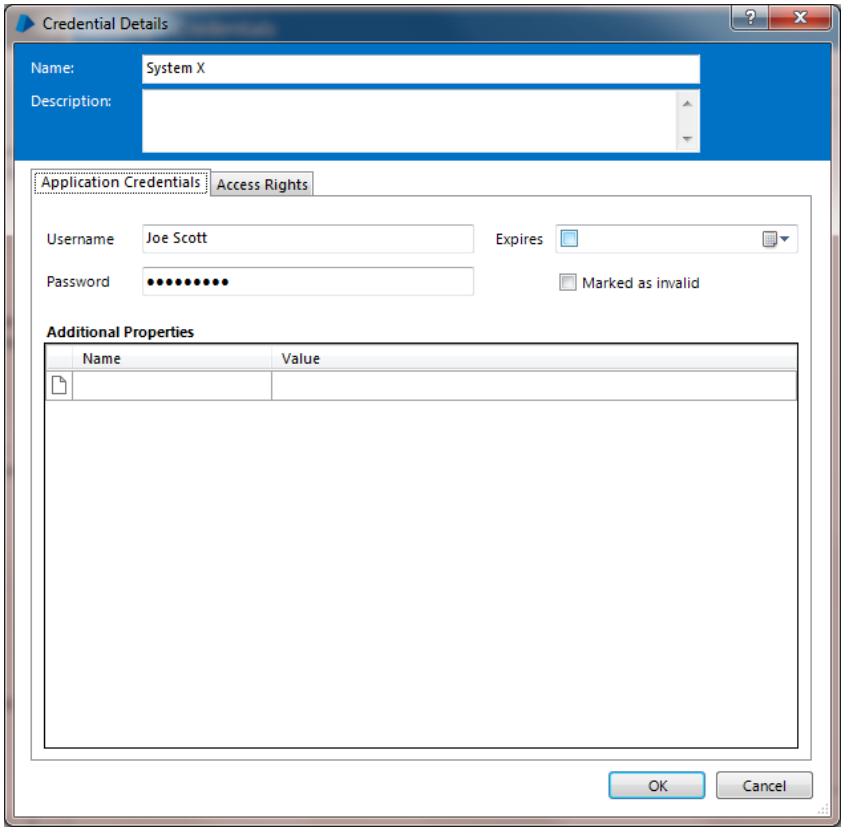


Figure 174: Credentials

12.14. Environment Locking

Sometimes it is necessary to synchronize access to a shared resource like a file. For example, suppose you had a Process that needed to access a shared file but you also need to run more than one instance of that Process. Potentially the Processes could clash if they tried to open the file at the same time.

One solution would be to remove the file accessing sequence from the Process and put it into a separate Process that will only ever run on one machine. Another way would to use Blue Prism's ***Environment Locking*** feature to control access to the file.

An Environment Lock is basically a key or token that a Process must obtain as "permission" to take a particular path. When there is only one lock and more than one instance of a Process is running, the instances must "compete" for the lock.

Exercise 12.14.1 Acquiring a Lock

Another special Business Object named ***Internal – Environment Locking*** is used to work with locks.

- Create a new Process **Lock Client 1** which acquires a lock according to the illustration below.
- Read the documentation (by clicking the information button) and consider the purpose of the input parameters. Make sure you collect the **Token** output parameter.

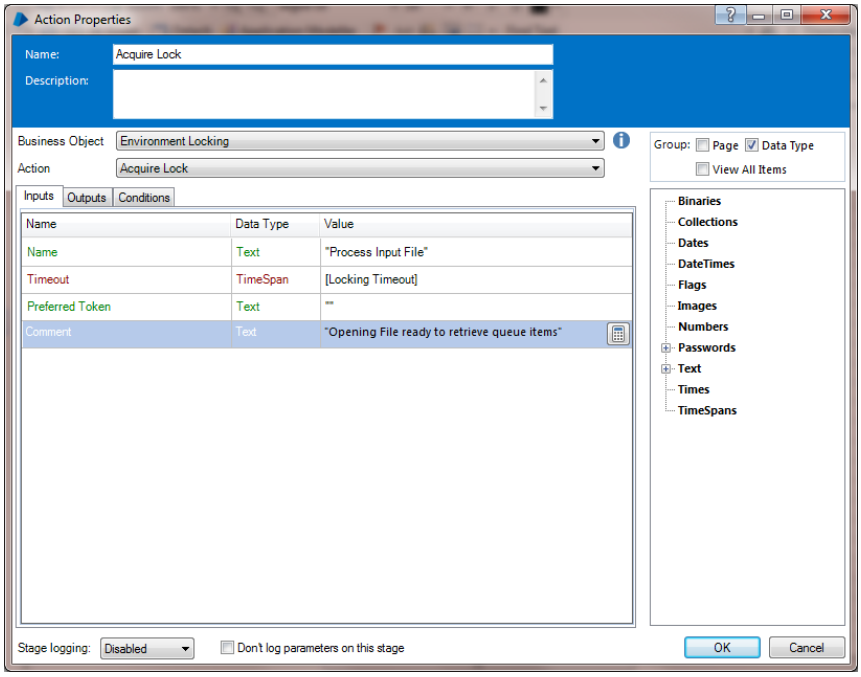


Figure 175: Acquiring a Lock

- Go to System Manager and look at the Environment Locking tab in the Workflow section.
- Review the list of active locks. Note the padlock status beside your item, indicating that the lock has been acquired.

Key Point

- When you request a lock, it will be created automatically if it does not already exist.

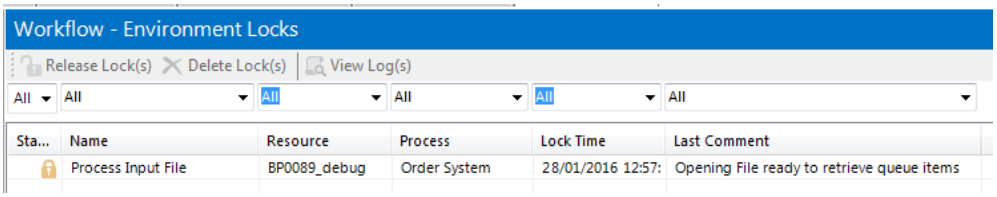


Figure 176: System Manager Environment Lock List

- Use File->Save As to clone your Process as **Lock Client 2**.
- Attempt to obtain the same lock by stepping through the Process. What happens? How do you know whether or not the lock acquisition has succeeded?

Exercise 12.14.2 Releasing a Lock

- Add a new stage to **Lock Client 1** to release the lock. Review the appearance of the lock list in System Manager. Now acquire the lock using **Lock Client 2**. Check that the acquisition succeeds.
- Do not release the lock during what follows. Modify Lock Client 2 so that its job is to acquire the lock and then wait 30 seconds. Save and publish to Control Room. Close your Process.

> *Tip: To wait 30 seconds, you can make use of an empty Wait stage in one of your Business Objects – just set the timeout to 30.*

- Check the status of the lock in System Manager. You should find that the lock has been released, even though the last thing you did was acquire it without releasing it.

Key Point

- The system will automatically release any locks that your Process neglects to release in the proper fashion, whenever the Process stops running. This includes Processes running in both Process Studio and Control Room.

Exercise 12.14.3 Proper Locking Management

- Do you think it is sensible to rely on the automatic release of locks or do you think it is better to do everything you can to ensure that your Process releases its own locks?
- What might happen if your Process "hogs" a lock, because it fails to release it in a timely fashion?

Exercise 12.14.4 Orphaned Locks

- Run **Lock Client 2** again from Control Room. Quickly return to System Manager and check the status of the lock. It should be locked by the Process.
- At the end of the 30 seconds, what do you expect to happen?
- Run the Process again to repeat the exercise. While the Process is still running, kill Blue Prism using Windows Task Manager.
- Return to the locking list and observe that this time the lock has not been released. It is said to have been **orphaned**.

When an unexpected crash or power cut interrupts a Process, you will not be able to rely on the software to automatically release the lock. In such cases you will need to manually intervene in System Manager to release the lock.

12.15 Command Line

The command line feature enables Blue Prism to be run from the command prompt or a batch file. Full details are provided in the help section, but an example of command usage is as follows:

AutomateC /run "My Process" /user admin mypassword

Exercise 12.15.1 Command Line Help

- Press the Windows Start button, select Run, type "cmd" and then press "OK".

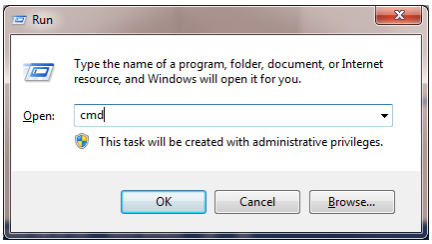


Figure 177: Window Run Prompt

- To see the range of command line options, paste **CD C:\Program Files\Blue Prism Limited\Blue Prism Automate** into the command window and press the Return key.

- Then do the same with this line **AutomateC.exe /help**

12.16 Resource PC

In a real-world environment, it is likely that Blue Prism will be running on multiple machines, with some machines designated as “masters” and some as “servants”. On the master machines, Blue Prism will be run as normal, just as you have been doing so far. However, on the servant machines, it is likely that Blue Prism will run in an alternative, minimal mode known as ***Resource PC***.

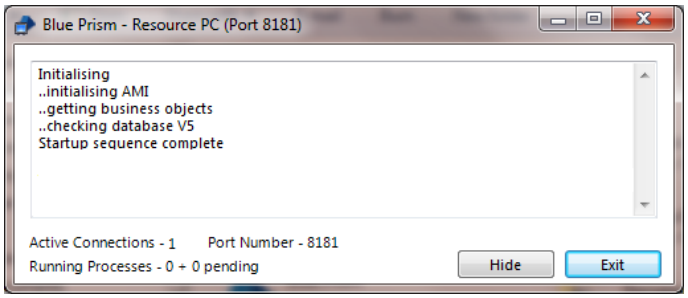


Figure 178: Resource PC

Machines running Resource PC play the role of servant, ready to receive orders from the Control Room of Blue Prism running on a master machine.

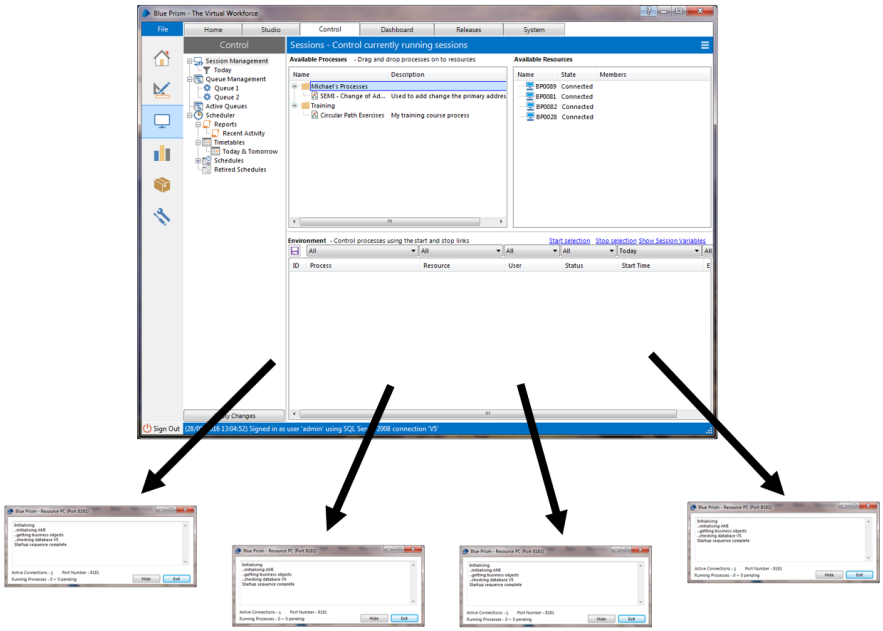


Figure 179: Control Room and Remote Resource PCs

Key Points

- Resource PC is Blue Prism launched in a different mode.
- Any machine with Blue Prism installed can run Resource PC

13. Further Application Types

Blue Prism is capable of interfacing with other application types, but so far we have only looked at the default option of Windows based and Browser based applications.

This section provides an overview to mainframe and Java applications. Separate datasheets are available on the Blue Prism Portal that provides more detail. In addition this section describes Surface Automation that can be when elements aren't accessible to Application Modeler.

13.1. Mainframe Applications

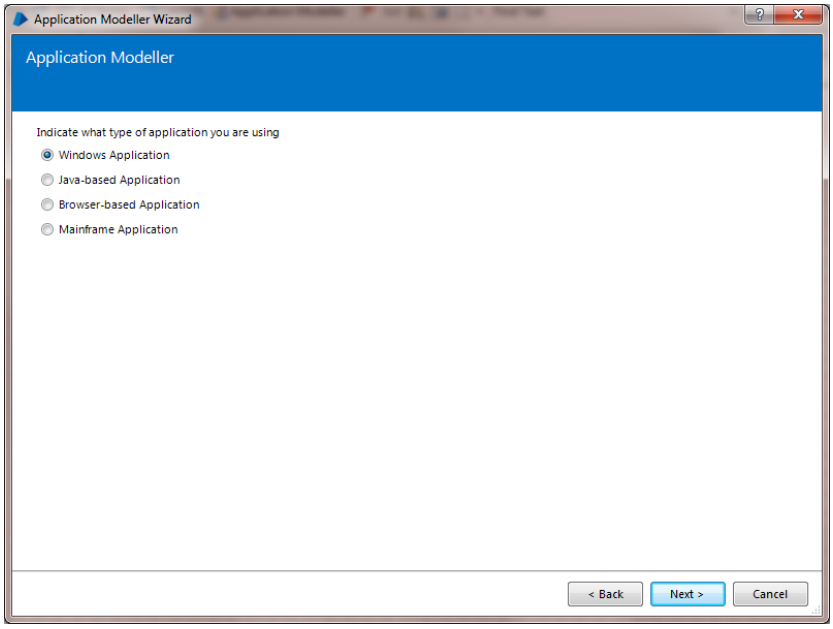


Figure 180: Application Modeler Application Types

Blue Prism is also able to interface with mainframe applications (sometimes known as green screen systems) and the Application Modeler wizard is used to specify the initial details.

- Application type.
- Session file path.
- Session identifier.

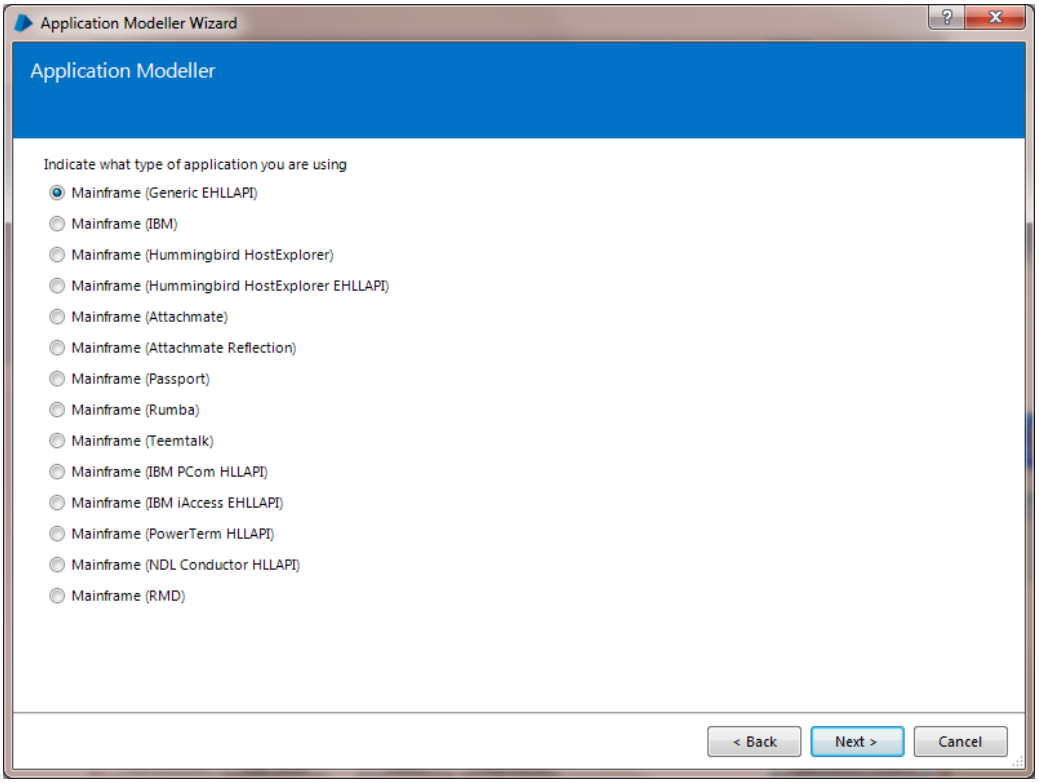


Figure 181: Mainframe Applications

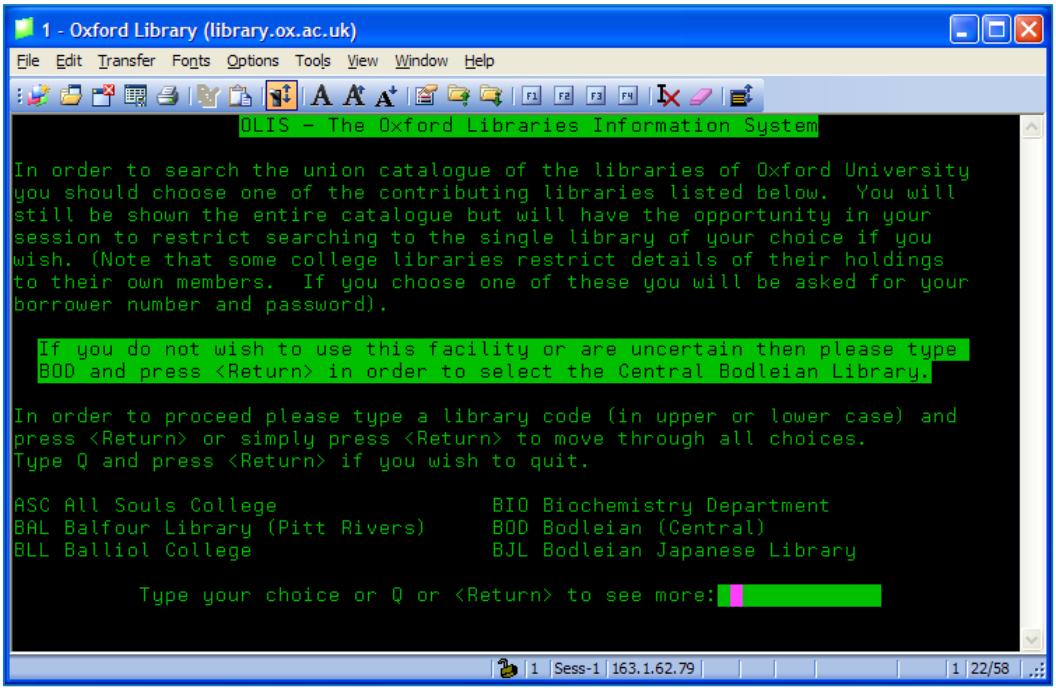


Figure 182: Hummingbird Mainframe Application

Mainframe applications are spied by identifying rectangular areas of the screen. Application Modeler does this by superimposing a grid onto the application to make the rows and columns of the screen visible. One unit of the grid represents one character space on the screen.

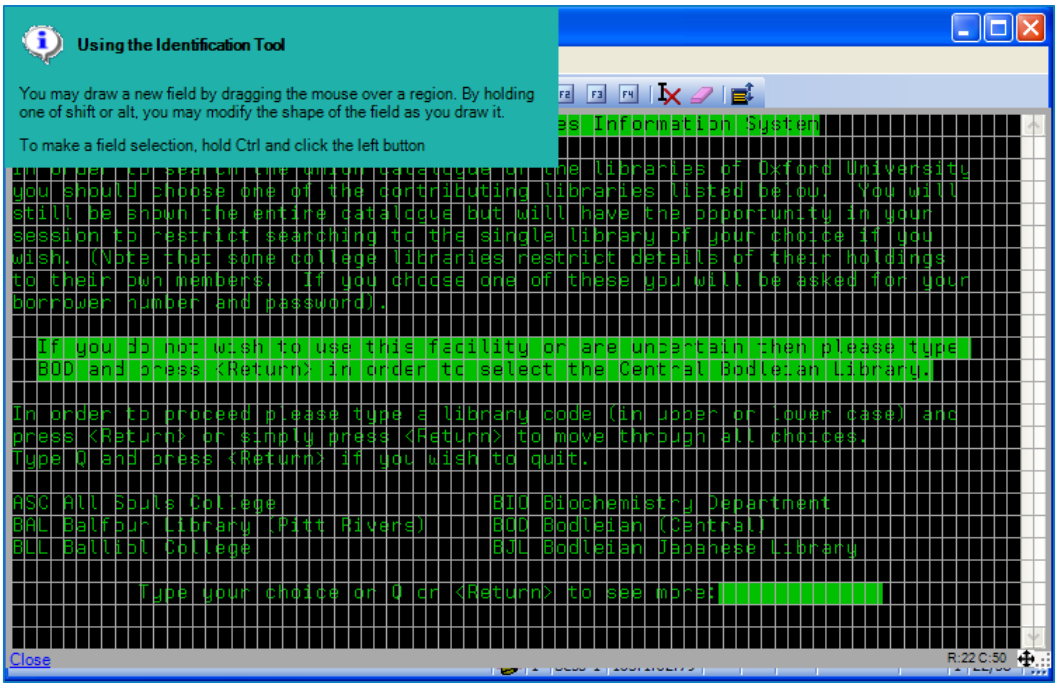


Figure 183: The Mainframe Grid

The operation to capture mainframe elements is slightly different to spying other applications. First, the grid must be created by holding “CTRL” and clicking on the application. With the grid in place, the mouse is used to drag and select rectangular areas. These areas can then be captured in Application Modeler using the mouse menu.

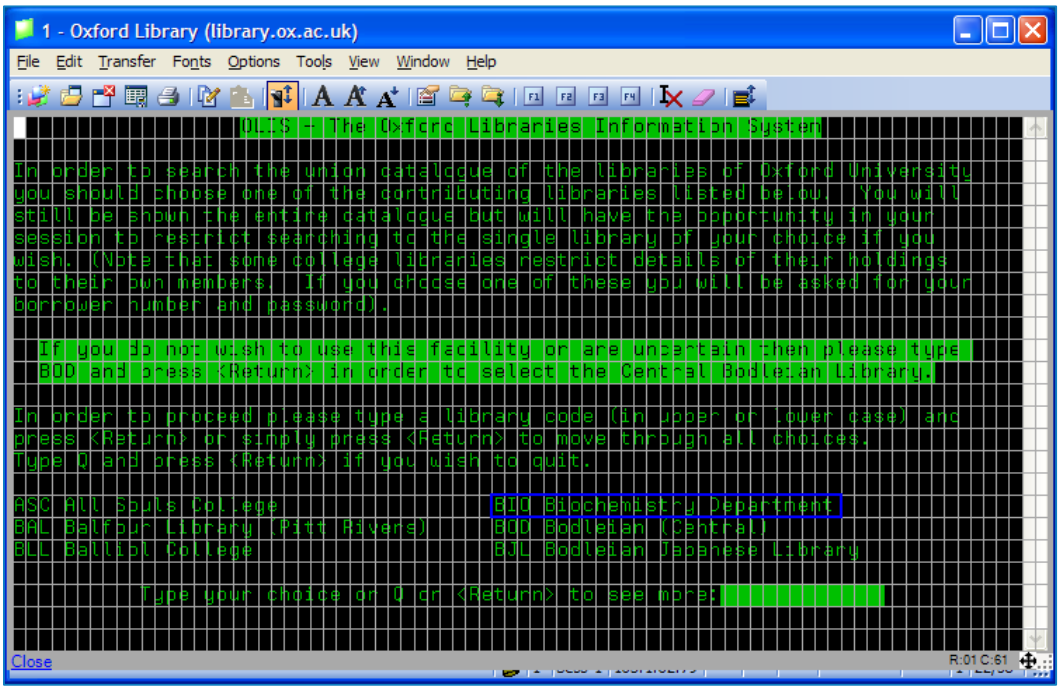


Figure 184: A Mainframe Region Highlighted in Blue

Mainframe elements are simpler than those from other application types and there are far fewer attributes to deal with in Application Modeler.

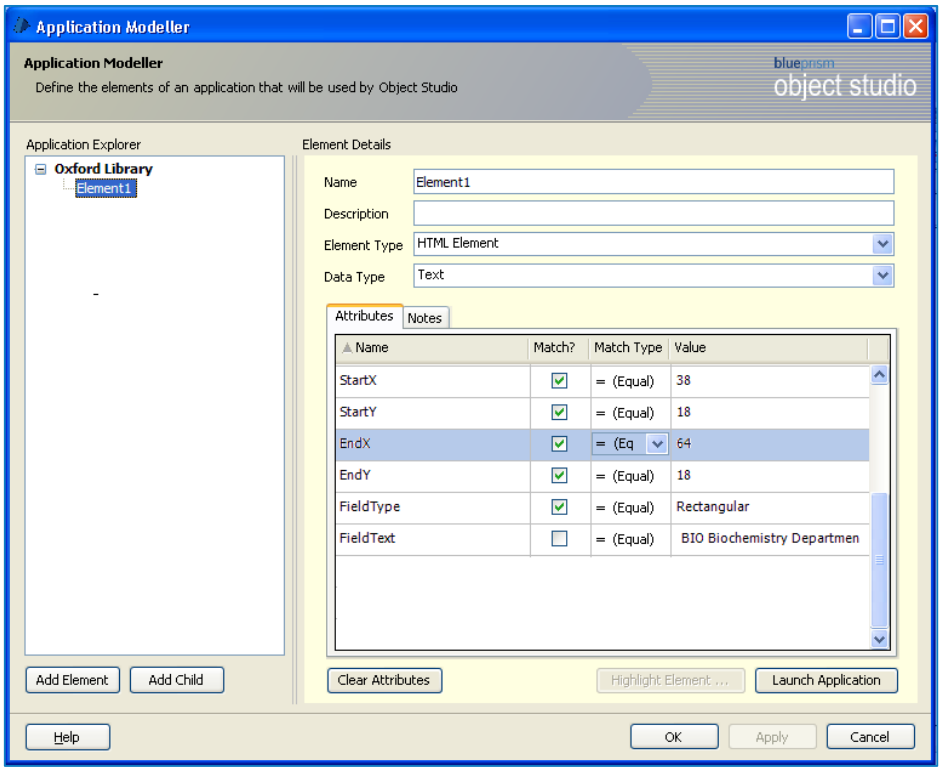


Figure 185: Mainframe Element Attributes

13.2. Java Applications

Blue Prism is also able to interface with Java applications, and creating a Java based Business Object is near identical to making a Windows-based Business Object.

You should find the procedure for spying elements and employing them in the Business Object is very familiar to you by now.

Ancestor Count

The Ancestor Count attribute is a very useful attribute for limiting the search that Blue Prism has to perform in order to identify your element.

Picturing the target application as a tree of elements (in which, for example, a table cell is the child of a table, which in turn is the child of a frame, which in turn is the child of a top-level window) the Ancestor Count attribute records how deep in the tree the element is found.

By matching on Ancestor Count, you can instruct Blue Prism to only search a subset of the tree, thereby improving performance. When you experience slow performance in a Java application, this should be the first attribute you experiment with.

The Match Index and Match Reverse attributes (described below) may also be used to improve the performance of a Java interface.

For some elements and environments, Blue Prism technology connectors (Windows/Java/HTML etc.) can't be used. Examples of this are where applications are presented in a virtualized state (i.e., via a Citrix server), and some bespoke application controls.

Blue Prism provides Surface Automation techniques that can be used as an alternative when technology connectors are not available. Some of these Surface Automation techniques are briefly described in the following Regions, Character Matching, and Global Clicks and Keys sections.

A detailed Surface Automation training course is available, separate to this Foundation course. This additional training course is recommended to anyone needing to use these techniques in their environment.

13.3. Match Index and Match Reverse

Match Index and Match Reverse

When identifying an application element using Application Modeler, you will notice that there are two attributes available named Match Index and Match Reverse. These attributes are available for all application types.

Match Index and ***Match Reverse*** are not attributes of the identified element, but instead are controllers of how Blue Prism will search for the element at run time.

When Match Index is set, Blue Prism will stop searching once an element has been found, instead of continuing to search for potential duplicates. Using this attribute may significantly increase the search speed (***especially when using the Java and Accessibility interfaces***) but should only be used in circumstances where duplicate elements are either unlikely or can be ignored.

When using Match Index, Match Reverse will make Blue Prism search in a bottom-up order rather than the default top-down order. Again, this may improve the search speed but only experimentation will tell.

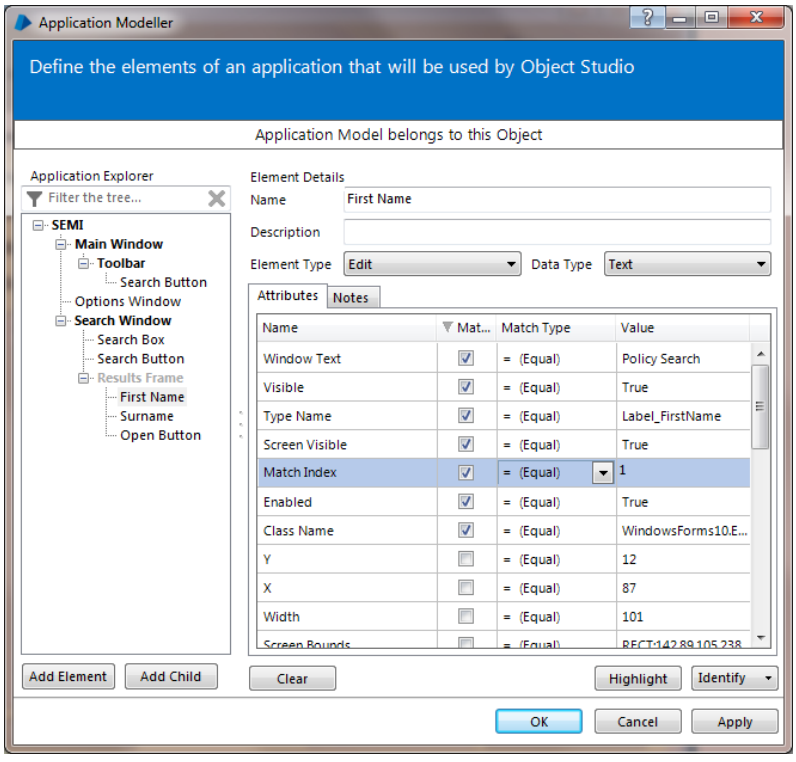


Figure 186: Match Index and Match Reverse Attributes

The match index value can be changed to any numeric value. For example, if set to 2, it will ignore the first element found that matches all the other selected attributes and find the second matching attribute instead if one exists.

This can be a very useful when identifying screen elements that are otherwise not unique. For example, in the earlier Object Studio exercises the Ordinal attribute was used to uniquely identify elements in the Order System application. If the Ordinal had not worked and elements were still not unique (returning multiple matches), Match Index could have been used instead.

13.4. Surface Automation

For some elements and environments, Blue Prism technology connectors (Windows/Java/HTML etc.) can't be used. Examples of this are where applications are presented in a virtualized state (i.e., via a Citrix server), and some bespoke application controls.

Blue Prism provides Surface Automation techniques that can be used as an alternative when technology connectors are not available. Some of these Surface Automation techniques are briefly described in the following Regions, Character Matching, and Global Clicks and Keys sections.

A detailed Surface Automation training course is available, separate to this Foundation course. This additional training course is recommended to anyone needing to use these techniques in their environment.

Key Point

- If Surface Automation techniques are required, additional training is required.

14. Conclusion

This brings you to the end of the Blue Prism Foundation Training Course.

We hope you gained a lot from the training course but if you require any further information please review the our Blue Prism Portal (https://portal.blueprism.com/) or website (www.blueprism.com) for further information.

Blue Prism Portal

The Blue Prism Portal is a resource provided to partners, customers, and developers which facilitates access to the latest software releases, framework and methodology templates, tutorials and guides, and supporting technical documentation

The Blue Prism Portal can be found at: https://portal.blueprism.com

Blue Prism Summary

Blue Prism's Robotic Automation software enables organizations to be agile and cost-effective through rapid automation of manual, rules based, back office administrative processes, reducing cost and improving accuracy by creating a "virtual workforce" to deliver dynamic and rapidly scalable workload capacity.

Blue Prism's proven technology plays a key role in enabling organizations to rapidly respond to business change through agile back office operations. Forrester Research has recently identified the "mega trend" of "Empowered Business Technology" and Blue Prism is one of the first examples of EBT in action.

For more information, email info@blueprism.com.