blueprism

Software Robots - the Virtual Workforce

Introduction to Surface Automation

TRAINING GUIDE

Version: 5.0.2

For more information please contact:

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

www.blueprism.com

Contents

The training materials and other documentation ("Training Materials") provided by Blue Prism as part of the training course are Blue Prism's Intellectual Property and Confidential Information. They are to be used only in conjunction with the Blue Prism Software which is licensed to your company, and the Training Materials are subject to the terms of that license. In addition, Blue Prism hereby grants to you a personal, revocable, non-transferable and non-exclusive license to use the Training Materials in a non-production and non-commercial capacity solely for the purpose of training. You can modify or adapt the Training Materials for your internal use to the extent required to comply with your operational methods, provided that you shall (a) ensure that each copy shall include all copyright and proprietary notices included in the Training Materials; (b) keep a written record of the location and use of each such copy; and (c) provide a copy of such record to Blue Prism on request and allow Blue Prism to verify the same from time to time on request.

For the avoidance of doubt, except as permitted by the license or these terms, you cannot (a) copy, translate, reverse engineer, reverse assemble, modify, adapt, create derivative works of, decompile, merge, separate, disassemble, determine the source code of or otherwise reduce to binary code or any other human-perceivable form, the whole or any part of the Training Materials; (b) sublease, lease, assign, sell, sub-license, rent, export, re-export, encumber, permit concurrent use of or otherwise transfer or grant other rights in the whole or any part of the Training Materials; or (c) provide or otherwise make available the Training Materials in whole or in part in any form to any person, without prior written consent from Blue Prism.

© Blue Prism Limited, 2001 - 2015

All trademarks are hereby acknowledged and are used to the benefit of their respective owners.
Blue Prism is not responsible for the content of external websites referenced by this document.

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

This course is intended for Blue Prism users who have successfully completed the Blue Prism Foundation course. The topics in this course are fairly advanced and are not suitable for novices.

The term ***thin client*** is used to describe an application that does not run on the local machine and uses a client/server architecture. A thick client application is one that is installed directly on the local machine, like Blue Prism itself. Although some consider web applications as ***thin client***, here we are looking at applications that operate on a remote machine and are presented to the user via virtualization software like Citrix.

With thin client, Blue Prism cannot use its normal integration techniques because the target application is virtual and there is very little that spy-modes like Win32 and AA are able to detect. An analogy could be to think of the surface of a thick client application as having a “landscape” made of buttons, fields, and checkboxes, etc., whereas the surface of a thin client application is smooth and featureless.

In order to integrate with a thin client application we must redefine the application surface as a series of rectangles or ***regions***.

2. Regions

Spying a thin client application using Win32 (or AA) will yield very few distinct elements from which to build an application model, and often the only element available is the main window. In the absence of available elements, interfacing with a thin client requires the use of ***regions***. A region is simply rectangular area of an element that we define in order to direct the user's (i.e., Blue Prism's) contact with the application.

To learn the basics, we will first create an interface to the MS Windows ***Calculator*** application. We will create a simple action that will perform a basic calculation and interpret the result.

2.1. Introductory Exercises

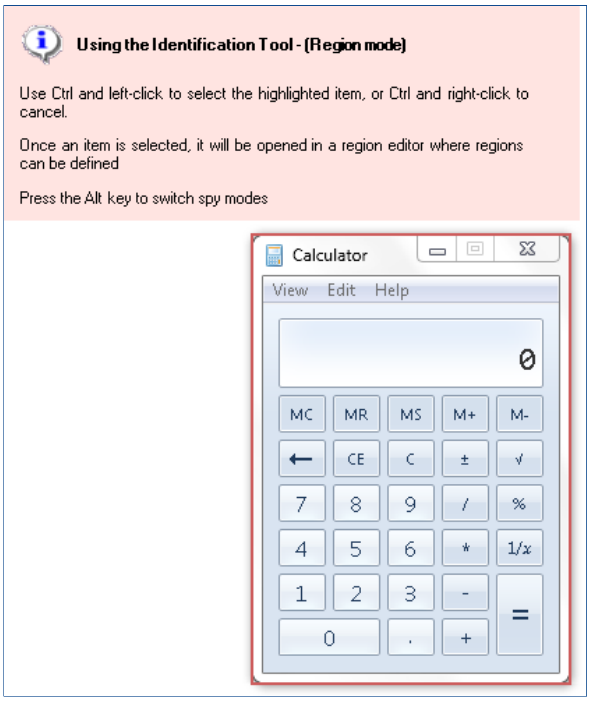
Exercise 2.1.1 Calculator Business Object

- Create a Business Object named "Calculator".
- Use the Application Modeler Wizard to describe the application. The executable path is ***C:\Windows\System32\calc.exe*** and we want to ***Launch*** the application rather than Attach.
- In Application Modeler, rename the default "Element 1" to ***Calculator Screen***.

Now we have our Business Object, we can create some regions, and to do this we will use the Blue Prism ***Region Editor***. The Region Editor provides us with a screenshot of our application on which we can define our regions.

Exercise 2.1.2 Region Mode

- Launch the application from Application Modeler if you haven't already done so.
- Select the Calculator Screen element and click the Identify Element button.
- Press the ALT key to cycle through the different spy modes, stopping at ***Region*** mode.



Using the Identification Tool - (Region mode)

Use Ctrl and left-click to select the highlighted item, or Ctrl and right-click to cancel.

Once an item is selected, it will be opened in a region editor where regions can be defined

Press the Alt key to switch spy modes

Figure 1: Region Spy Mode

- Move the mouse over the application to see that it highlights elements much in the same way as Win32.

Key Point

- Although we can spy Calculator as normal, for this exercise we are going to pretend that we can't and treat it as a thin client application

- Using the identification tool select the entire Calculator application (as above) and CTRL+left-click to open the Region Editor.

2.2. Region Editor

The Region Editor displays a screenshot of the element that was spied using the region spy mode. Its purpose is to provide us with a "drawing area" on which we can create and edit regions.

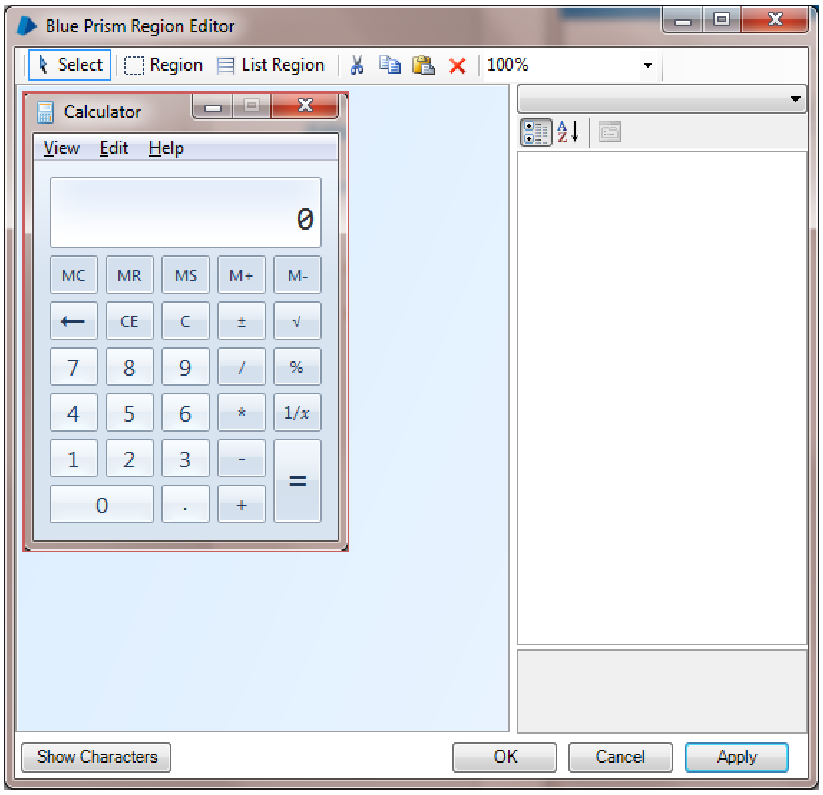


Figure 2: Region Editor

The region Editor let you create two types of region: the default is a basic rectangle and the other is a more complex "list" of rectangles which we will look at later.

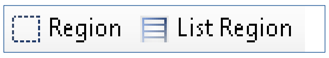
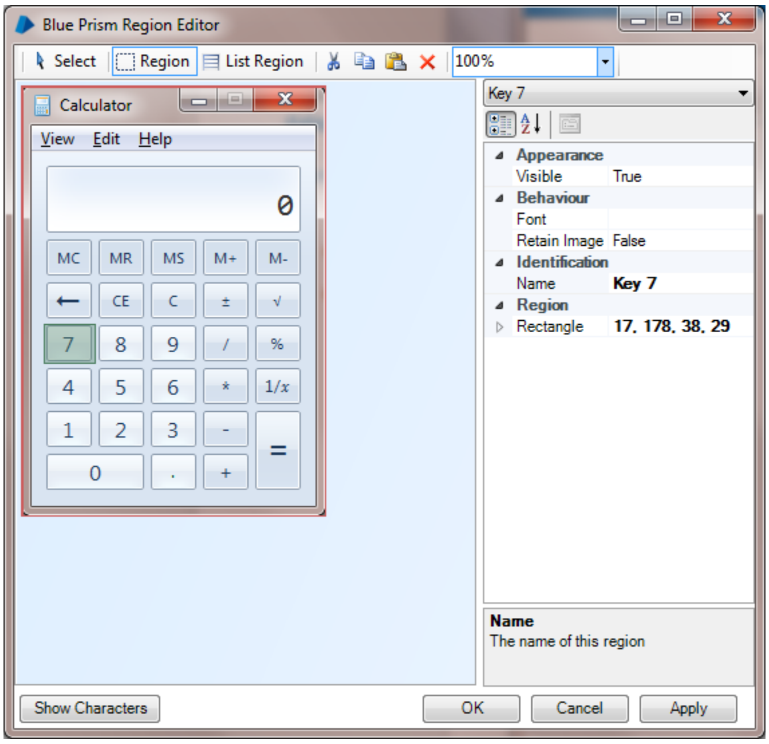


Figure 3: Region Type Toolbar Buttons

Exercise 2.2.1 Creating a Basic Region

- Press the Region toolbar button to select the default region type.
- Drag the mouse diagonally over the 7 button to draw a box on top of the Calculator screenshot. Don't worry about making the region too exact - an approximate fit over the button is fine.
- See how a new region has been created and its properties are displayed in the right-hand panel.

- Change Name property value (under the Identification section) to rename the ***new*** region as ***Key 7***.



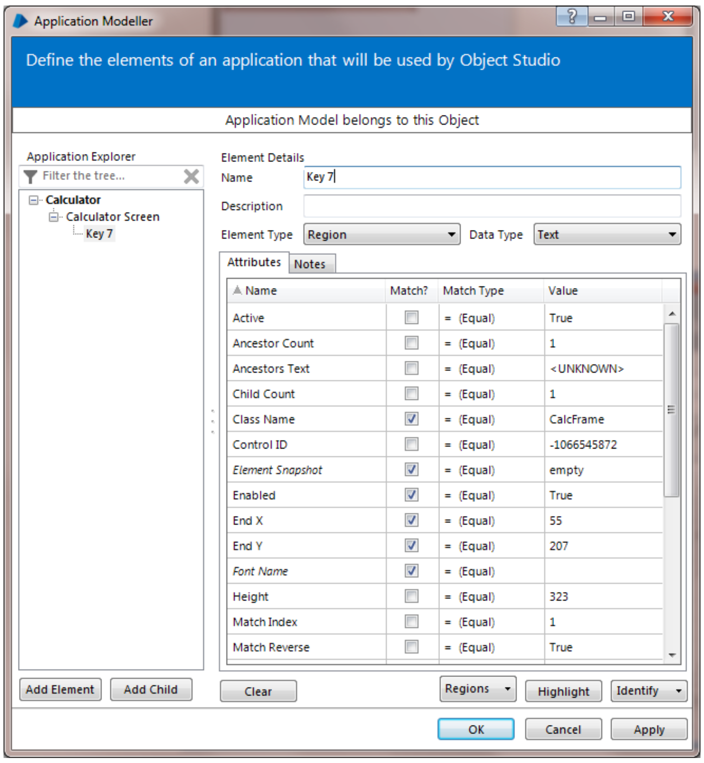
Key 7

Appearance	
Visible	True
Behaviour	
Font	
Retain Image	False
Identification	
Name	**Key 7**
Region	
Rectangle	**17, 178, 38, 29**

Name
The name of this region

Figure 4: Key 7 Region

- Now Click "OK" and return to Application Modeler.
- You will now be able to see the new Key 7 region element we have just created.



Define the elements of an application that will be used by Object Studio

Application Model belongs to this Object

Element Details

Name: Key 7

Description:

Element Type: Region   Data Type: Text

Name	Match?	Match Type	Value
Active	☐	= (Equal)	True
Ancestor Count	☐	= (Equal)	1
Ancestors Text	☐	= (Equal)	<UNKNOWN>
Child Count	☐	= (Equal)	1
Class Name	✓	= (Equal)	CalcFrame
Control ID	☐	= (Equal)	-1066545872
Element Snapshot	✓	= (Equal)	empty
Enabled	✓	= (Equal)	True
End X	✓	= (Equal)	55
End Y	✓	= (Equal)	207
Font Name	✓	= (Equal)	
Height	☐	= (Equal)	323
Match Index	☐	= (Equal)	1
Match Reverse	☐	= (Equal)	True

Figure 5: Application Modeler

Notice how the region is a child of the Calculator Screen window element we selected when we first spied Calculator.

Key Point

- Region elements are captured as children of the application element initially selected to open the Region Editor.

Exercise 2.2.2 Highlighting a Region

Once captured, a region can be highlighted, much like any other element of the Application Model.

- Launch the application from Application Modeler if you haven't already done so.
- Select the Key 7 element and press the "Highlight Element" button.
- See that the element is highlighted on the Calculator application.

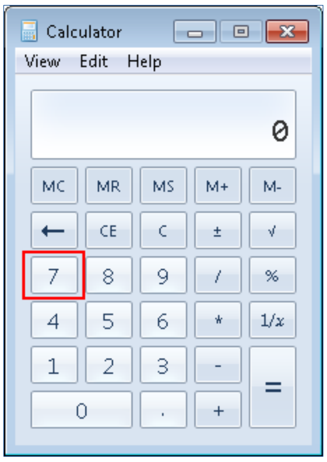
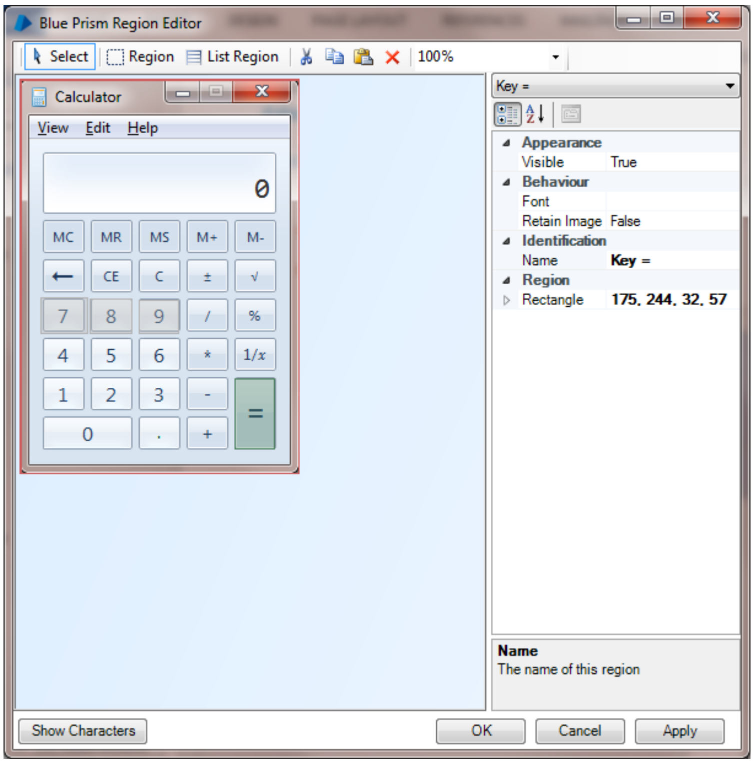


Figure 6: Region Highlighting

Exercise 2.2.3 Creating More Regions

In this exercise, we'll make some more regions to enable us to use more of Calculator's buttons.

- Return to Application Modeler and select the ***Calculator Screen*** element.
- Press the ***Region*** button (to the bottom right of Application Modeler) to open Region Editor.
- In the same way that you created the Key 7 region, create regions for the 8, 9, * (multiply) and = (equals) buttons.
- When completed your Region Editor should look something like this:



Appearance	
Visible	True
Behaviour	
Font	
Retain Image	False
Identification	
Name	Key =
Region	
Rectangle	175, 244, 32, 57

Name
The name of this region

Figure 7: Region Editor with Multiple Regions

You can select a region using either the Select tool or by selecting the region name from the drop-down list in the top-right corner.

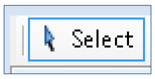


Figure 8: Select Toolbar Button

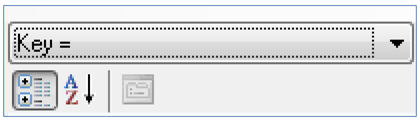


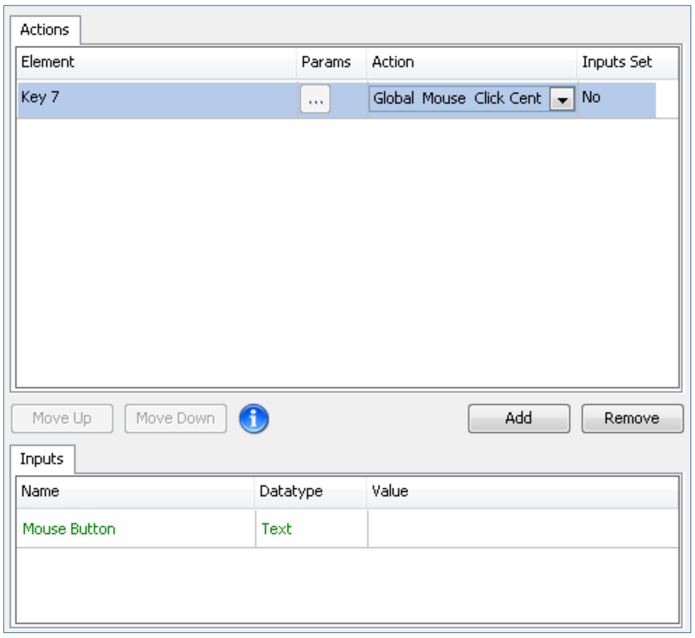
Figure 9: Region Name Drop-down List

2.3. Execute Actions

Just like with other Application Modeler elements, Blue Prism provides functionality to use regions as a way of interacting with an application. The key actions supported by the regions we have created are ***Global Mouse Click*** and ***Global Mouse Click Center***. As their names suggest, these actions are a way of making the mouse click on the region.

Exercise 2.3.1 Global Mouse Click Center

- Add a new page to your Calculator Business Object and add a new Navigate stage to the diagram.
- Open the properties of the Navigate stage and drag in the ***Key 7*** region from the left-hand element list.
- Select ***Global Mouse Click Center*** from the list of available actions.
- OK the properties form and link up your diagram to check that the 7 button can be pressed.



Element	Params	Action	Inputs Set
Key 7	...	Global Mouse Click Cent	No

Name	Datatype	Value
Mouse Button	Text	

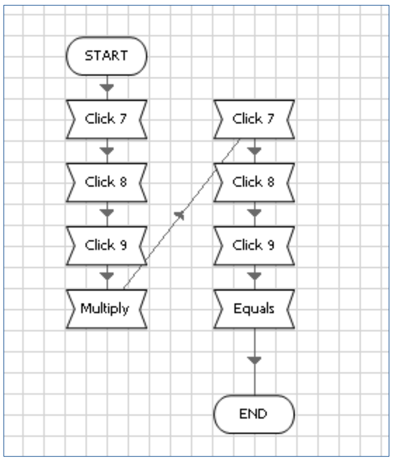
Figure 10: Global Mouse Click Center

- Continue working on your page so that it is able to perform the calculation 789 * 789.

  * *Remember to include the Equals button at the end.*

Exercise 2.3.2 Running the Page

Your page should now look something like this:



- Launch Calculator using the Launch button if you haven't already done so.
- Press the Run button (or press F5) to run your page.
- See that the mouse clicks the buttons to perform the calculation.

3. Character Matching

Blue Prism is capable of ***character matching***, the ability to discern text characters from an image. In brief, the technique involves creating a font definition, effectively a template of characters the application uses. This template can then be used to inspect a screenshot of the application, and by finding matching patterns of pixels, characters, and words can be deduced.

3.1. Identifying Fonts

In the previous exercises, we created a Business Object to press the buttons of Calculator to work out an expression. We now want to read the result of that expression, but without using the standard spying techniques.

Key Point

- To reiterate the point, we can spy Calculator as normal but for this exercise, we are going to pretend that we can't and treat it as a thin client application.

Exercise 3.1.1 Calculator Result Region

We want to use Character Matching to interpret the result of our expression, and to do so we need a screenshot of the Calculator display area.

- Launch Calculator and manually key in the numbers 1234567890.
- Open the Region Editor but rather than just pressing the Regions button, select the ***View overlaid on new screenshot*** option from the drop-down list. This will force Region Editor to refresh its screenshot of Calculator so we can see the numbers we just keyed in.

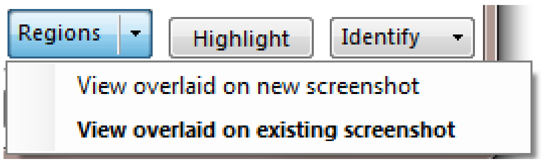
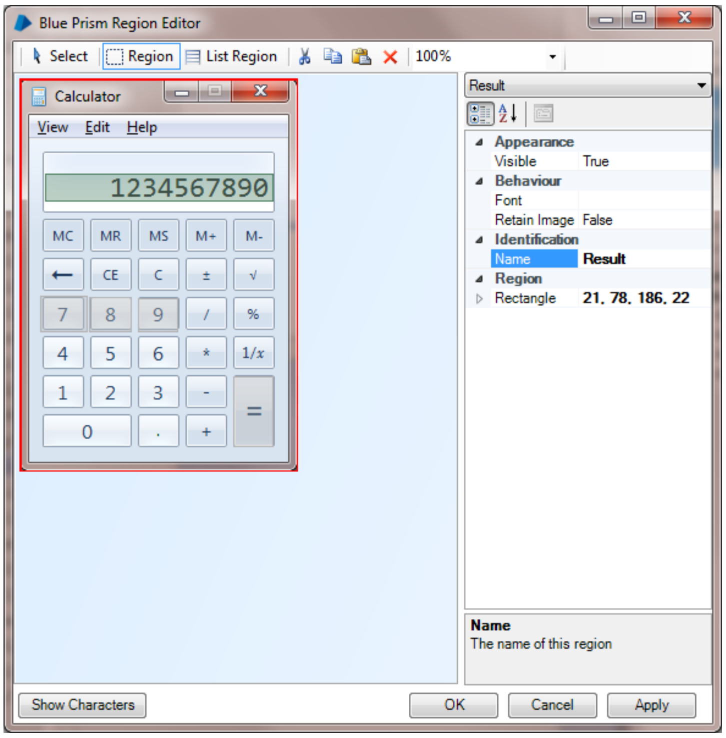

Regions
Highlight
Identify
View overlaid on new screenshot
View overlaid on existing screenshot

Figure 11: View Overlaid on New Screenshot

- Create a new region for the Calculator display area and rename it Result. Your Region Editor should look something like this:



Result

Appearance	
Visible	True
Behaviour	
Font	
Retain Image	False
Identification	
Name	Result
Region	
Rectangle	21, 78, 186, 22

Name
The name of this region

Figure 12: Calculator Result Region

Exercise 3.1.2 Region Font

If a region is to be used for Character Matching, it can be associated with a font in Region Editor. If a font definition is already held in Blue Prism, it will be listed in Region Editor. If the font definition does not yet exist, it can be created, either from a list of system fonts (fonts installed on the PC as part of MS Windows) or as a bespoke font (a font unique to the target application).

In this exercise, we will create a font definition from the list of system fonts.

- With the ***Results*** region selected, find the Font property on the right-hand properties list.
- Select New > Identify System Font, as shown below.

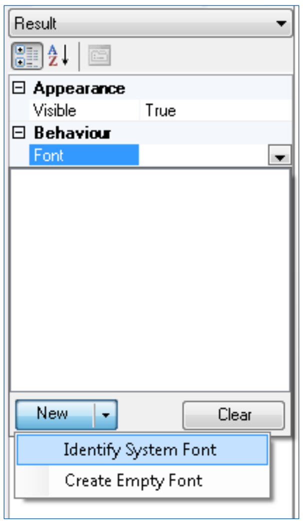


Figure 13: Identify System Font

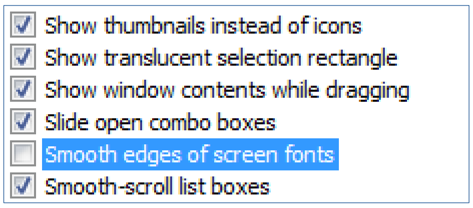
At this stage you may receive a warning message regarding Font Smoothing.

Font Smoothing

Font Smoothing is a Windows setting designed to make fonts easier to read by blurring (anti-aliasing) the edges of text characters. This setting disrupts Character Matching and we need it to be switched off on all Blue Prism PCs.

Exercise 3.1.3 Switching off Font Smoothing

- To switch off font smoothing in Windows 7, navigate to Control Panel > System and Security > System > Advanced System Settings > Performance Settings.
- Select the Visual Effects tab and from the list presented uncheck the ***Smooth edges of screen fonts*** option.

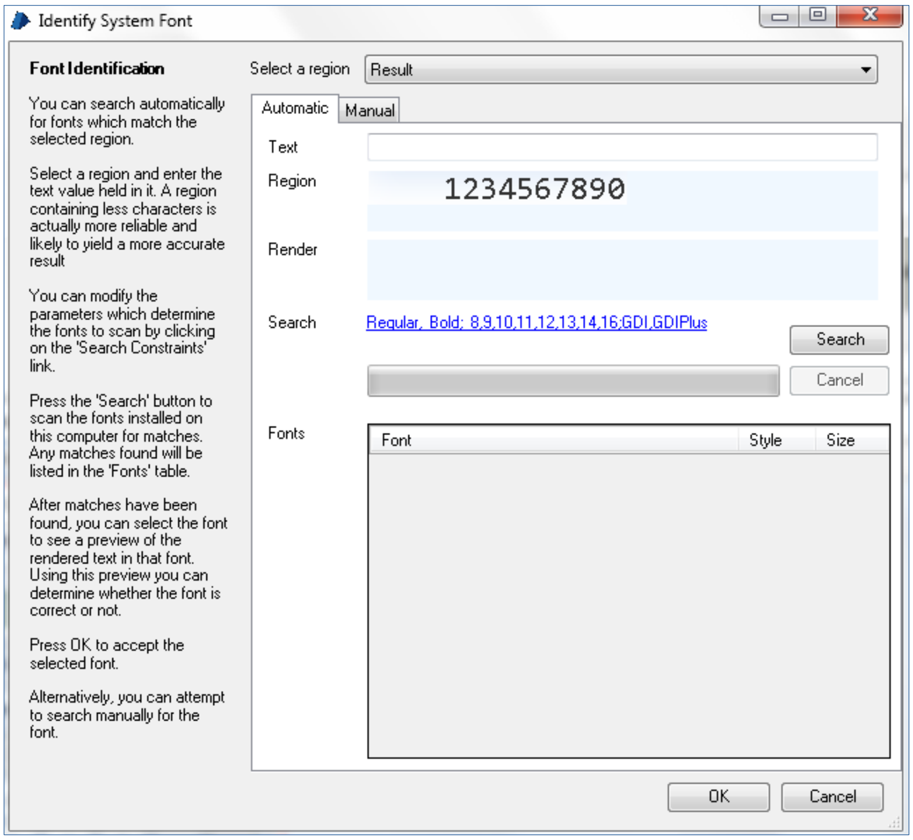


- [✓] Show thumbnails instead of icons
- [✓] Show translucent selection rectangle
- [✓] Show window contents while dragging
- [✓] Slide open combo boxes
- [ ] Smooth edges of screen fonts
- [✓] Smooth-scroll list boxes

Figure 14: Windows 7 Visual Effects

3.2. Identify System Font

With Font Smoothing disabled, selecting ***Identify System Font*** will open the Identify System Font screen, as shown below.



Identify System Font

Font Identification

You can search automatically for fonts which match the selected region.

Select a region and enter the text value held in it. A region containing less characters is actually more reliable and likely to yield a more accurate result

You can modify the parameters which determine the fonts to scan by clicking on the 'Search Constraints' link.

Press the 'Search' button to scan the fonts installed on this computer for matches. Any matches found will be listed in the 'Fonts' table.

After matches have been found, you can select the font to see a preview of the rendered text in that font. Using this preview you can determine whether the font is correct or not.

Press OK to accept the selected font.

Alternatively, you can attempt to search manually for the font.

Select a region: Result

Automatic | Manual

Text:

Region: 1234567890

Render:

Search: Regular, Bold; 8,9,10,11,12,13,14,16;GDI,GDIPlus

Figure 15: Identify System Font Screen

Exercise 3.2.1 Identifying a System Font (Part 1)

Before we can match the characters, we need to first identify the font. Read the instructions on the left to familiarize yourself with how fonts are identified.

- In the Text field, type in the same characters shown in the region, i.e., 1234567890. This tells Blue Prism what characters we are looking to match.
- Click the Search button.
- As it searches, Blue Prism uses each font to create a bitmap image of "1234567890" as it would appear using that font.
- It then compares that image with the region to see if it ***could be** a match*.
- Potential fonts are added to the "Fonts" list and once the search is complete, selecting an item in the Fonts list will show how "1234567890" looks in that font.

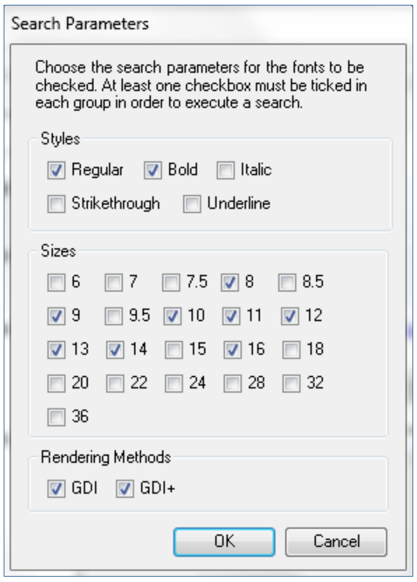
Key Point

- Automatic search requires a degree of experimentation; it may not yield any results or might produce only partial matches.

Exercise 3.2.2 Identifying a System Font (Part 2)

In the previous exercise, the default search settings produce no successful matches; the search was too "narrow" so we will try adjusting the parameters to include larger fonts.

- Click the blue link next to the Search label to bring up the Search Parameters screen.



Search Parameters

Choose the search parameters for the fonts to be checked. At least one checkbox must be ticked in each group in order to execute a search.

Styles

[✓ Regular] [✓ Bold] Italic
Strikethrough Underline

Sizes

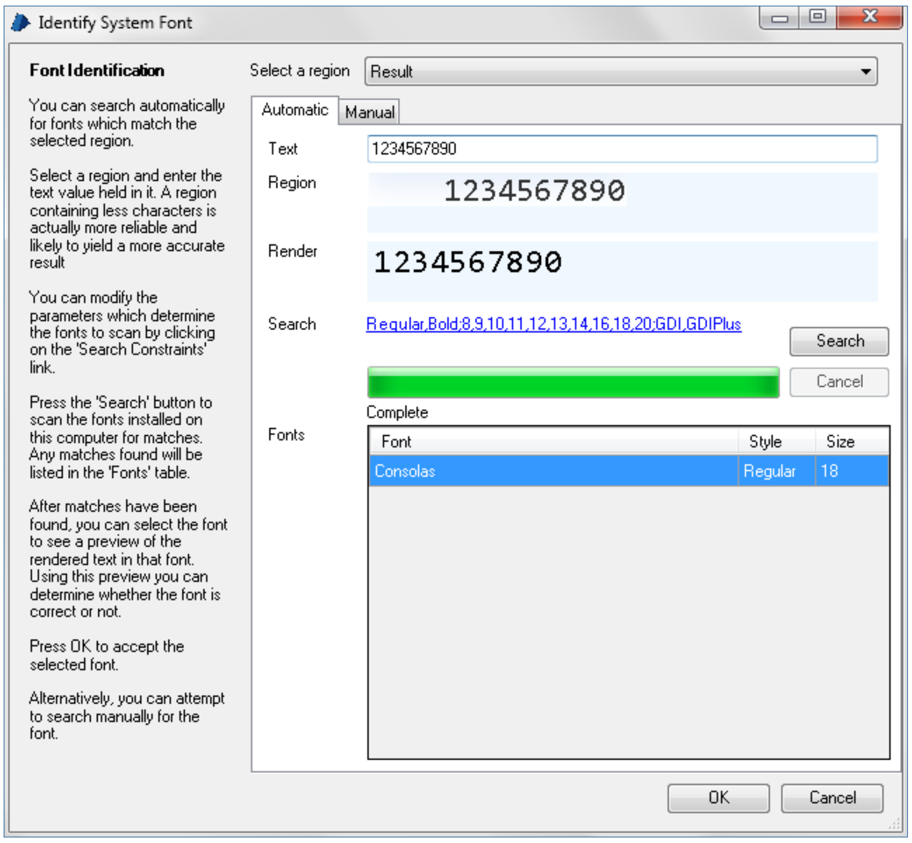
6 7 7.5 [✓ 8] 8.5
[✓ 9] 9.5 [✓ 10] [✓ 11] [✓ 12]
[✓ 13] [✓ 14] 15 [✓ 16] 18
20 22 24 28 32
36

Rendering Methods

[✓ GDI] [✓ GDI+]

Figure 16: Search Parameters Screen

- Increase the scope of the search by selecting the font sizes 18 and 20 (you can speed up the search by deselecting the previously searched sizes).
- Click "OK" to confirm and then search again.
- The "Consolas" font should be identified as a match, as shown below.



Identify System Font

Font Identification

You can search automatically for fonts which match the selected region.

Select a region and enter the text value held in it. A region containing less characters is actually more reliable and likely to yield a more accurate result

You can modify the parameters which determine the fonts to scan by clicking on the 'Search Constraints' link.

Press the 'Search' button to scan the fonts installed on this computer for matches. Any matches found will be listed in the 'Fonts' table.

After matches have been found, you can select the font to see a preview of the rendered text in that font. Using this preview you can determine whether the font is correct or not.

Press OK to accept the selected font.

Alternatively, you can attempt to search manually for the font.

Select a region: Result

Text: 1234567890

Region: 1234567890

Render: 1234567890

Search: Regular,Bold;8,9,10,11,12,13,14,16,18,20;GDI,GDIPlus

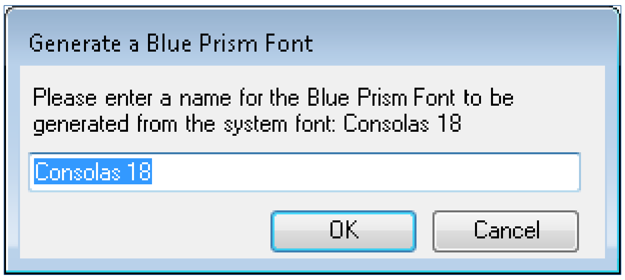
Complete

Fonts:

Font	Style	Size
Consolas	Regular	18

Figure 17: Automatic Font Search Results

- Select the ***Consolas 18*** font and click "OK".
- You can rename the font at this point but in this case, we will simply accept the default name.

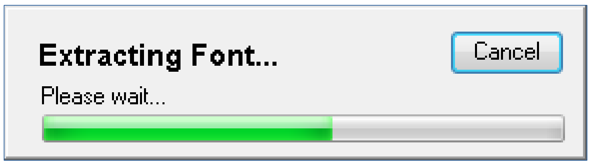


Generate a Blue Prism Font

Please enter a name for the Blue Prism Font to be generated from the system font: Consolas 18

Consolas 18

- Blue Prism will then generate a font definition and store it in the Blue Prism database. This will make the font definition available to other Blue Prism machines connected to the database.



Extracting Font...

Please wait...

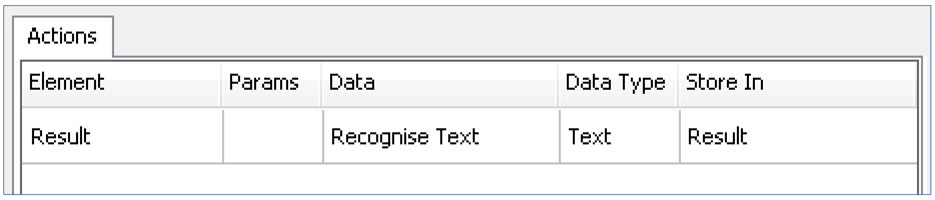
3.3. Read Actions

Now we have identified the Consolas font, we can use a Read action to interpret the text from the Calculator field.

Exercise 3.3.1 Recognize Text

- Close Application Modeler and return to your Calculator Business Object.

- Add a new Read stage and link it in at the end of the page.
- Open the Read stage properties and drag in the ***Results*** region from the Application Model.
- Use the ***Recognize Text*** action to extract text data from the region and store it in a Data Item.

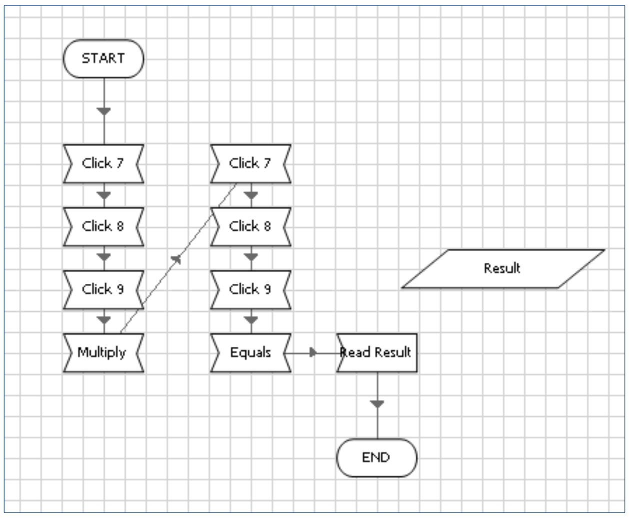


Actions

Element	Params	Data	Data Type	Store In
Result		Recognise Text	Text	Result

Figure 18: Recognize Text

- Your object may now look like the following:



- Run the page and you'll notice that the result is not populated. Blue Prism has failed to recognize the text and has not provided any error message. Why is this?
- The reason the text can't be recognized that Blue Prism doesn't know what color the text is and what color the background is. We need to tell it what to look for.
- To help Blue Prism identify the text, we must provide the font color or the background color as an input to the Read stage. In Windows 7, the Calculator background color is a gray gradient so we will identify the font color instead.

Exercise 3.3.2 Identify Font Color

To use colors as input parameters, they must be provided in hexadecimal format (or hex for short) as a text value, e.g., “FF000000”.

- To determine the font color, take a screenshot of Calculator using the ***PrntScrn*** key and paste the screenshot into Microsoft Paint.
- Zoom in and use the ***Color Picker*** tool click on one of the result numbers.

Figure 19: MS Paint Color Picker Button

- Then open the Edit Colors window by pressing the toolbar button.

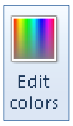
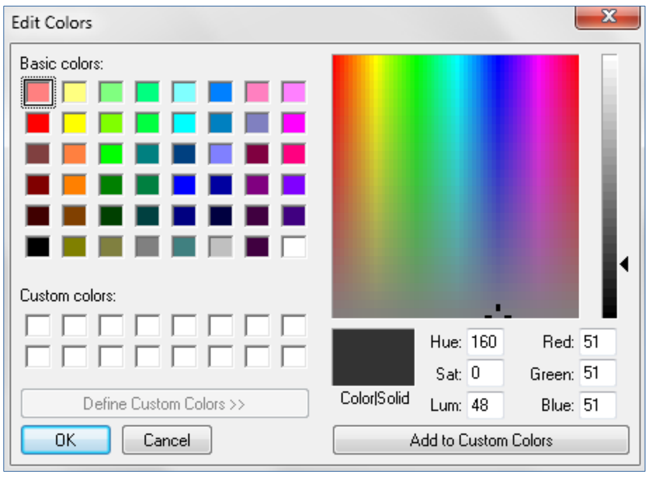


Figure 20: MS Paint Edit Colors Button



Hue: 160
Sat: 0
Lum: 48
Red: 51
Green: 51
Blue: 51

Figure 21: MS Paint Edit Colors Window

- Confirm that the color in the Color|Solid box is the color of the text and not the surrounding area and record the Red, Green and Blue (RGB) number, in this case 51, 51, and 51.
- We know need to convert the number to hexadecimal. This can be done in MS Excel using the following formula =DEC2HEX(number). If your version of Excel doesn’t support this function add the Analysis ToolPak add-in or use the Windows Calculator and set to Scientific in Windows XP or Programmer in Windows 7.
- Each RGB reference needs to be calculated individually. For example, for RGB code 63, 129, 179

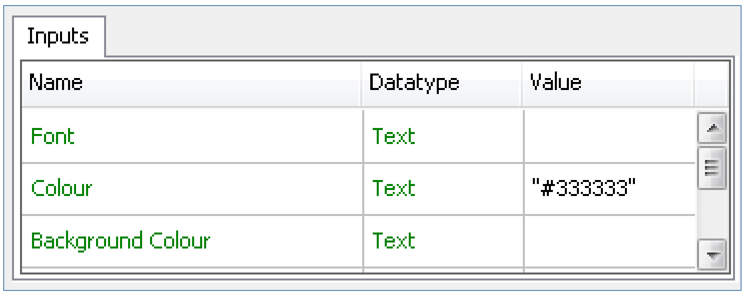
  DEC2HEX(63) is ***3F***

  DEC2HEX(129) is ***81***

  DEC2HEX(179) is ***B3***

- The hex code you would provide to Blue Prism would be ***#3F81B3*** (note the use of the prefix hash symbol ***#***).

- Similarly the hex code for 51,51,51 is ***#333333***
- Go back to your Reader action and set the Color input parameter using the expression "#333333"



Inputs

Name	Datatype	Value
Font	Text	
Colour	Text	"#333333"
Background Colour	Text	

Figure 22: Recognize Text Input Parameters

* *The Font input parameter can be left blank because we have already associated the Results region with the Consolas font in the Region Editor.*

* *The Background input parameter can also be left blank because we only need to provide one color for the Read stage to work out the difference between foreground and background colors.*

- Before running your page, make sure Object Studio is not maximized and that Calculator is visible and not obscured by any other window.
- Run the page and see that the Read stage extracts a text value into your Data Item.

3.4. Recognize Text Input Parameters

As we have seen, the Recognize Text method requires input parameters for Font, Color, and Background Color values. Their usage is described as follows:

Font

- This is the name of the font definition the Read stage should use.
- It can be left blank if the region has a font assigned to it in the Region Editor.
- If the region has no font assigned, the Read stage will use the "system" font of the PC.

Color

- This is the hex code of the foreground color the Read stage should use.
- It must be left blank if a background color has been specified.
- If left blank and no background color is specified, the Read stage will assume the foreground color to be black.

Background Color

- This is the hex code of the background color the Read stage should use.
- It must be left blank if a foreground color has been specified.
- It can also be set to "auto" to indicate that the least prevalent color in the region is the foreground color. This is useful if the color of the text in the region is inconsistent.

4. Send Keys

In controlling the application we have learned how to create regions and use the mouse to click on the region. Typically however a user will need to also send keystrokes to the application.

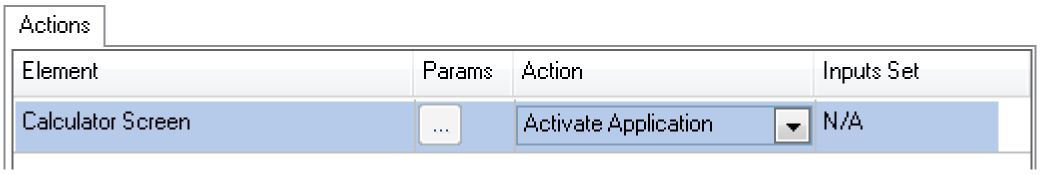
Normally in Blue Prism you would use the Write stage to send data to an application. This is not possible with a thin client application though, (remember it is a virtual application that does not really exist on the local machine) so we must send keystrokes to the application. To do this we use the ***Global SendKeys*** method.

In the previous exercises, using Calculator, we executed our calculation by clicking buttons with the mouse but, just like a human user, we could achieve the same effect by using the keyboard. For us to do this though, we must ensure that the Calculator is in focus and not hidden by any other windows, including Blue Prism itself.

Exercise 4.1.1 Application Focus

In an earlier exercise, we stipulated that Object Studio must not be maximized and that Calculator must be clearly visible. This was because if Calculator was obscured, the mouse clicks and the Recognize Text function might not succeed. We can ensure Calculator is in focus by making the ***Activate Application*** action part of our page.

- Return to your Calculator Business Object and add a new Navigate stage at the start of your page.
- Open the properties form and drag in the Calculator Screen element.
- Select the Activate Application action from the drop-down list and press "OK".

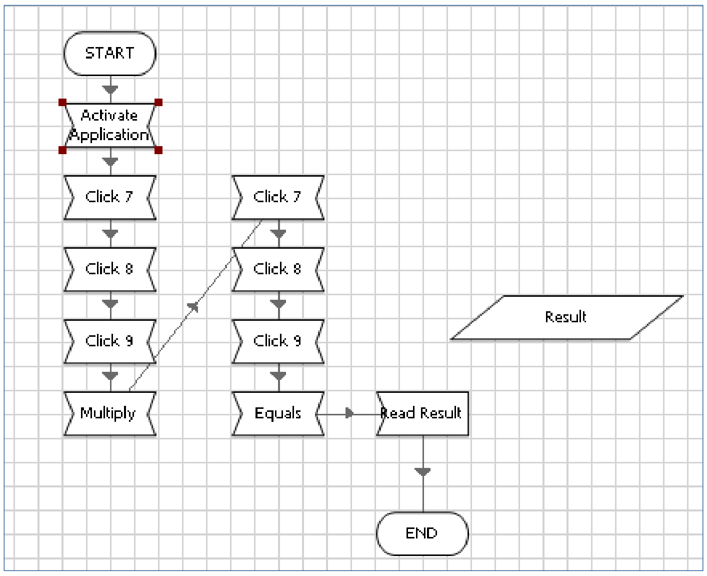


Actions

Element	Params	Action	Inputs Set
Calculator Screen	...	Activate Application	N/A

Figure 23: Activate Application

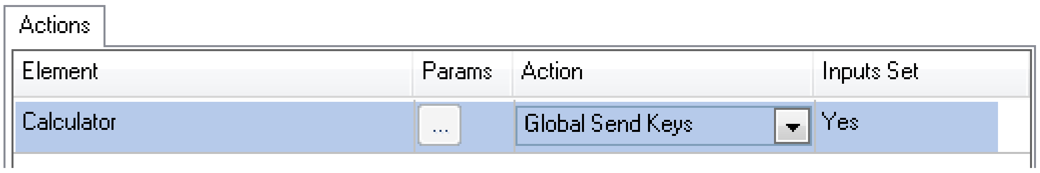
- Your page should look something like this:



Exercise 4.1.2 Global Send Keys

- Create a new page in your Calculator Business Object.

- Copy the Activate stage, the Read stage, and the Data Item from the previous page and paste them on to the new page.
- Add a new Navigate stage under the Activate and open its properties form.
- Drag in the very top (root) node from the Application Model and select the ***Global Send Keys*** action from the list.

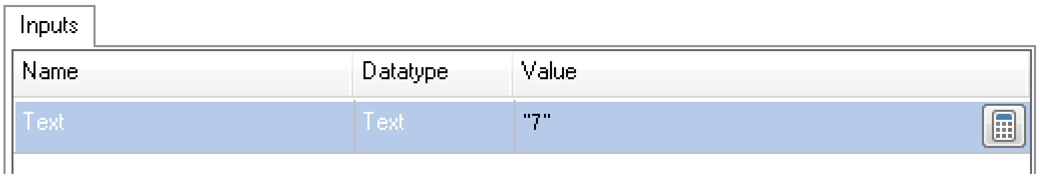


Actions

Element	Params	Action	Inputs Set
Calculator	...	Global Send Keys	Yes

Figure 24: Global Send Keys

- Before closing the properties form, enter the Text input parameter to indicate to the Action stage which keystrokes to send. In this case the expression will be "7"



Inputs

Name	Datatype	Value
Text	Text	"7"

- Continue to add more Navigate stages so that the page sends the keystrokes for 789 * 789 = .You can do this a series of single keystroke actions (i.e., 7 then 8 then 9, etc.) or by sending all the keystrokes from one Navigate stage.
- Run the process. You should see it perform the calculation much like the other page, except that because we are not using the mouse we do not see the mouse pointer move.

5. Image Recognition

Safe automation of the presentation layer can only exist if our process flow is aware of changes to the screen and we are confident that the screen elements we are interfacing with are the correct elements.

In order to interface with the application safely, we need to not only be able to read the text on the screen but also be able to interpret images.

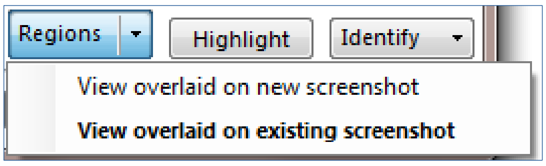
5.1. Waiting for Images

As you will be aware from your training, Wait stages play an important part of a Blue Prism process and enable us to navigate safely through an application. When interfacing with a thin client, we need to wait for images and we use the Region Editor to record element snapshots.

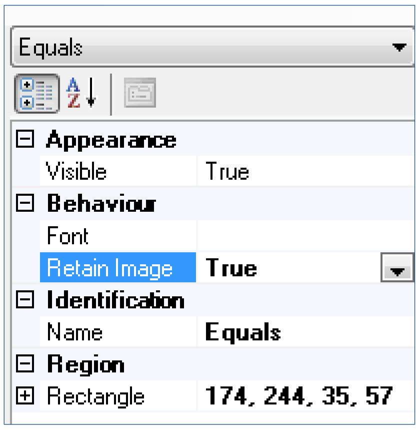
Exercise 5.1.1 Retain Image

Application Modeler is able to keep a screenshot of a region as part of the make-up of a region element. This setting is switched off by default but can easily be switch on if necessary.

- Open the Region Editor using the existing screenshot.


Regions
Highlight
Identify
View overlaid on new screenshot
View overlaid on existing screenshot

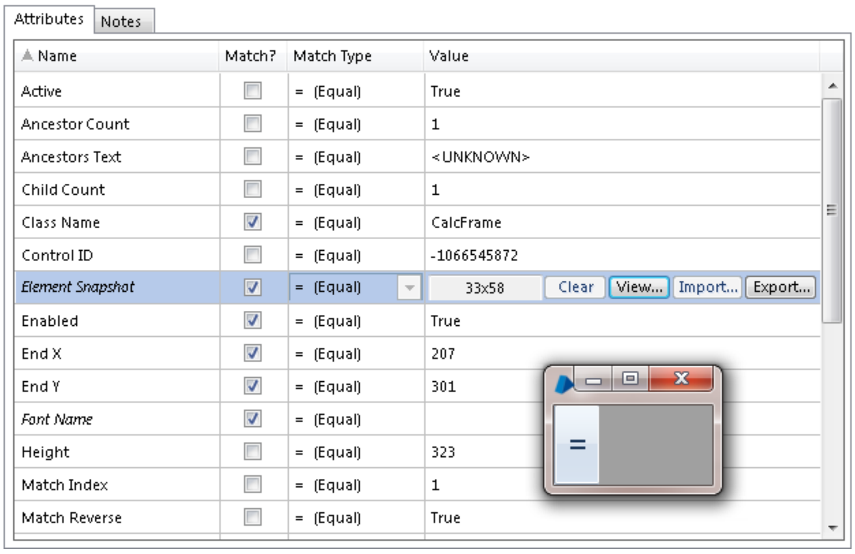
- Select the ***Equals*** region from the drop-down menu and change the ***Retain Image*** property to “True”.



Equals

Appearance	
Visible	True
Behaviour	
Font	
Retain Image	**True**
Identification	
Name	**Equals**
Region	
Rectangle	**174, 244, 35, 57**

- Close the Region Editor to return to the Application Modeler.
- Inspect the attributes of the Equals region and you’ll see an attribute named ***Element snapshot***. This is where a region image is saved.
- Click the View button to see the image.



Attributes | Notes

Name	Match?	Match Type	Value
Active	☐	= (Equal)	True
Ancestor Count	☐	= (Equal)	1
Ancestors Text	☐	= (Equal)	<UNKNOWN>
Child Count	☐	= (Equal)	1
Class Name	✓	= (Equal)	CalcFrame
Control ID	☐	= (Equal)	-1066545872
Element Snapshot	✓	= (Equal)	33x58 Clear View... Import... Export...
Enabled	✓	= (Equal)	True
End X	✓	= (Equal)	207
End Y	✓	= (Equal)	301
Font Name	✓	= (Equal)	
Height	☐	= (Equal)	323
Match Index	☐	= (Equal)	1
Match Reverse	☐	= (Equal)	True

Figure 25: Element Snapshot Attribute

Now we have made the snapshot part of the region, we can use Wait stage to effectively say, "Wait until this region of the application exists and ***looks exactly like this***."

Exercise 5.1.2 Launch and Wait for Image

- Create a new page in our Calculator object named ***Launch Calculator***
- Have the page launch the application and wait for the ***Equals*** region to appear. You will need to use the ***Contains Image*** condition within your Wait stage.

Waiting for images is an important function when automating thin client applications. It is important that you confirm each element before interacting with it - for example, confirming the existence of a label before populating the field next to it.

Exercise 5.1.3 Relative Position

- Reopen the Application Modeler of the Calculator object and ensure that the Calculator application has been launched.
- Select the ***Equals*** element and then click the Highlight Element button.
- Now manually set the Calculator to Scientific mode using the View menu

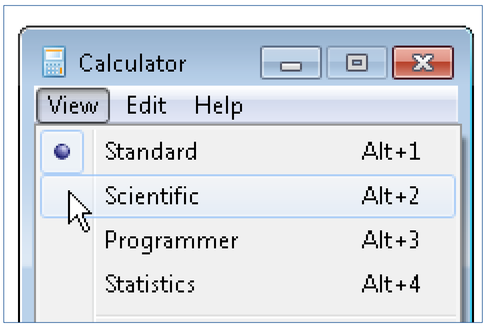
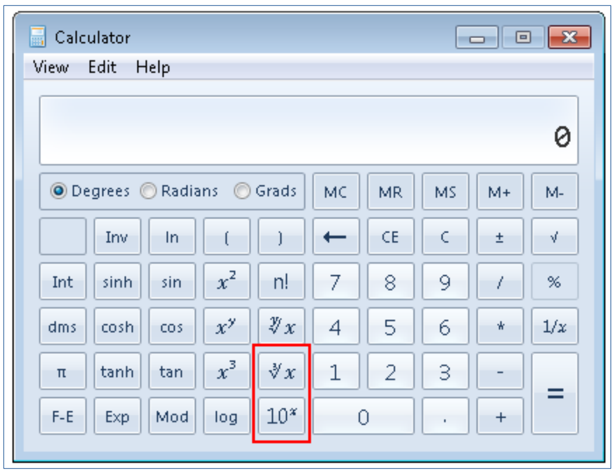


Figure 26: Calculator Scientific Mode

- Click the Highlight Element button again and notice that the region is no longer over the Equals button.



The highlighted area no longer sits over the Equals button. This is because Blue Prism is highlighting a region relative to the original parent element, in this case, the main window of Calculator.

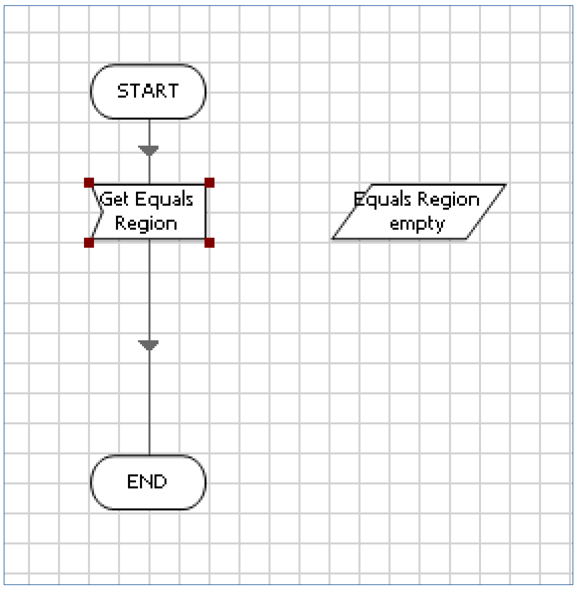
Unlike thick client elements, the region element is essentially a rectangular area, ***relative to the top-left*** corner of its parent. To interface with thin client applications, Blue Prism Best Practice recommends qualifying the position of all elements prior to interaction, and we can do this by reading images.

Exercise 5.1.4 Reading an Image

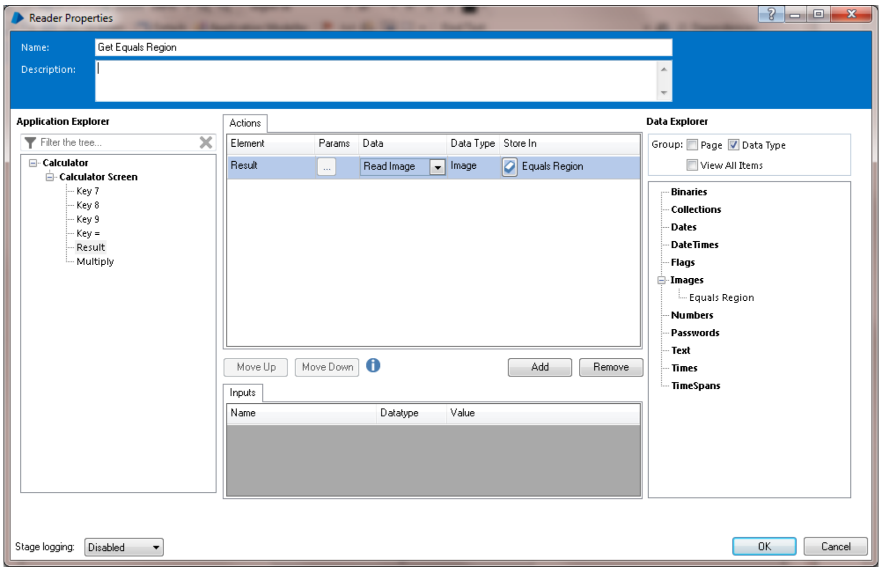
- Reset the Calculator to Standard mode and create a new page in your Business Object.
- Add a Read stage named ***Get Equals Region***
- Use the Read Image action to capture a screenshot of the region in a Data Item.

  * *Tip: Use the Image data type for your image.*

- Your action should look something like this:



- Your Read stage should look something like this:



Reader Properties

Name: Get Equals Region

Description:

Application Explorer

- Calculator
  - Calculator Screen
    - Key 7
    - Key 8
    - Key 9
    - Key =
    - Result
    - Multiply

Actions

Element	Params	Data	Data Type	Store In
Result	...	Read Image	Image	Equals Region

Inputs

Name	Datatype	Value

Data Explorer

Group: Page [✓ Data Type]

View All Items

- Binaries
- Collections
- Dates
- DateTimes
- Flags
- Images
  - Equals Region
- Numbers
- Passwords
- Text
- Times
- TimeSpans

Stage logging: Disabled

- Launch the Calculator and step through the action to populate the Data Item.

Exercise 5.1.5 Image Data Item Current Value

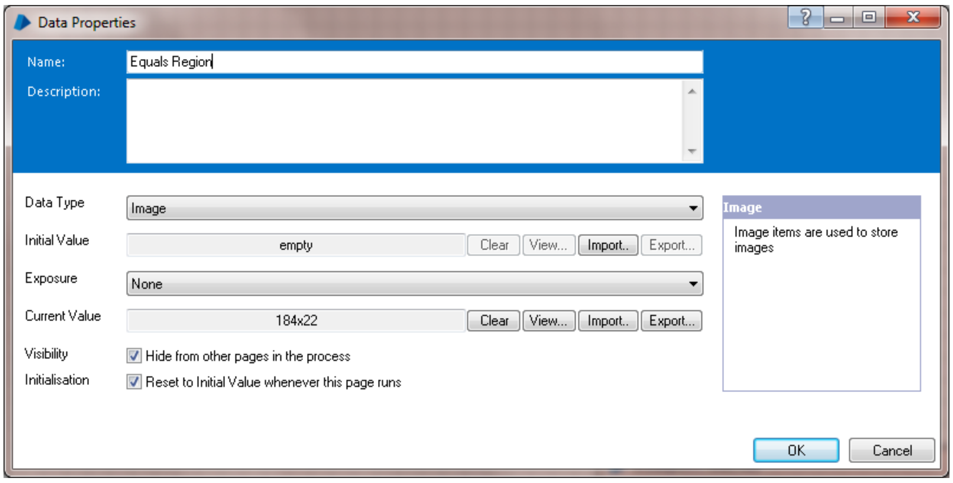
We now have an image of the Equals region.

- Access the properties of the Data Item to view the image.
- Set the Calculator to Scientific mode and step though the process again.
- See that now we have captured a different image - the images we capture are ***runtime*** images.

Exercise 5.1.6 Image Data Item Initial Value

In order to track an image, we first need to create a Data Item containing the image we need to track. This will act as a reference for us to check the runtime image against.

- Reset the Calculator to Standard mode and capture the Equals region again (i.e., step through the page again).
- Open the Data Item properties and view the image to confirm it is correct.



Data Properties

Name: Equals Region

Description:

Data Type: Image

Initial Value: empty

Exposure: None

Current Value: 184x22

Visibility: ✓ Hide from other pages in the process

Initialisation: ✓ Reset to Initial Value whenever this page runs

Image

Image items are used to store images

Figure 27: Data Item Properties

- Staying in the Data Item properties form, export the current value of the Data Item as a .PNG file.

    * *Tip: The exported file is only temporary and we're not going to need to keep it.*

- Close the Data Item properties form.
- Create a new Data Item named ***Equals Button*** and set the data type to ***Image***.
- Set the Initial Value of the Data Item by importing the saved .PNG file.
- We will now use this Data Item in the next section.

5.2. Tracking Elements around the Screen

Although we can wait for a region to contain the snapshot we retained from the Region Editor, this will only prove successful if the image is in the same relative position to the parent element. If the application has changed, or we have had to scroll to find the correct part of the application, the correct element may be in a different position.

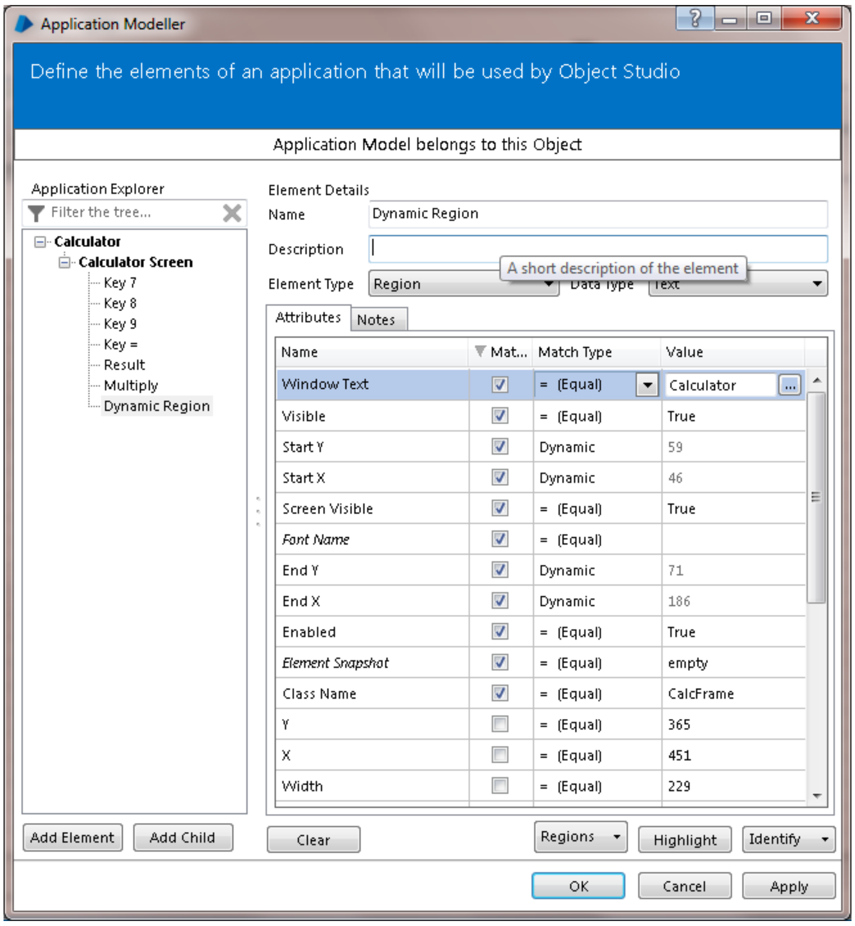
To find the element, we must search for it on the screen.

- Amend the page from the previous exercise so it only contains the ***Equals Button*** Data Item.
- We want to capture a screenshot of the application during run time. As we won't know how big the application will be (standard mode v scientific mode), we will get these details from the application before capturing a screenshot.
- First open Application Modeler and then Region Editor.
- Create a new region anywhere on the screen and call it ***Dynamic Region***.

    * *Tip: It doesn't matter where you put the region because we will move it around the screen during runtime.*

- Close Region Editor and select the element in Application Modeler. Change the Start ***X, Start Y, End X****,* and ***End Y*** attributes to dynamic as shown below.

    * *Tip: You can sort the matched attributes by selecting the match column.*



Application Modeller

Define the elements of an application that will be used by Object Studio

Application Model belongs to this Object

Application Explorer

- Calculator
  - Calculator Screen
    - Key 7
    - Key 8
    - Key 9
    - Key =
    - Result
    - Multiply
    - Dynamic Region

Element Details

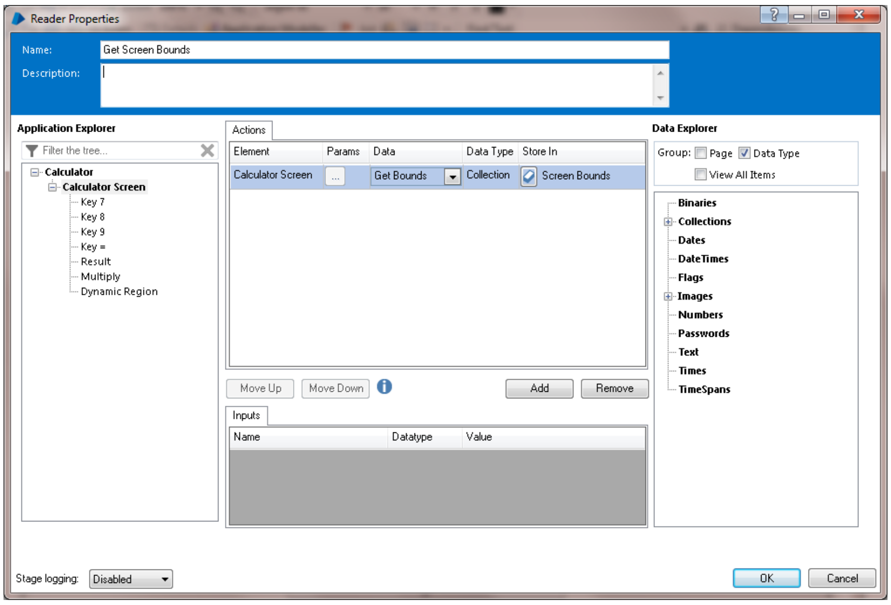
Name	Dynamic Region
Description	
Element Type	Region
Data Type	Text

Attributes | Notes

Name	Mat...	Match Type	Value
Window Text	✓	= (Equal)	Calculator
Visible	✓	= (Equal)	True
Start Y	✓	Dynamic	59
Start X	✓	Dynamic	46
Screen Visible	✓	= (Equal)	True
Font Name	✓	= (Equal)	
End Y	✓	Dynamic	71
End X	✓	Dynamic	186
Enabled	✓	= (Equal)	True
Element Snapshot	✓	= (Equal)	empty
Class Name	✓	= (Equal)	CalcFrame
Y		= (Equal)	365
X		= (Equal)	451
Width		= (Equal)	229

Figure 28: Dynamic Attributes

- Close Application Modeler.
- Add a Read stage to the page named ***Get Screen Bounds*** and a Collection named ***Screen Bounds***
- Tip: Don’t create any fields in the collection.
- In your Read stage, use the “Get Bounds” action to return the bounds of the ***Calculator Screen*** element to the Get Screen Bounds collection.
- Your Read stage properties should look something like this:


Reader Properties

Name: Get Screen Bounds

Description:

Application Explorer

- Calculator
  - Calculator Screen
    - Key 7
    - Key 8
    - Key 9
    - Key =
    - Result
    - Multiply
    - Dynamic Region

Actions

Element	Params	Data	Data Type	Store In
Calculator Screen	...	Get Bounds	Collection	Screen Bounds

Inputs

Name	Datatype	Value

Data Explorer

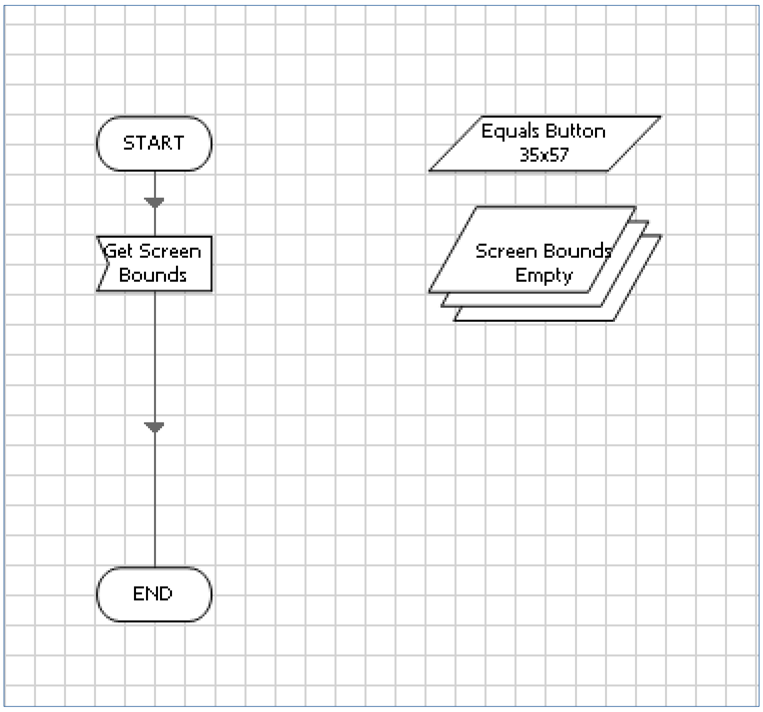
Group: ☐ Page ☑ Data Type ☐ View All Items

- Binaries
- Collections
- Dates
- DateTimes
- Flags
- Images
- Numbers
- Passwords
- Text
- Times
- TimeSpans

Stage logging: Disabled

Figure 29: Reader Properties Form

- Your page should look something like this:

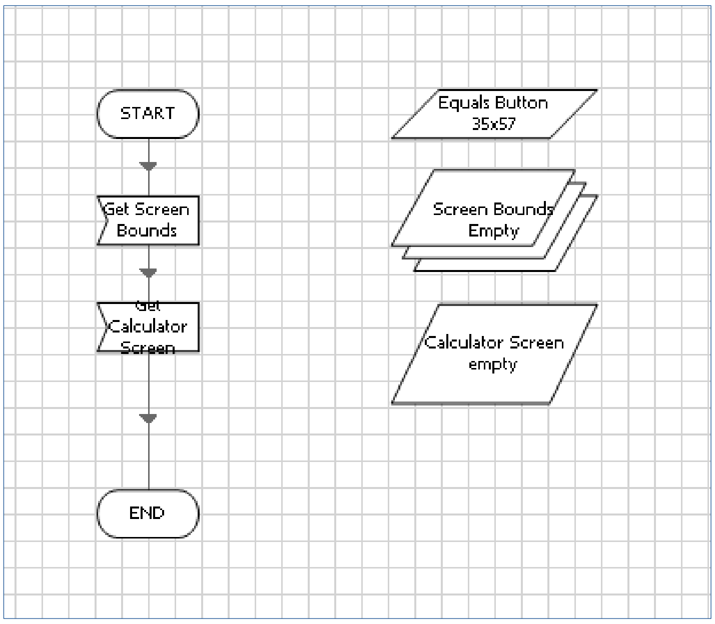


- Test your page and inspect the results in the Collection - the Collection captures the position and size of the Calculator Screen element.

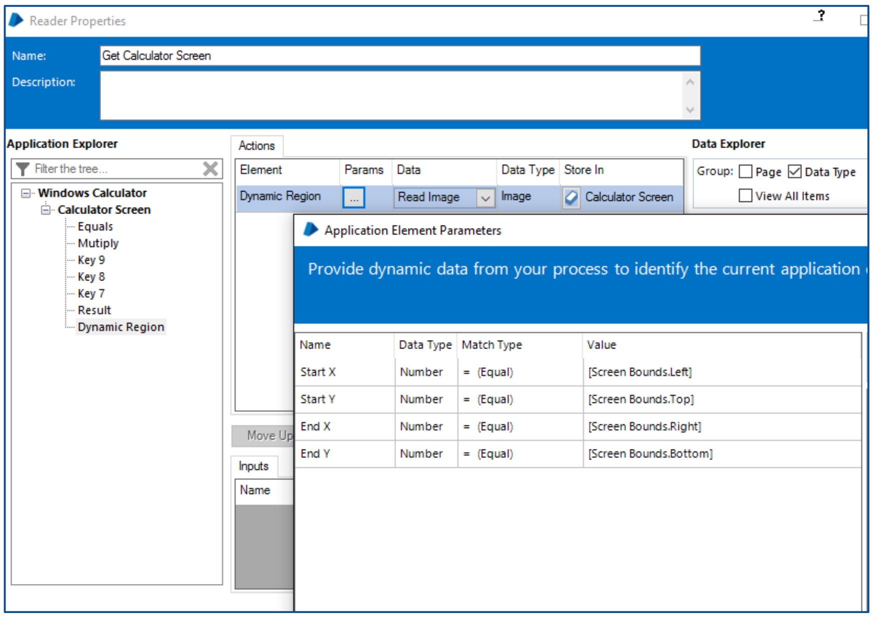
Now we want to take a snapshot image of the screen using the newly acquired bounds.

- Add a Read stage to your page to capture the image using the dynamic attributes of the Dynamic Region.
- Call the Read stage ***Get Calculator Screen*** and the Data Item ***Calculator Screen***

- Your page should now look something like this:



- Your Read stage properties should look something like this:



Name: Get Calculator Screen

Description:

Element	Params	Data	Data Type	Store In
Dynamic Region	...	Read Image	Image	Calculator Screen

Application Element Parameters

Provide dynamic data from your process to identify the current application

Name	Data Type	Match Type	Value
Start X	Number	= (Equal)	[Screen Bounds.Left]
Start Y	Number	= (Equal)	[Screen Bounds.Top]
End X	Number	= (Equal)	[Screen Bounds.Right]
End Y	Number	= (Equal)	[Screen Bounds.Bottom]

- Step though the process and inspect the properties of the Calculator Screen image Data Item. It should have captured the image of the Calculator screen.
- Set the Calculator to Scientific and step though the action again.

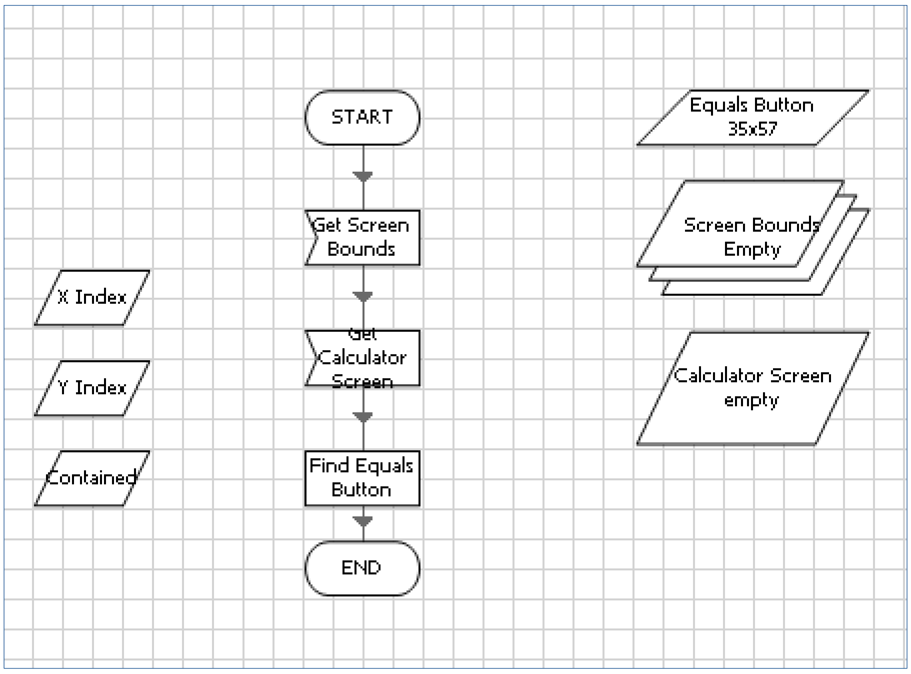
We now have a Data Item containing the Equals button reference image and a Data Item containing an image of the whole Calculator screen. This means we are able to search for Equals button image and find its position within the Calculator screen image.

Once we have those coordinates, we can use them as attributes of the Dynamic Region element, thus enabling us to send a mouse click to the correct position.

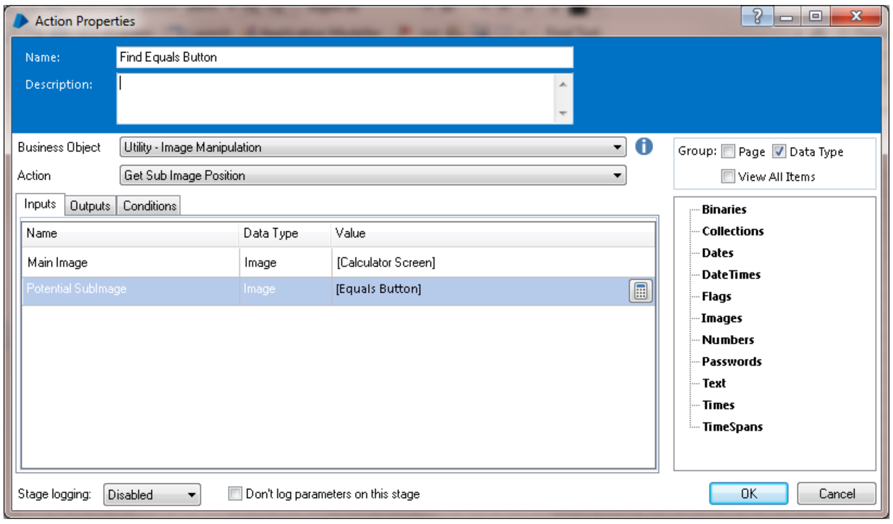
Exercise 5.2.1 Image Utility Business Object

To search for an image within an image, we need to use the Utility Business Object named ***Utility - Image Manipulation***.

- If you haven't already done so, import the object from the file C:\Program Files\Blue Prism Limited\Blue Prism Automate\VBO\ BPA Object - Utility - Image Manipulation.xml
- Refresh Object Studio by pressing the Refresh button
- Add a new Action stage named "Find Equals Button"
- Select the "Get Sub Image Position" action from the Utility - Image Manipulation object to return the X and Y coordinates of the Equals button image.
- Your page should now look something like this:



- Your Action stage properties should look something like this:



Name: Find Equals Button

Description:

Business Object: Utility - Image Manipulation

Action: Get Sub Image Position

Group: [ ] Page [✓ Data Type]

[ ] View All Items

Inputs | Outputs | Conditions

Name	Data Type	Value
Main Image	Image	[Calculator Screen]
Potential SubImage	Image	[Equals Button]

Stage logging: Disabled

[ ] Don't log parameters on this stage

- Step though the page and observe the X and Y values returned.
- Change the calculator to scientific mode and step though the page again.

Exercise 5.2.2 Dynamic Region

To be able to click the Equals button no matter where it is on the screen, we will need to use a region with its coordinates exposed as ***dynamic attributes***.

Recall from the X and Y values returned when you set the Calculator to normal and scientific mode that the Y value doesn't change. This is because the Equals button only moves horizontally within the container element. Hence we only need to make the Start X and End X coordinates dynamic.

Remember why we need the coordinates of the Equals button. We want to be able to perform the calculation we created in 1.2 even if the Calculator button moved around the screen.

So, using what we have just learned:

- Capture images for the ***7, 8, 9****,* and ***** buttons.
- Find the positions of these buttons.
- Use the ***Dynamic Region*** element to click on the buttons and perform the 789 * 789 calculation with Calculator in both Normal and Scientific mode.
- Notice the time it takes to find the buttons. We will look at performance optimization later, but can you think of any way of improving the performance? Hint: the bigger the image you search in, the longer it takes.

6. Summary

During this introduction to Surface Automation we have learned:

- How to identify our application elements using regions.
- How to track elements around the screen.
- How to control an application safely using mouse clicks and keystrokes.
- How to identify fonts and read text from the screen.

The next stage in Blue Prism Surface Automation training is the Surface Automation Training Course where you will learn:

- How to interact with particular element types.
- How to create a font when you can’t match to a system font.
- How to address character conflicts.
- How to improve performance.