Blue Prism

Best Practice Build Overview

blueprism®
Robotic Process Automation Software

Delivering the **world's most successful digital workforce.**

blueprism.com

Object Layer

Design Overview

Object Layer Overview

Design Concepts

- Generally we look to have one object per screen with a Get and Set action.
- There are no hard and fast rules regarding how to break up an application's functionality into objects within Blue Prism other than you can't have too many objects; only too few.
- You should have learnt during your training that having multiple objects within the object layer provides a more efficient and scalable design because:

More developers can work concurrently.

A running process consumes only the actions it requires.

Application modeller is smaller and less prone to latent error

Whenever a change is made within the object layer the effect on and risk to the process layer is minimised.

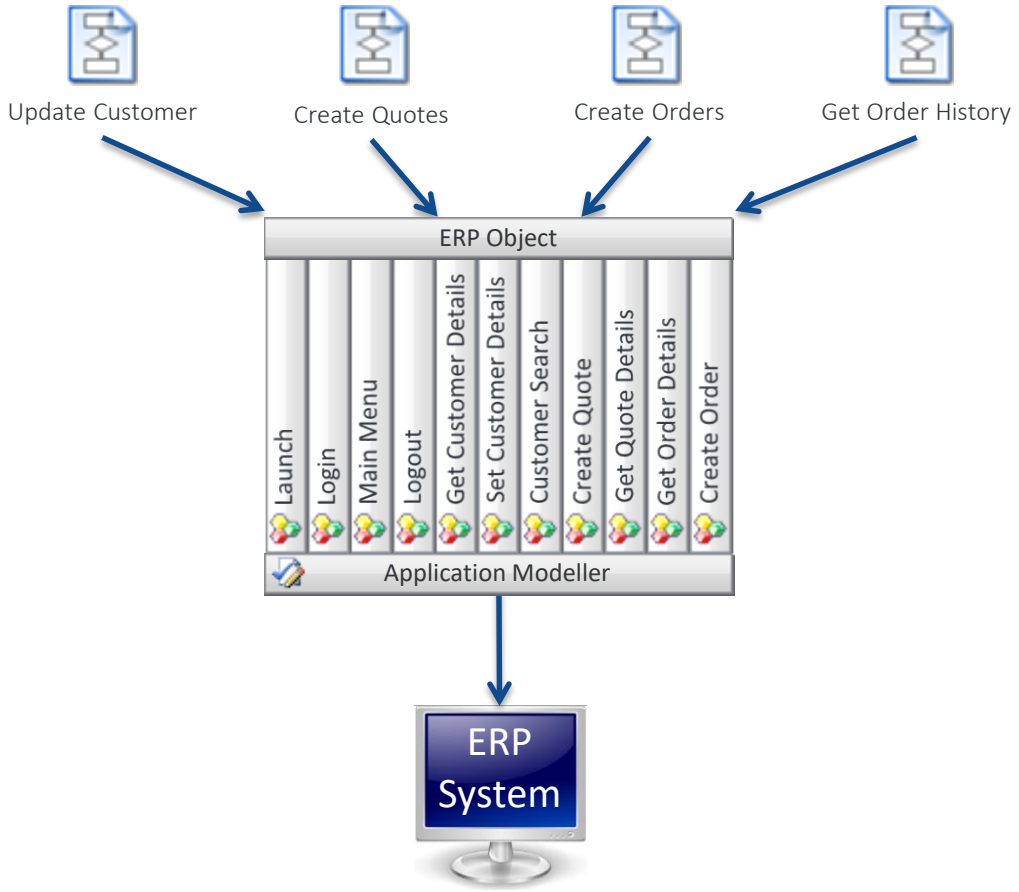
Standard Object Design

Recap

Here we have four Blue Prism processes….

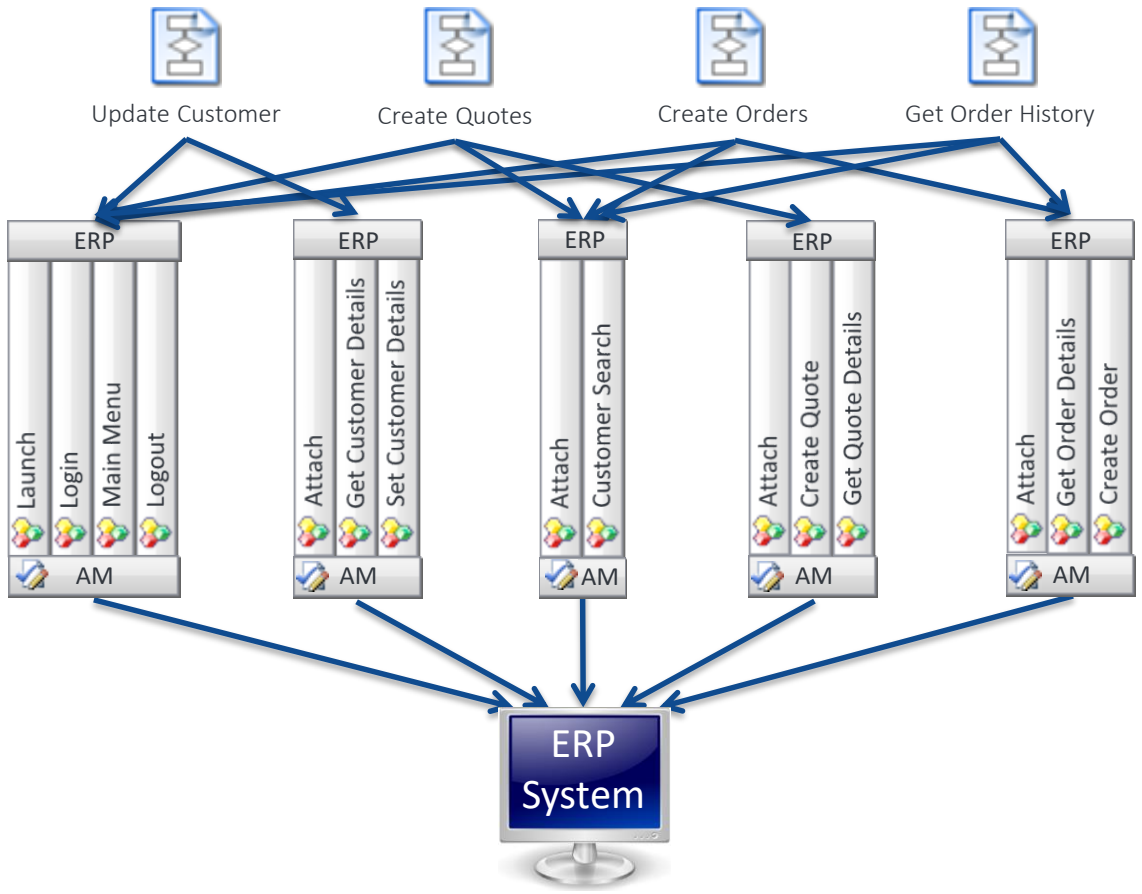
Here's our Blue Prism object….

Here's our target system….



Standard Object Design

Recap



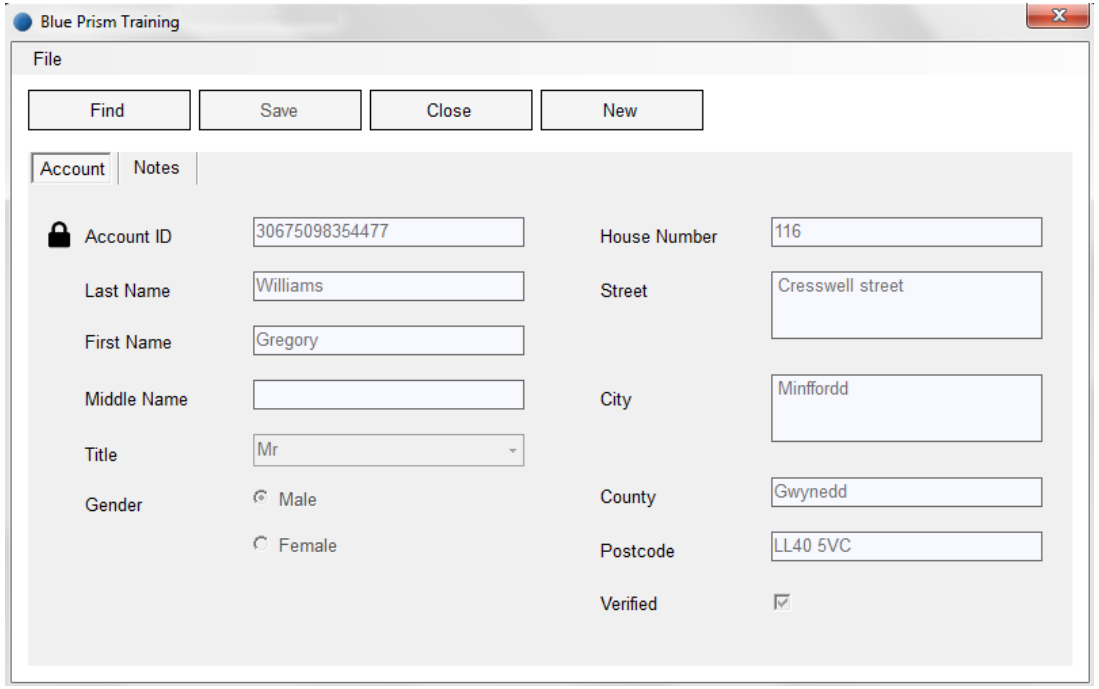
- ✓ Our object layer is more efficient
- ✓ We can have up to five developers building the objects
- ✓ Our processes consume fewer actions at runtime
- ✓ Our objects are smaller with fewer elements in Application Modeller
- ✓ And most importantly of all, when we change an object we minimise the effect on the process layer.

Object Example

Account Details Screen

Training App – Account Details object

Action	Inputs	Outputs
Attach		
Navigate	Find, Save, Close, New, Lock/Unlock	
Get Account Details		Account ID, Last Name, Middle Name, First Name, Title, Gender, House No., Street, City, County, Postcode, Verified.
Set Account Details	Account ID, Last Name, Middle Name, First Name, Title, Gender, House No., Street, City, County, Postcode, Verified.	



Attaching

Attach

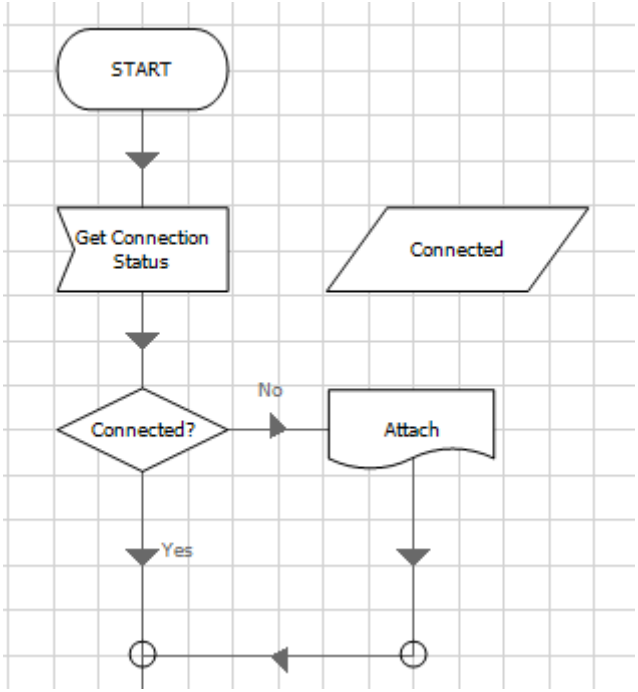
You will have noticed that when we break up the object layer we have added a new action called Attach. This is added to each object apart from the object that launches the target system.

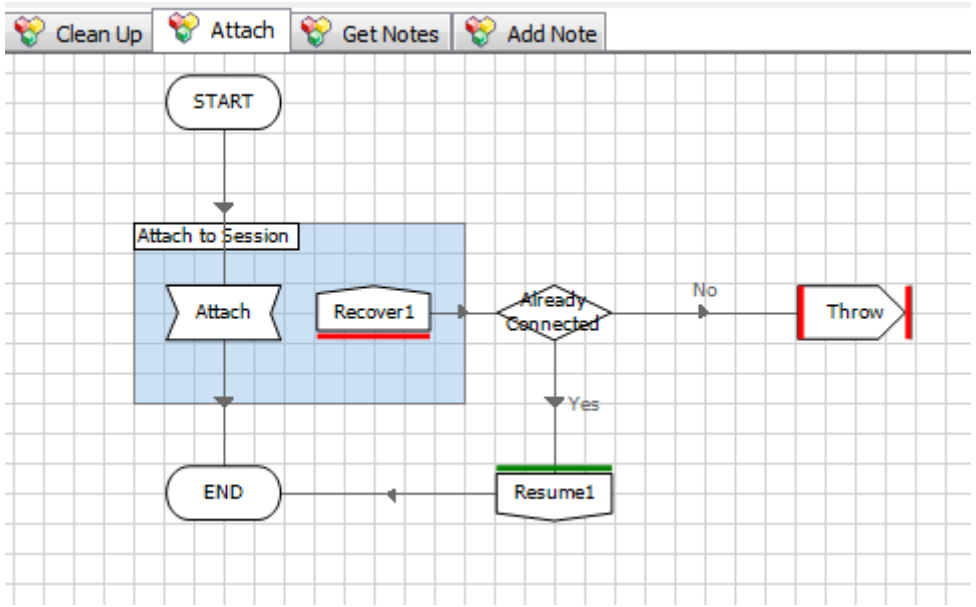
Once the target system has been launched you can either:

Call all the Attach actions in the objects you will use

Or

Check at the start of each action to see if the session is connected and call Attach when required.





Object Layer

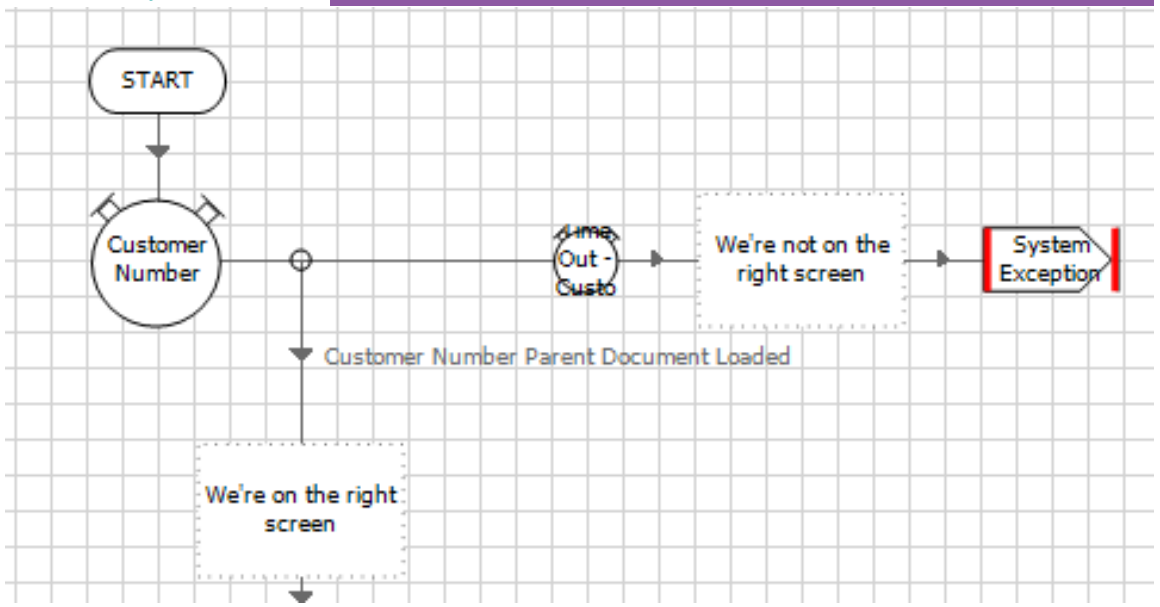
Best Practices

Development Best Practice - Objects

Actions

Wait stage at start of each action

This will confirm the process is on the correct path and absorb system latency to increase the resilience of the process.

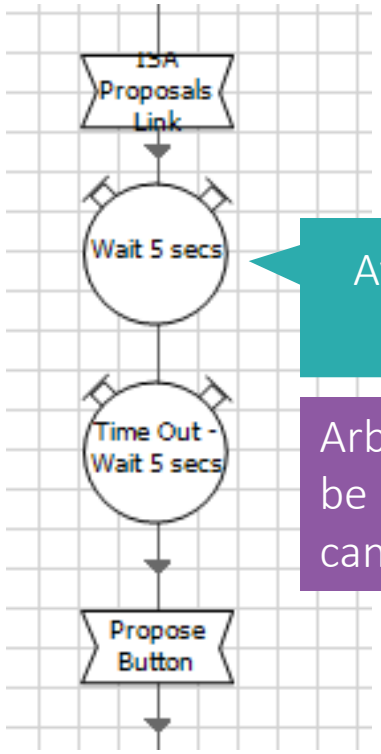


Always throw exception on timeout

Do not try and recover the process following the wait stage. Throw the exception and let the process handle it. The process may choose to try again a few times or restart the system or ultimately raise an alert.

Development Best Practice - Objects

Actions

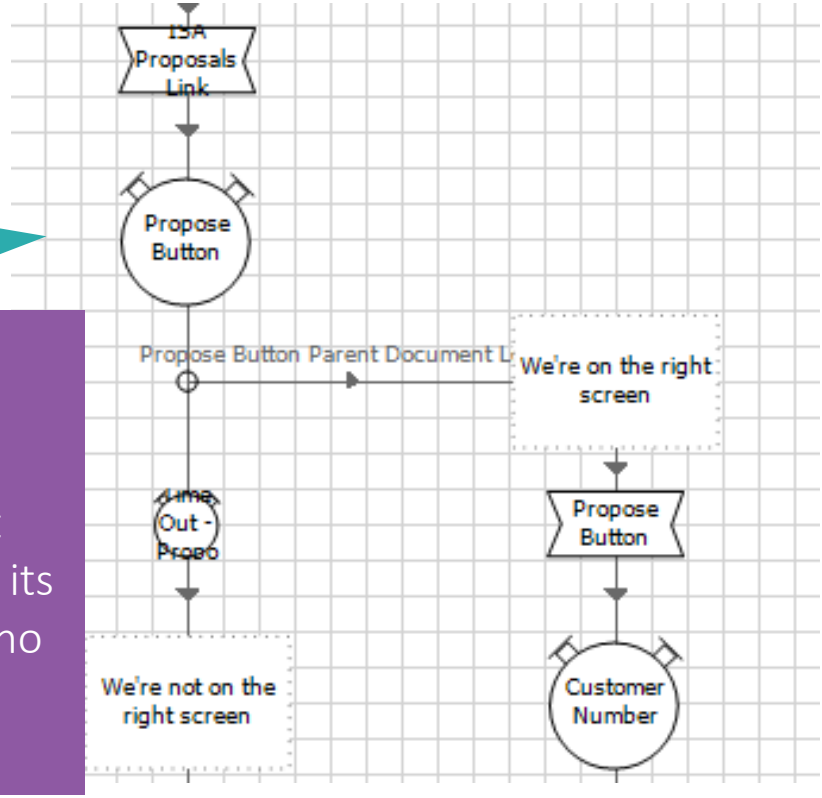


Avoid using arbitrary waits

Arbitrary waits should only be used if a screen change cannot be waited for.

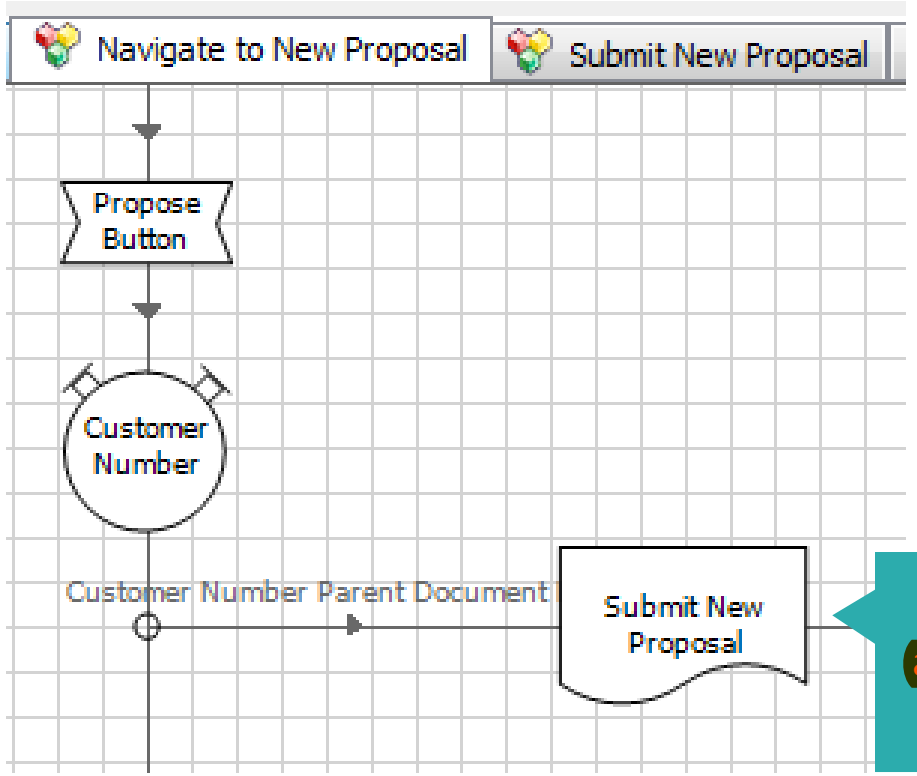
Always wait for the screen to change

Use wait stages after Navigate stages or any stage that causes the screen to update.
This will absorb any latency but also ensure the process runs at its fastest. In this example there's no point waiting 5 seconds if the system is available after 1.



Development Best Practice - Objects

Actions



Do not call published actions from within an object

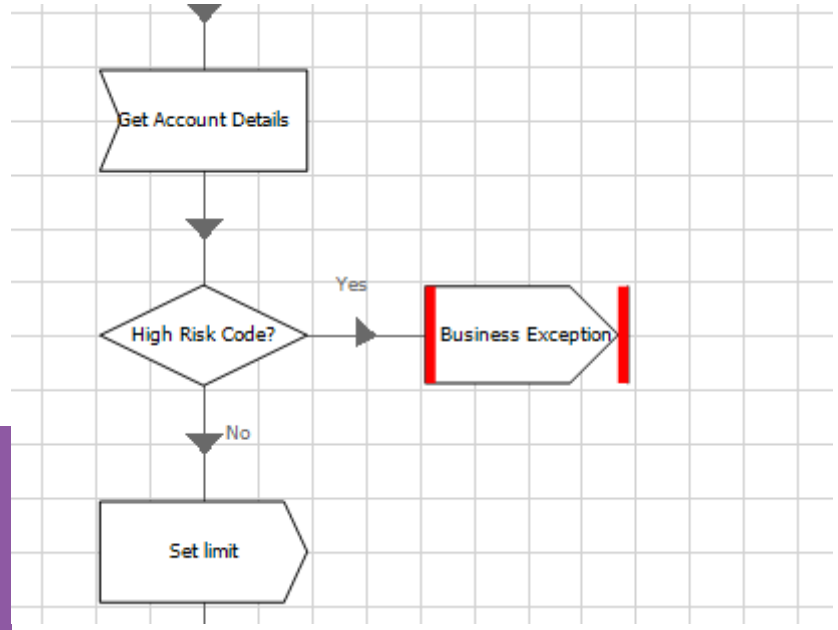
Here the process should call the “Navigate to New Proposal” action and then call the “Submit New Proposal” action.
This will make exception handling far easier and actions more reusable.

Development Best Practice - Objects

Actions

Do not make business decisions in the object

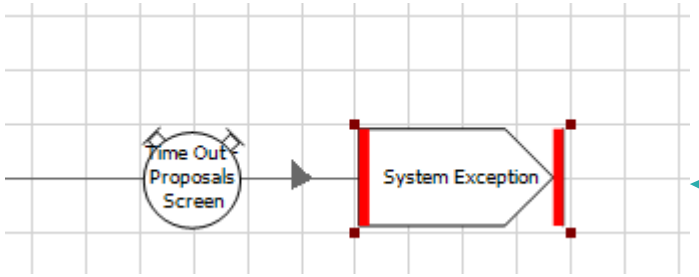
In this example our action gets account details from the screen and applies a new limit unless the risk codes are of a specific value.



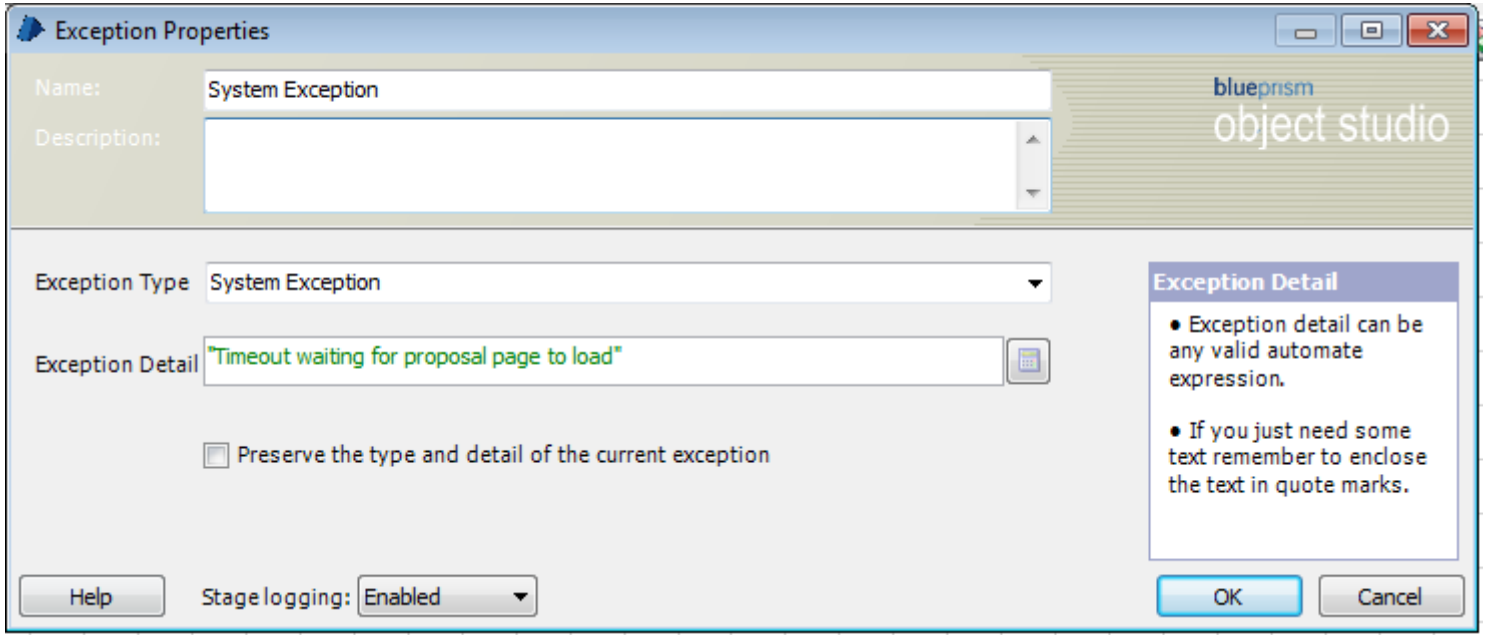
The object design should be such that there are two actions not one.
One action to “Get Account Details” that returns all the account details on the screen and another action to “Apply new limit”.
The decision to apply a new limit based on the risk code should be made by the process not the object.
This enables other processes to call the “Get Account Details” action without applying a new limit.

Development Best Practice - Objects

Actions



For all exceptions provide appropriate Type and Detail

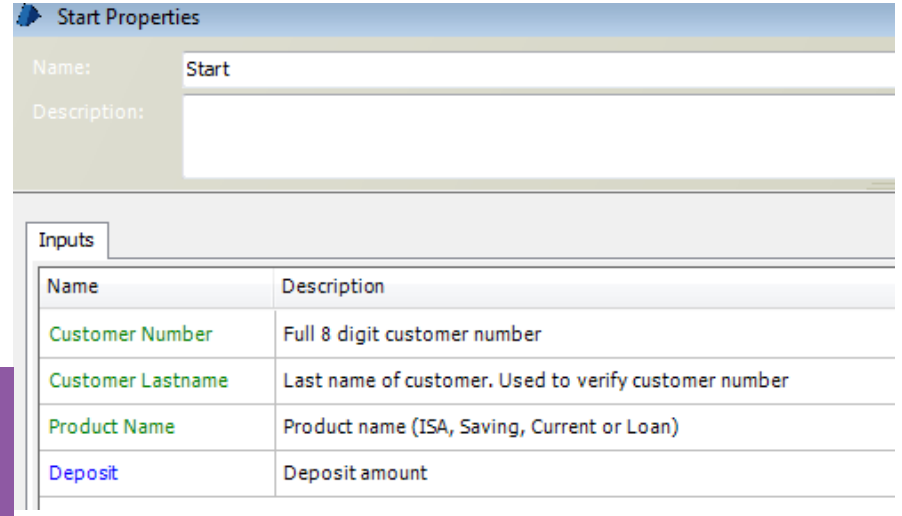


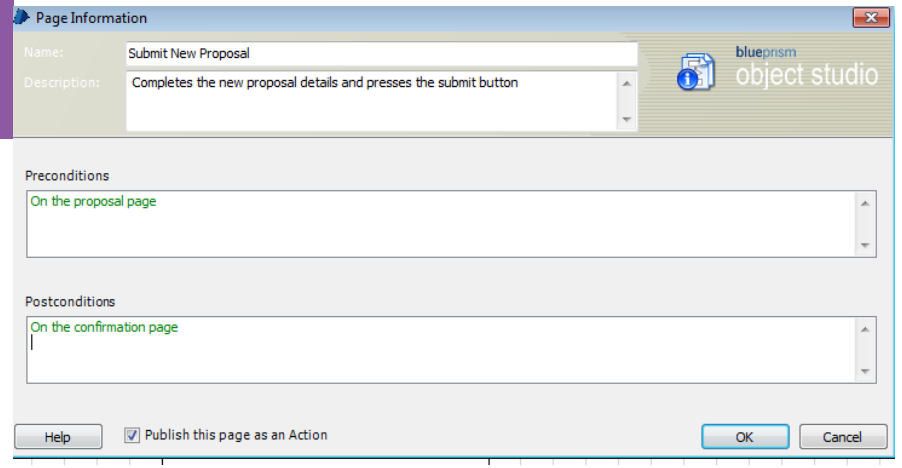
Development Best Practice - Objects

Actions

Provide descriptions to Inputs, Outputs and Actions

This removes ambiguity and also provides content for auto-generated Business Object Definition (BOD) document.





1.8 Submit New Proposal

Completes the new proposal details and presses the submit button

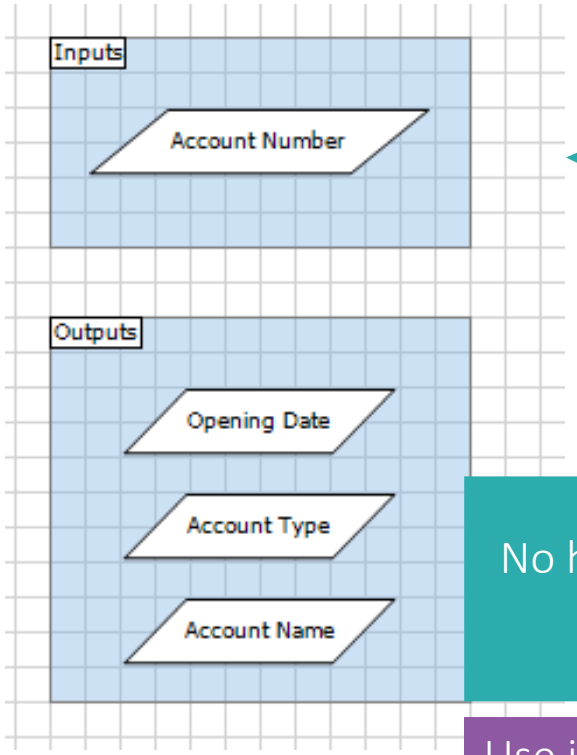
Preconditions:
On the proposal page

Endpoint:
On the confirmation page

Parameter	Direction	Data Type	Description
Customer Number	In	Text	Full 8 digit customer number
Customer Lastname	In	Text	Last name of customer. Used to verify customer number
Product Name	In	Text	Product name (ISA, Saving, Current or Loan)
Deposit	In	Number	Deposit amount
Confirmation Code	Out	Text	System generated code provided on successful submission

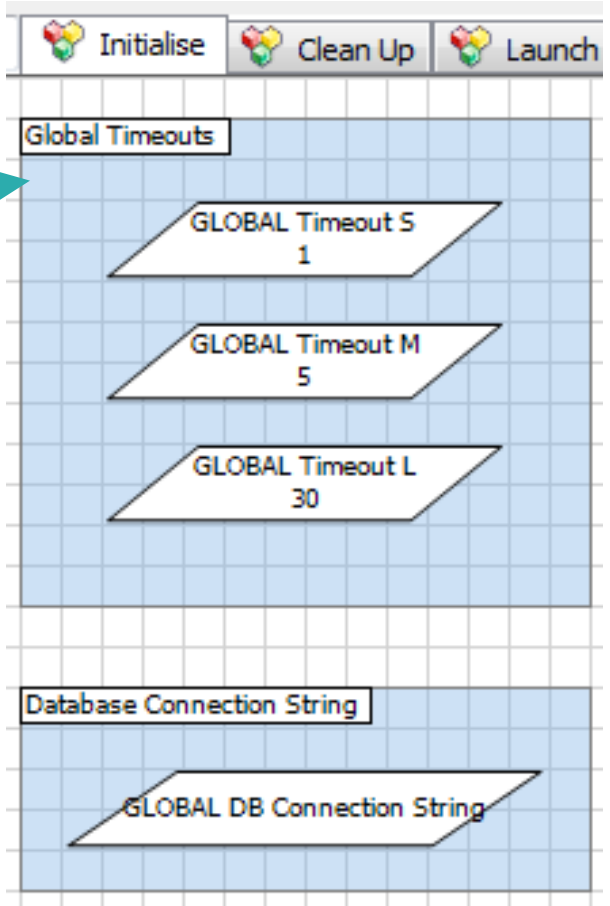
Development Best Practice - Objects

Data Items

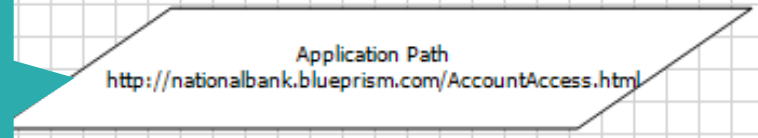


Group data item relevant blocks

Store global data items on the Initialise Page



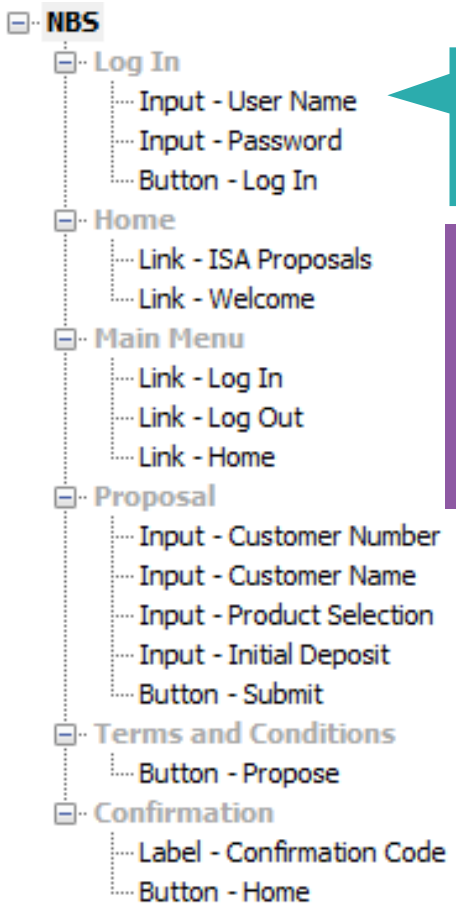
No hardcoded data in data items



Use input parameters and have the process provide the values. If need be, the process can consume an environment variable for parameters that may change.

Development Best Practice - Objects

Application Modeller



Adheres to local naming convention

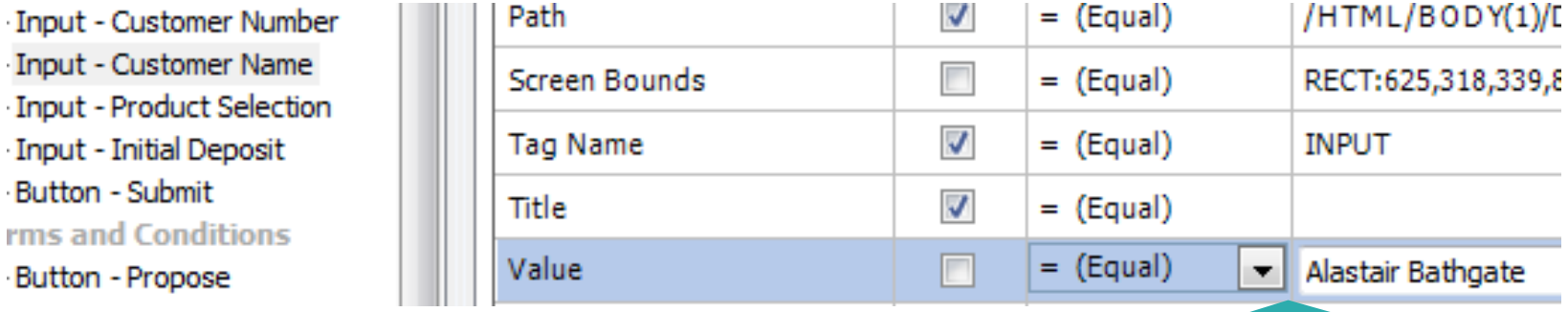
Typically this is {element type} – {element name} e.g.
Button – Submit
Input – Account Number
List – Products

Logically Laid Out

Create sections for each part of the screen.
Makes support easier and mitigates the risk of incorrect elements being re-spied

Development Best Practice - Objects

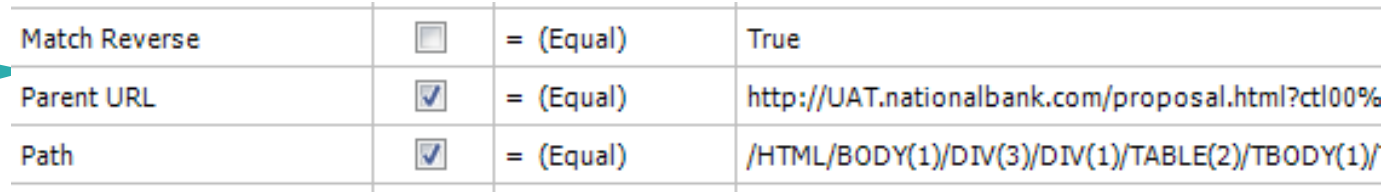
Application Modeller



No customer data

Customer data captured within element attributes could breach data security.

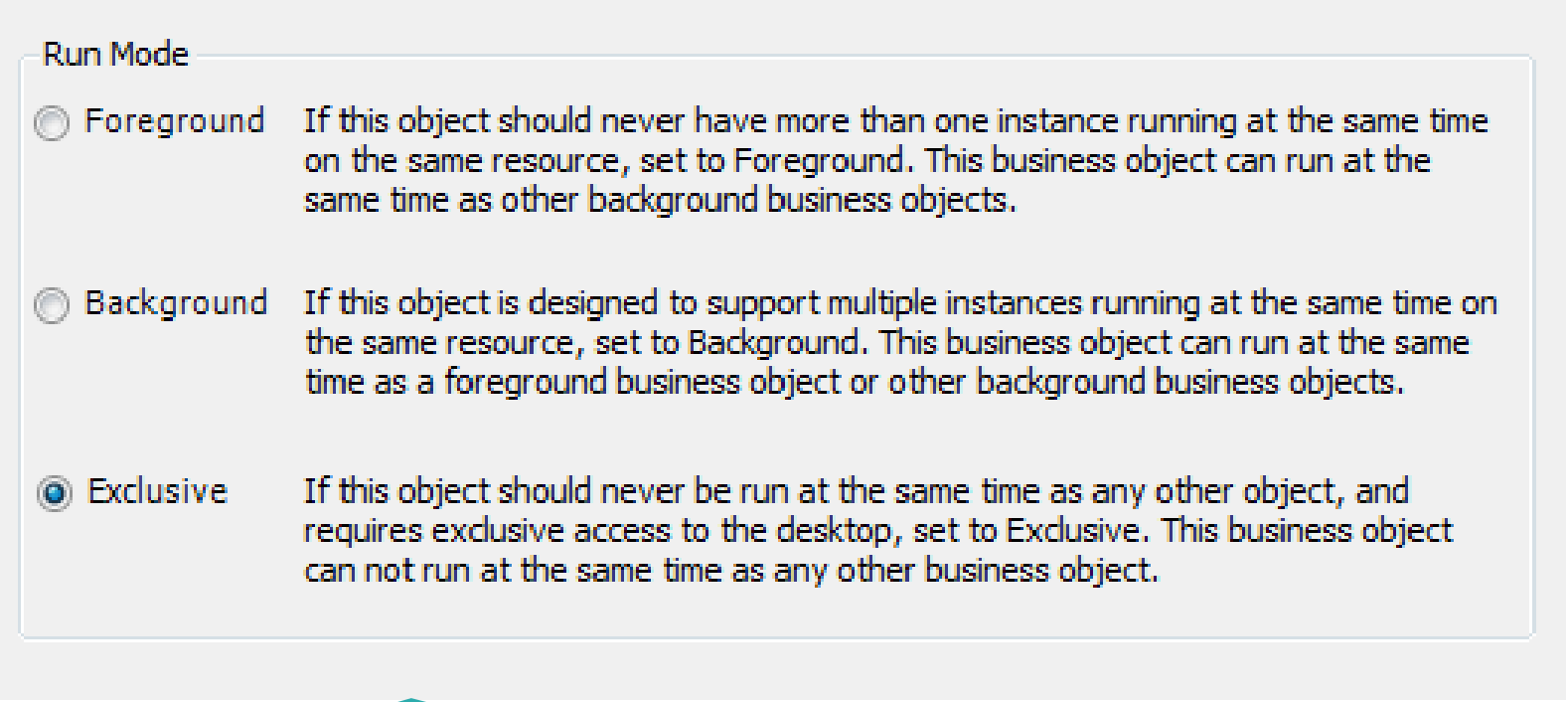
No environment specific data



Environment specific data will cause the process to fail when migrated. If required make the value dynamic.

Development Best Practice - Objects

Exposure



Always set an objects exposure

Process Layer

Best Practices

Process Solution

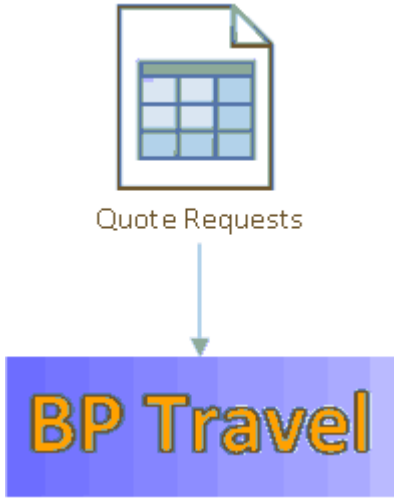
Creating a Process

The process I am going to build for this session is the Create Quotes process. This business process has the following basic steps:

- Get work requests from an Excel Workbook
- For each request, perform some navigation and an update in an application called BP Travel

Before we start we have made sure of the following:

- The design is complete:
- The PDI document for this process can be downloaded alongside this tutorial.



The objects required have already been built. The objects I am using are:

- BP Travel – Basic Actions
- BP Travel – Quotes
- MS Excel VBO

The Work Queue and Environment/Session variables in the design have been created

- I am using a Work Queue called ‘Quote Requests’

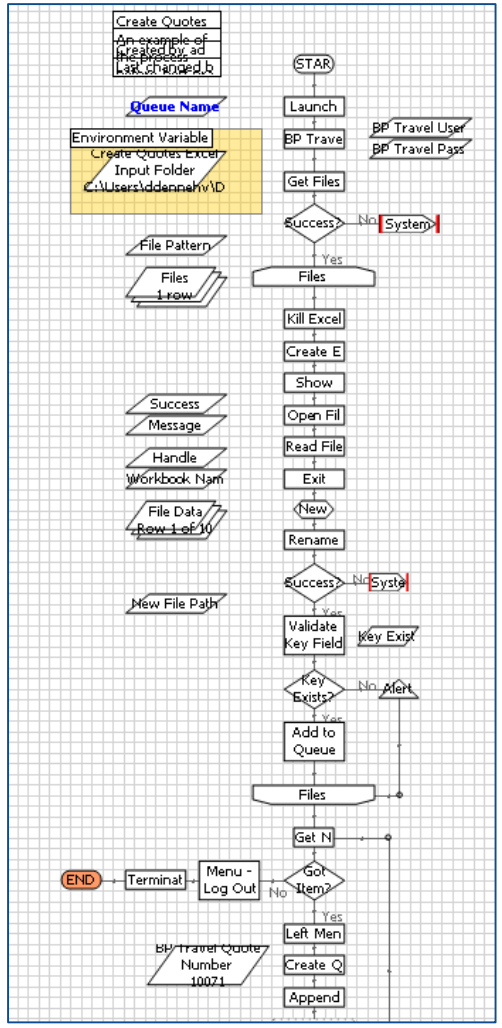
Process Solution

Bad Practice

On the right is the Main Page of the first attempt at creating the process.

It needs improving because:

- There is just one large difficult to read page
- The 'look and feel' of my process is bespoke to me, this will make it difficult for others to support.
- There is no exception handling logic
- If a minor issue occurs in a system the process will terminate
- The process is difficult to control whilst it is running because there are no session variables



How can we improve the process?

- **This process needs improving to make it more granular, more robust, and easier to control.**

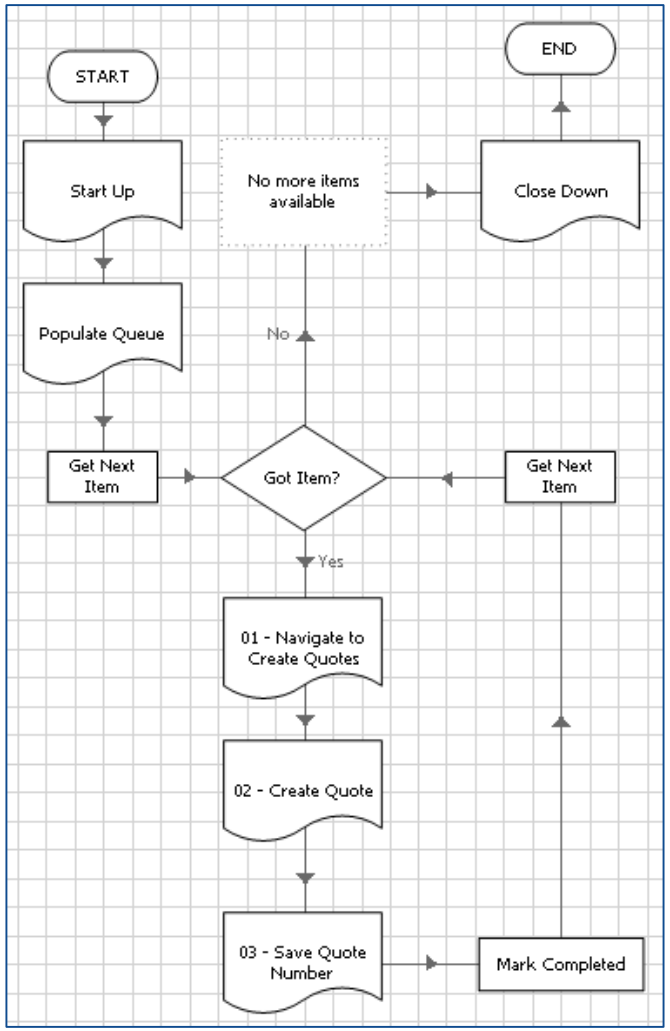
Process Solution

A simple Main Page – using Sub-Pages

On the right is the Main Page of my process after I have moved most of the process logic into Sub-Pages

New Sub-Pages:

- Start Up – A sub-page to that launches and logs into applications
- Close Down – a sub-page that logs out and exits applications
- Populate Queue – a sub-page that loads work from source into a Blue Prism Work Queue
- Work Pages – my process has a few 'work' sub-pages for navigating and updating the systems it uses



Why is this better?

- **It is now easier to quickly understand what the process does just by looking at the Main Page.**

Process Solution

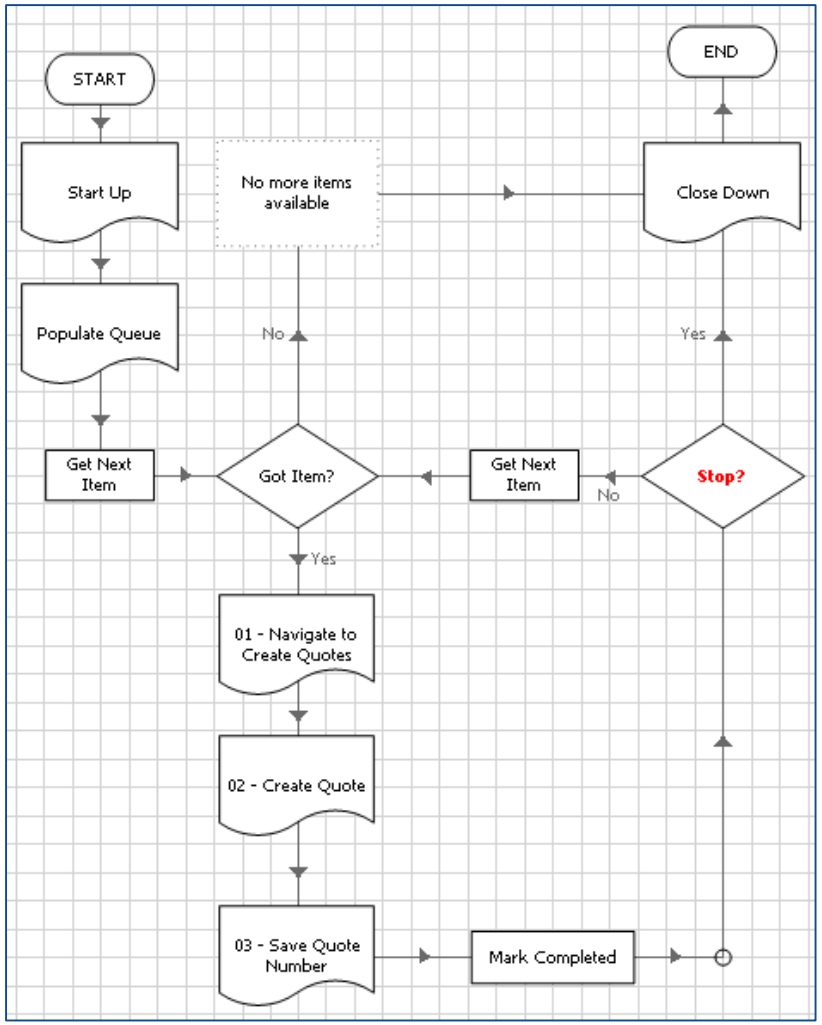
Adding Control – The Stop? decision

On the right is the Main Page of my process after adding a Stop? decision stage.

The Stop? decision evaluates session variables after each Work Queue item has been worked.

Stop? Decision session variables:

- Stop ASAP flag – flag session variable to indicate if the process should stop after working the current case
- Stop Time – time session variable that is set to the time after which the process should stop working
- Stop After Items– number session variable indicating how many more items to work before stopping



Why is this better?

- **The Controller can now easily alter the running of a process session from within Control Room.**

Process Solution

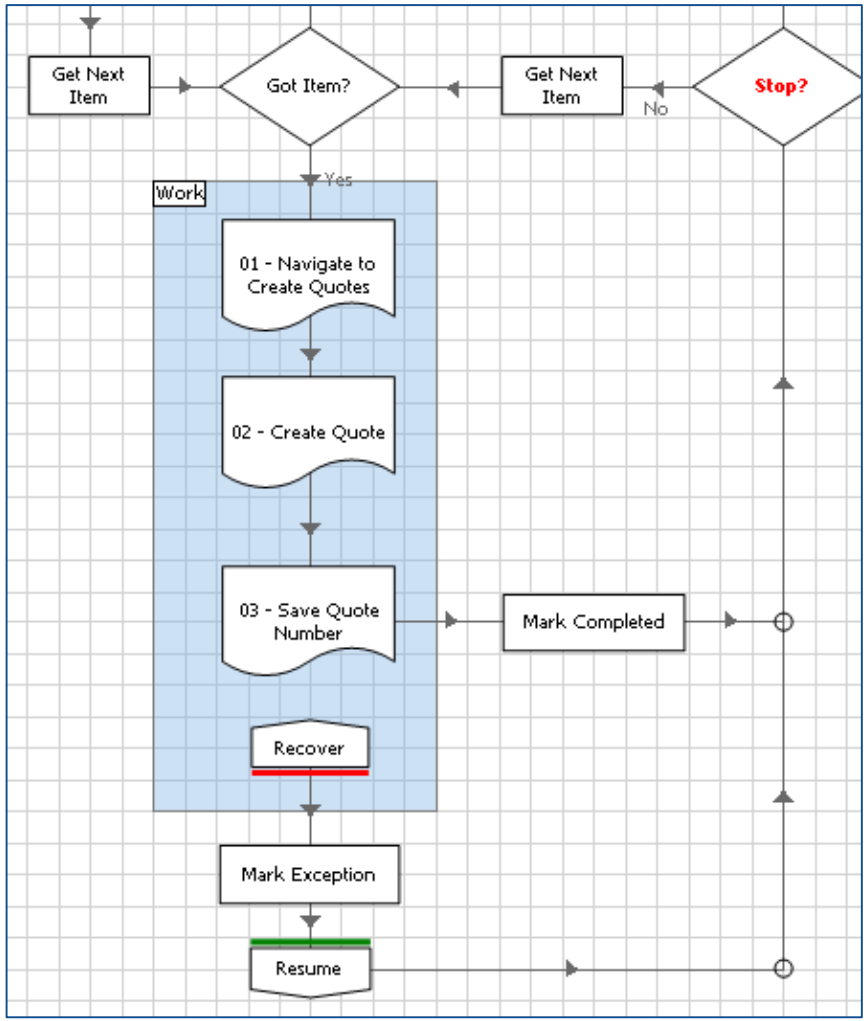
Robustness – Main Page Exception Block

If an exception ‘bubbles up’ to our Main Page from a Sub-Page we don’t want the process to terminate.

Main Page Block

In the flow shown on the right I have added the following:

- A block around the main work interface sub-pages, with a Recover stage..
- A Mark Exception action stage, that marks the current Work Queue item as an exception.
- A Resume stage so that the flow can continue on and attempt to work the next Work Queue item.



Why is this better?

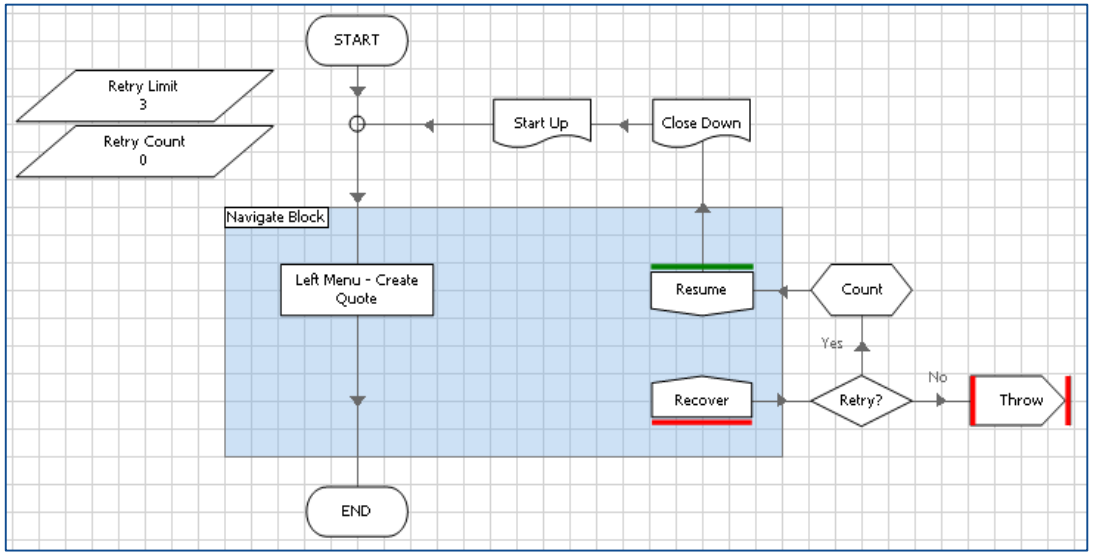
- **All Work Queue items should have a result set by the process, either Completed or Exception**
- **The process should not just terminate if one case has a problem.**

Process Solution

Robustness – Sub-Page Retry Loops

If an exception occurs on a sub-page it could be a one-off issue (i.e. a network timeout) that would be fixed if the process flow simply tried again.

- **Sub Page Retry Loop**
- **On all my 'work' Sub-Pages I have done the following:**
  - Catch any exceptions with a Recover Stage
  - Evaluate the exception with a Retry? decision
  - If there has been less than 3 attempts and the exception is a type I want to retry, loop around to try again
  - If trying again, tidy up the system being used, here we are simply restarting it
  - If not trying again, 'throw' the exception up to the main page



Why is this better?

- **For one-off application errors, or systems that have reliability issues, the retry loop improves the chances of a Work Queue item being successfully worked.**

Process Solution

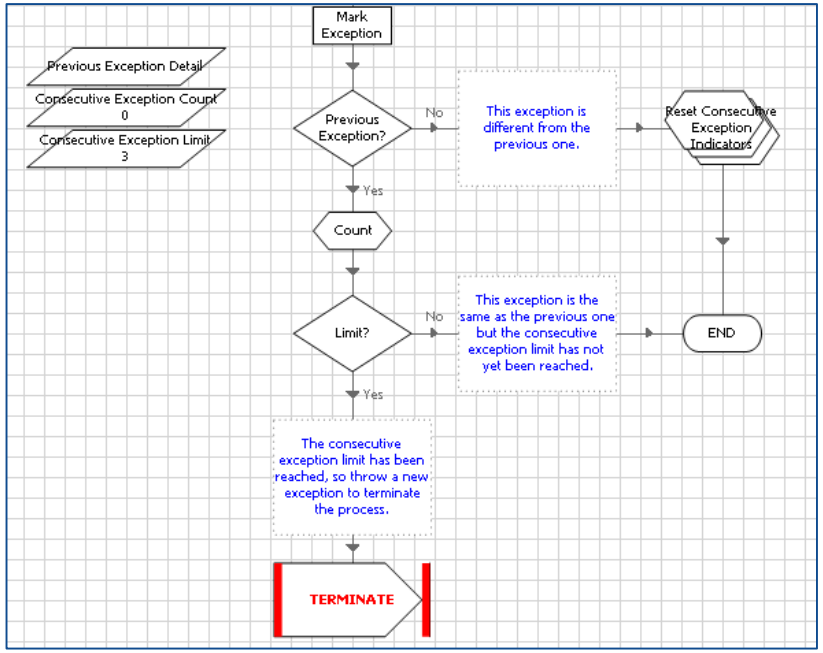
Robustness – Concurrent Exceptions

If the same exception occurs for every Work Queue item a process attempts to work, it is best to terminate the process so that a Controller can investigate the issue.

Mark Item As Exception

The Main Page on the right calls a Sub-Page that does the following:

- Compares the current exception to the previous one
- Terminates the process (‘throws’ the Exception) if the same exception occurs repeatedly for configurable number of concurrent cases.



Why is this better?

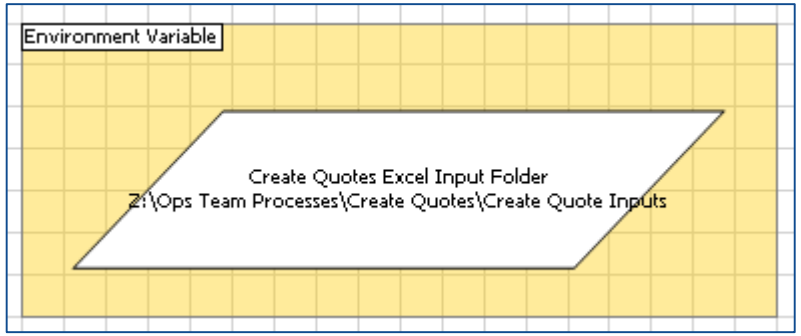
- **If a system is not available or has changed we don’t want our process to attempt every Work Queue item and mark all items as Exceptions.**

Process Solution

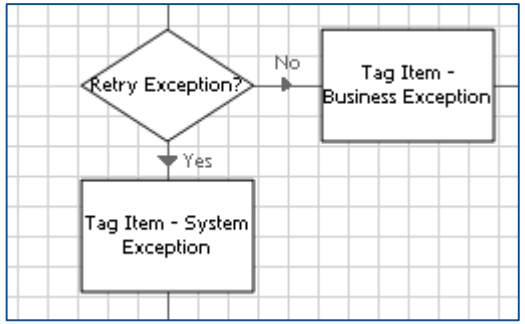
Other Improvements

The final improvements to my process is to ensure that I use Environment Variables to store any configurable process data, and to use tags to capture any MI I require.

- **Environment Variables**
  - Environment Variables should be used to store configurable information such as:
  - Network Paths
  - Email, database, or web service configuration
  - System configuration such as URLs



- **Work Queue Tags**
  - Work Queue tags are reported on the Blue Prism Performance Report
  - All Exception Work Queue items should be tagged as either 'System Exception' or 'Business Exception'
  - Other tags are useful to get total numbers of different work or case types



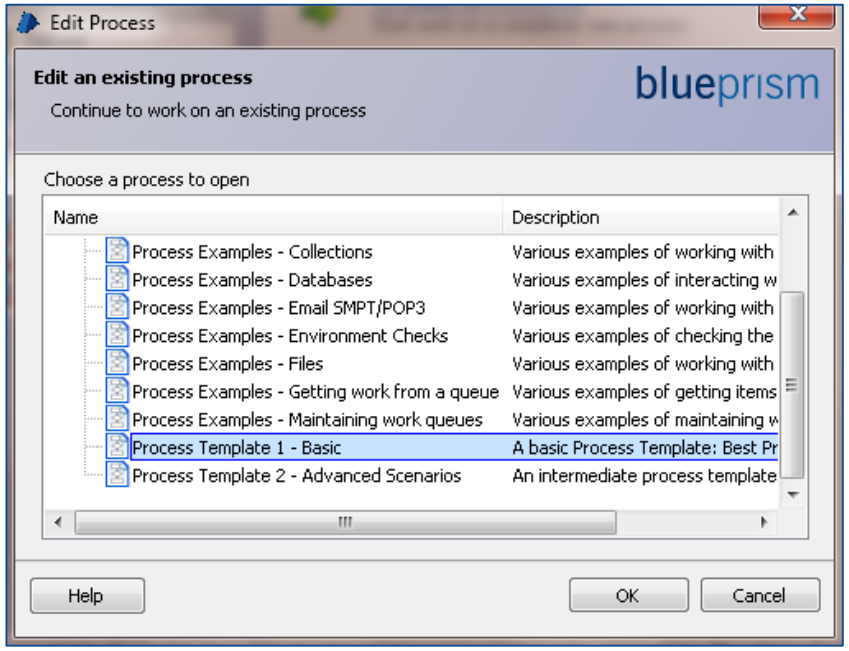
Process Solution

Blue Prism Process Templates

Familiarise yourself with Process Templates

If you look at the process created in this tutorial and compare it to a Blue Prism Process Template (available on the Portal), you will see that they look extremely similar.

Our Process Templates have the same 'look and feel' and all the same features to make processes robust and easier to understand and support.



Always use Process Templates

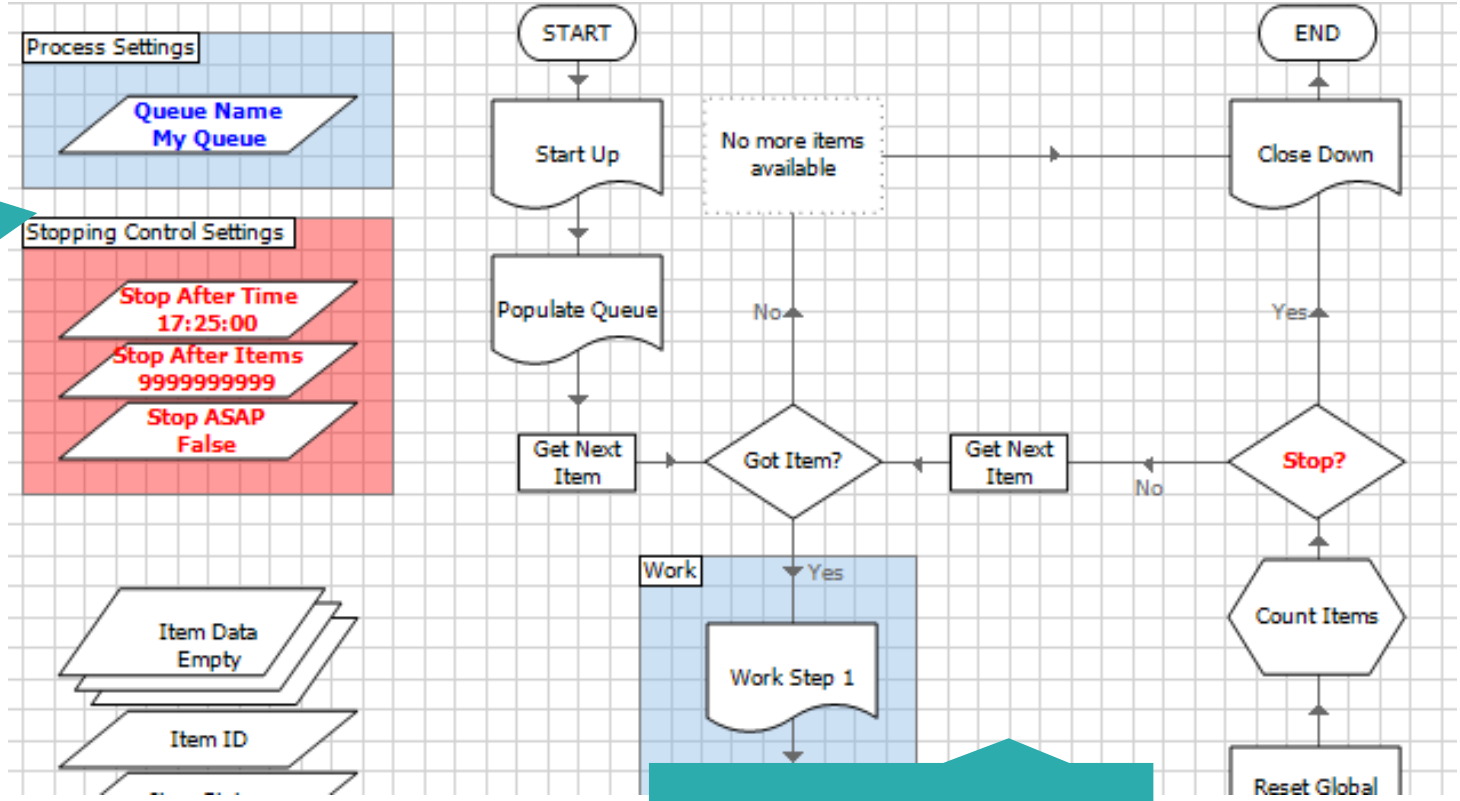
- You should always use Blue Prism Process Templates as the starting point when creating a new process.
- Save time, the Process Templates have already made a start on your process logic
- Ensure best practice, Process Templates already contain all the lessons of this Tutorial.

Development Best Practice – Process

Structure

Use standard Blue Prism templates or templates provided by the local design authority

The standard logic of the templates enables familiarity that makes support easier.



Main page should contain high level process steps

All process detail and decision should be divided into logical sub pages

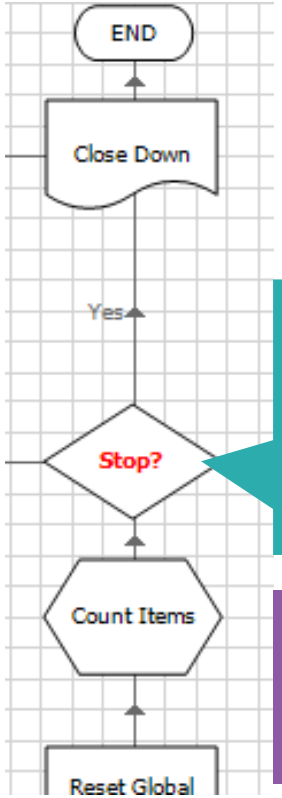
Development Best Practice – Process

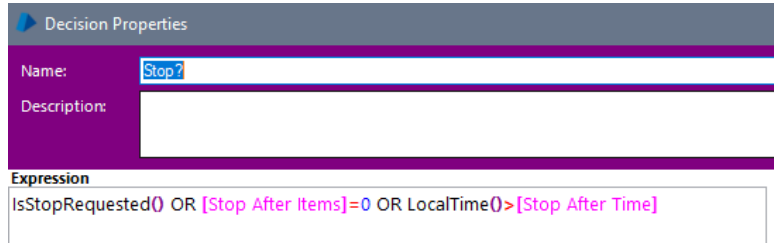
Stopping existing sessions

If the queue controller decides that existing sessions must be stopped in order to reach the target, it will iterate through the sessions and choose the oldest sessions (ie. the sessions that were started at the earliest times) to send a stop request to.

Once a controller sends a stop request to a session, the session is marked as Stopping. In this state, it is still 'running' - i.e.. it is contributing to the 'Active Sessions' count in the queue, and it is not available to run further sessions on.

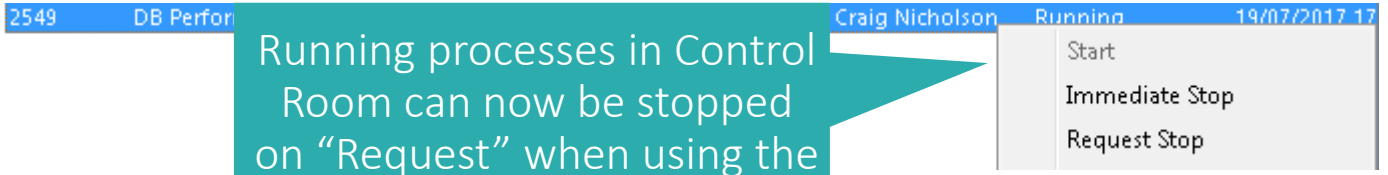
In order to stop correctly, the worker process must check the IsStopRequested() function regularly, cleaning up its environment and exiting the process when the function indicates that a stop has been requested.





Use standard Blue Prism templates or templates provided by the local design authority

The standard logic of the templates enables familiarity that makes support easier.

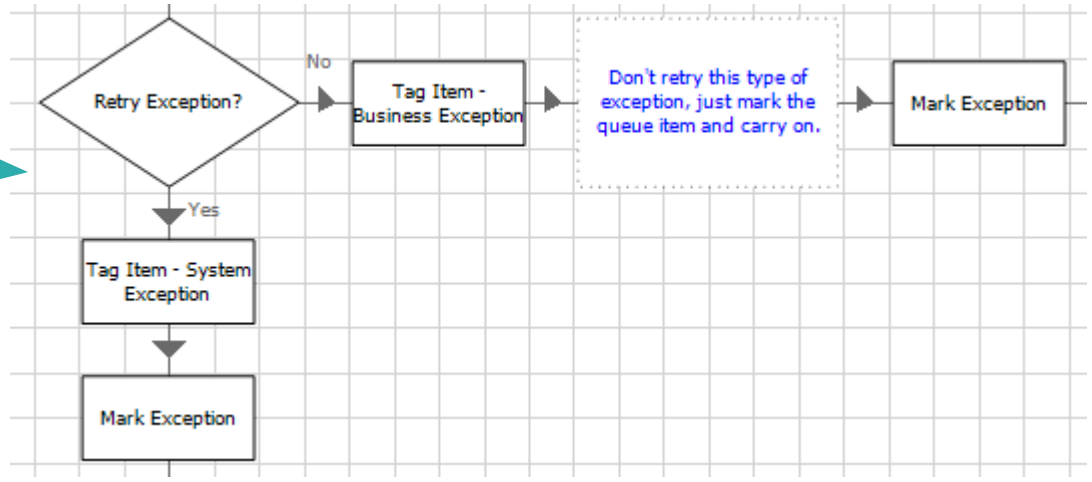


Running processes in Control Room can now be stopped on “Request” when using the IsStopRequested function in the process

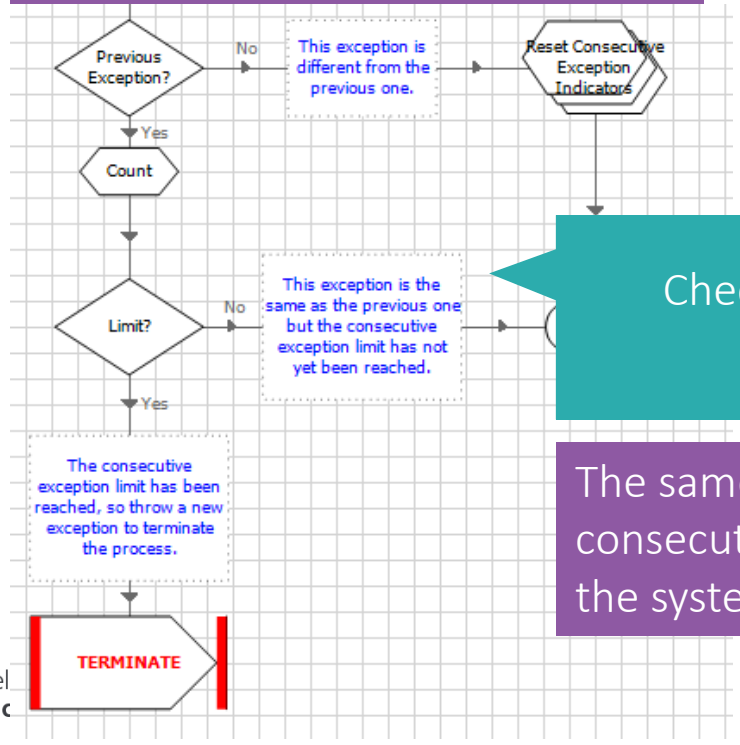
Development Best Practice – Process

Retries

Correct use of retries



If the work queue permits retry system exceptions. Do not retry business exceptions



Check for consecutive system exceptions

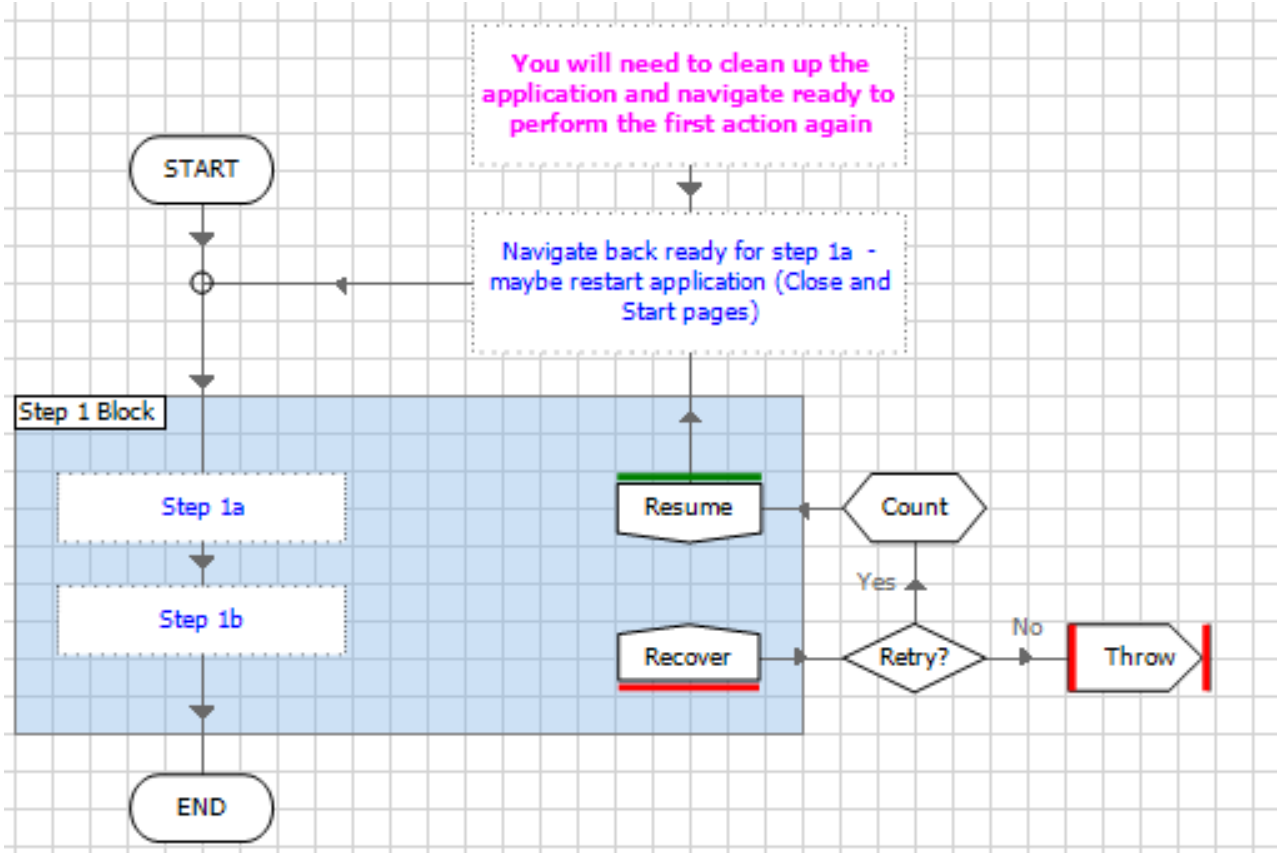
The same system exception across consecutive cases could be a sign that the system has changed.

Development Best Practice – Process

Retries

Correct use of retries

Where possible retry system exception within the process. This may require special navigation or even a restart of the system



Process Solution - Recap

What have we Learnt?

- ✓ A process should be broken down into pages that are easy to view and support
- ✓ A Stop? decision makes processes more controllable
- ✓ For best practice exception handling, your process should have the following:
- ✓ An exception block on the Main Page to handle exceptions that ‘bubble up’
- ✓ Retry loops on work sub pages – to attempt system errors again
- ✓ A check to ensure the same system exception is not occurring on every case
- ✓ Environment variables should be used to make the process configurable from System Manager
- ✓ Tags are an easy way to store Management Information
- ✓ If you have not already, download and use Blue Prism Templates from the Blue Prism Portal
- ✓ With the templates are instruction documents that have additional detail about the features within the templates.

Development Best Practices

Naming Conventions & Standards

Naming Conventions / Standards

Objects

Objects = APP NAME + SCREEN NAME

Example:

WebSTP – Search Page

WebSTP – Memo Page

A “Basic Actions” object is also recommended to allow for launching, terminating and other functions which are universally applicable within the action

Example:

WebSTP – Basic Actions

Launch

Log Out

Refresh Page

Open Print Dialogue

Terminate

Where there are multiple versions of the same application (not object) within the company

Example:

WSO (v3.5) – Basic Actions

WSO (v4.2) – Basic Actions

Naming Conventions / Standards

Process

Logical description - No peoples names, version numbers, application names (unless it's appropriate)

01 – Create Quote – Populate Queue

First process that should be run

02 – Create Quote - Main

Does the rest of the work (*calls each of the websites, which are sub processes*)

Create Quote - Gas

Create Quote - Electricity

Create Quote - Telephone

Are not numbered as they are not executed in any specific order/ad-hoc by the main process depending on what is on the queue

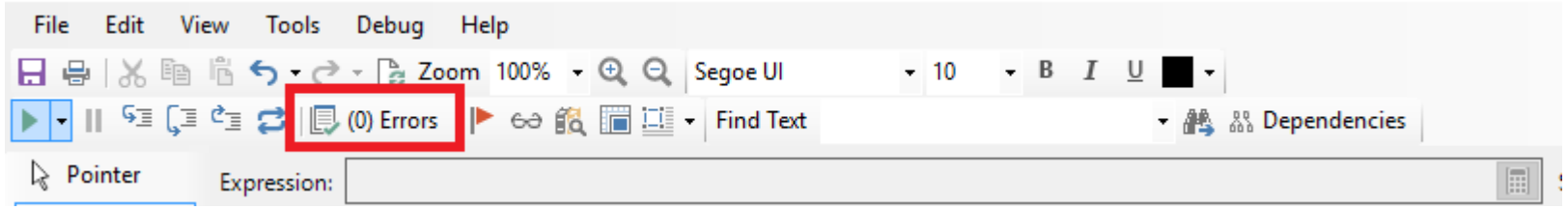
In the example all of the processes can be put into a Create Quote group

Development Best Practices

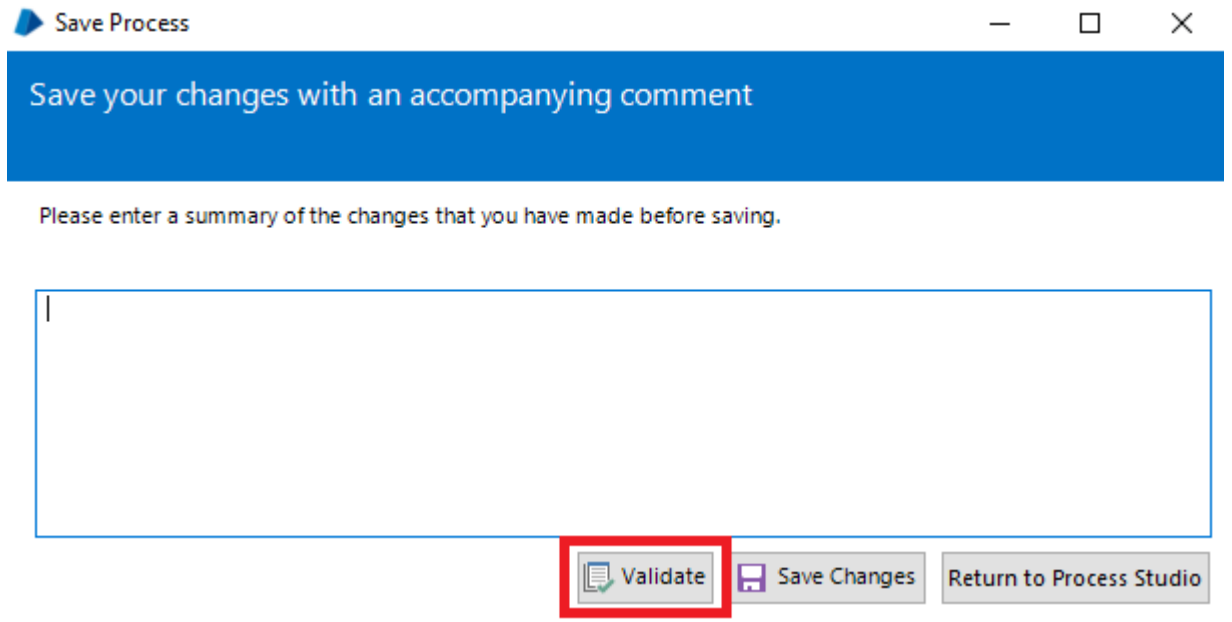
Further Best Practice

Development Best Practice

Further Best Practice



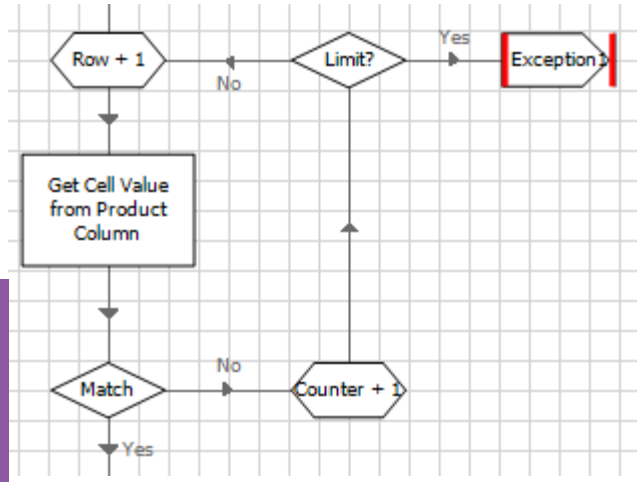
Always check for errors or validate when saving.



Development Best Practice

Further Best Practice

Manufactured Loops have defence mechanism



When creating your own loops always create a counter and throw an exception if an excessive limit is reached.
This will prevent the flow ever entering an infinite loop.

Stage local adheres to local design control

Development Best Practice

Interface-Specific Best Practice