



Software Robots - the Virtual Workforce

Surface Automation

BASIC TRAINING GUIDE

Version: 6.0.0

For more information please contact:

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

www.blueprism.com

Contents

Introduction	4
1 Regions.....	5
1.1 Image Location Method	6
Exercise 1.1.1 Creating an image region	6
Exercise 1.1.2 Search Padding.....	8
Exercise 1.1.3 Maximum Search Padding.....	10
Exercise 1.1.4 Mouse clicks.....	10
1.2 Relative Position.....	12
Exercise 1.2.1 Related regions	12
Exercise 1.2.2 Multi-level dependencies.....	15
Exercise 1.2.3 Relative position with padding.....	16
Exercise 1.2.4 Relative position with maximum padding	16
1.3 Coordinate Location Method	18
Exercise 1.3.1 Creating a basic coordinate region	18
Exercise 1.3.2 Obscuring a coordinate region.....	20
Exercise 1.3.3 False positives with a coordinate region	21
2 Data Input	22
2.1 Send Keys	22
Exercise 2.1.1 SA Training Business Object	22
Exercise 2.1.2 Application focus.....	23
Exercise 2.1.3 Global Send Keys	25
Exercise 2.1.4 Field focus	26
Exercise 2.1.5 Paste from clipboard.....	28
2.2 Mouse Click.....	29
Exercise 2.2.1 Login Action	29
3 Data Output	30
3.1 Character Matching	30
Identifying Fonts	30
Exercise 3.1.1 Region font.....	30
Exercise 3.1.2 Identifying a system font (Part 1)	32
Exercise 3.1.3 Recognise Text	33
Exercise 3.1.4 Colours and Recognise Text	35

Exercise 3.1.5	Hex codes.....	36
Exercise 3.1.6	Fonts and Recognise Text.....	36
	Recognise Text Input Parameters	39
3.2	Read Text with OCR.....	39
Exercise 3.2.1	Read Text With OCR.....	40
4	Synchronisation.....	41
Exercise 4.1.1	Waiting for an image region.....	41
Exercise 4.1.2	Waiting and Search Padding	42
Exercise 4.1.3	Waiting and colour tolerance.....	43
Exercise 4.1.4	Coordinate regions and Retain Image.....	43
5	Summary.....	45
6	Appendix.....	46
	Font Smoothing.....	46
	Further Reading	46

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 - 2017

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

Introduction

This course is intended for Blue Prism users who have successfully completed the Blue Prism **Foundation Training** course. The topics in this course are quite advanced and are not suitable for novices, so if you haven't yet done the Foundation course, stop reading now.

This course will cover the basic principles of an integration method called **Surface Automation**. Surface Automation is a technique for working with images and although primarily used on **thin-client** applications, it can also be applied to any application with a user interface.

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 some virtualisation software, such as Citrix.

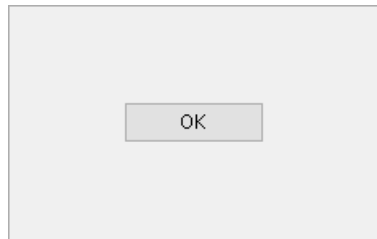
With a thin client, Blue Prism cannot use its normal integration techniques because the target application is virtual and there is very little that spy modes like Win32 and AA are able to detect. Often it is only the main window of the virtualisation software that can be spied conventionally. An analogy could be to think of the surface of a thick client application as having a 'landscape' made of buttons, fields, and checkboxes, whereas the surface of a thin client application is smooth and featureless.

To integrate with a thin client application, we must redefine the application surface as a series of **regions**.

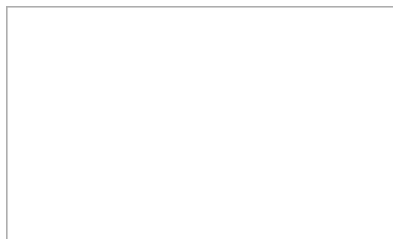
1 Regions

In Blue Prism a region is a rectangular area 'geographically' bound to a parent Win32 element. Blue Prism finds the parent element first and then focuses in on the child region. In version 5, regions were essentially a set of coordinates but with the advent of version 6, new functionality has been added to make Surface Automation a much more powerful integration technique.

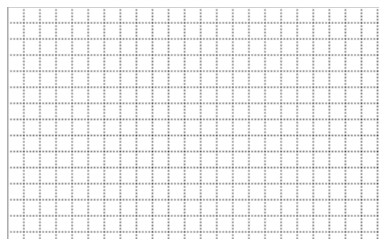
We will begin by discussing the concept of a region. Below is an imaginary thin client application consisting of a plain, borderless window with a button in the centre.



If we were to spy this application as a thick client, Win32 and AA mode would 'see' the application effectively like this.



With 'normal' spy modes, Blue Prism can only detect features of the local virtualisation application, such as its size and position, but it is blind to the virtualised elements within. The only local information we have on the remote application, is the way it looks, i.e. the pixels used to display it to us.



In Surface Automation we create regions in order to divide the image into meaningful sections, to build up an application model, and to design sequences of interaction. By capturing screenshots from the regions at 'run time', we can extract information about the images, compare them against the 'design time' model and make logical deductions.

In our imaginary thin client, we could model a region of the button text to enable us to check at run time that the button is present, where it is located, and to direct a mouse click on to it.

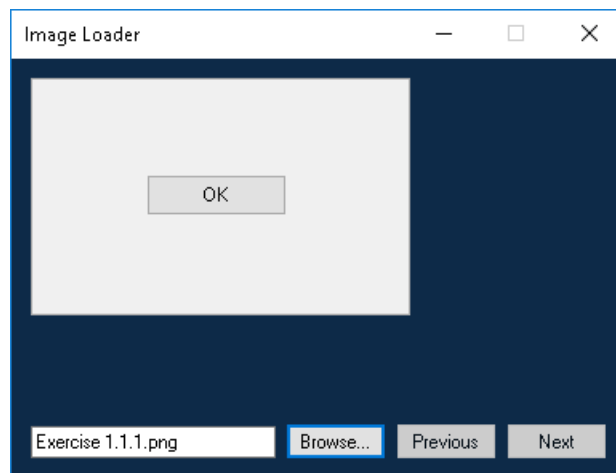


1.1 Image Location Method

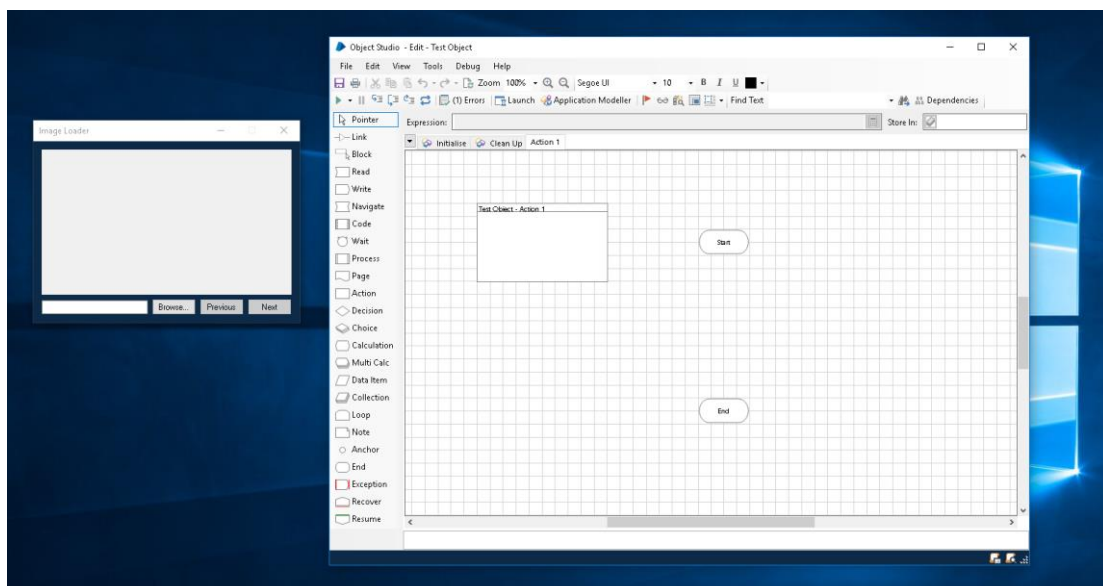
Exercise 1.1.1 Creating an image region

For most of the exercises in this document we will employ a simple training application to present different types of image, simulating the effect of a thin client.

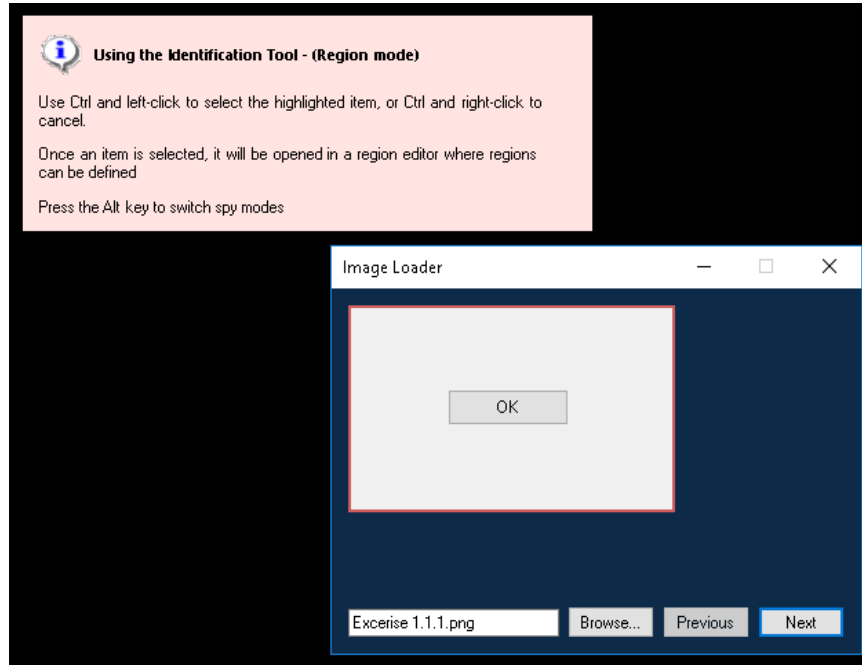
- Launch **Image loader.exe** and open the **Exercise 1.1.1.png** image.



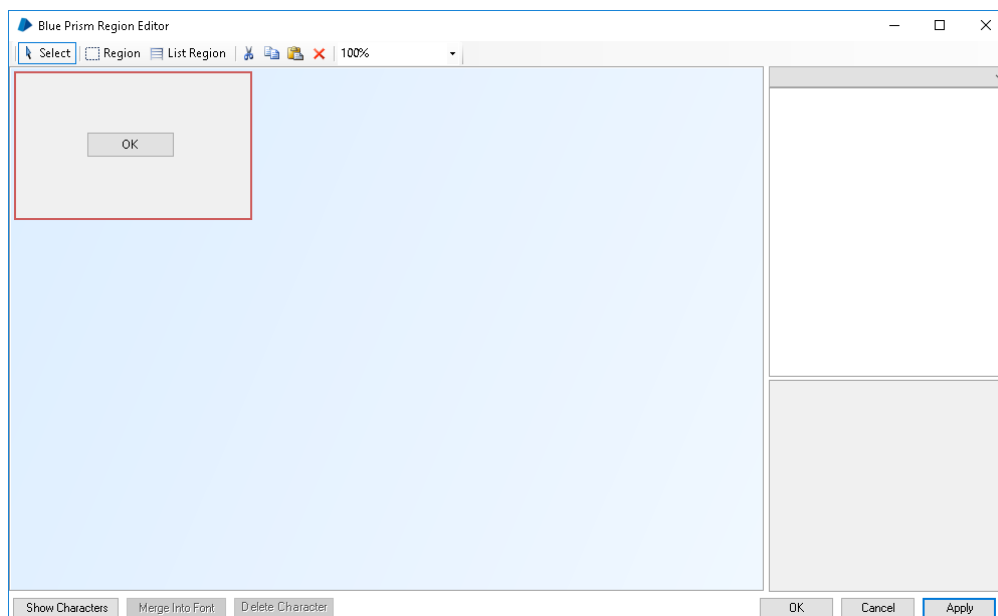
- Create a Business Object that attaches to Image Loader.
- Resize Object Studio to cover about two thirds of the desktop and move Image Loader to the other side, something like this.



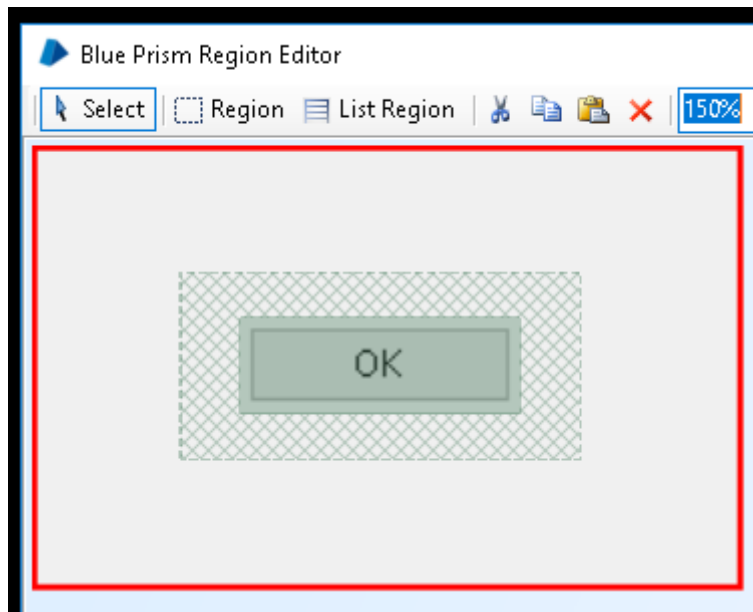
- Open Application Modeller and select Element1 in the tree view on the left. Press Identify to begin spying.
- The default spy mode is Win32, but we don't want to use that here. Press ALT **twice** to engage Region mode.



- Select the Picture Box element in the usual way with a CTRL and left click. This will open **Region Editor**.



- In Region Editor you may wish to use the drop-down list on the toolbar to zoom in and make things easier to see.
- To create a region, select the Region button on the tool bar and draw a box over the button, like below. We will discuss the border of cross hatching later.



- Press OK to close Region Editor.
- In the application model you will see that under 'Element1' a 'Region 1' node has appeared. Select this node and press Highlight. The region should highlight much like a normal Win32 element.

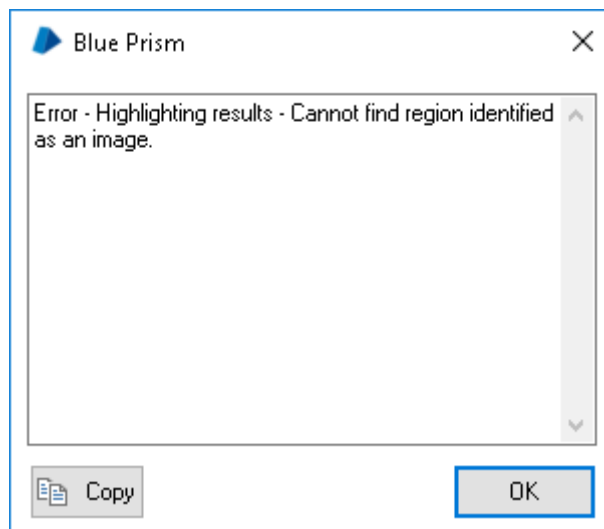
Exercise 1.1.2 Search Padding

You will probably have been wondering about the cross-hatching surrounding your region, so let's now see what it's for.

- Load the [Exercise 1.1.2a.png](#) file and notice that the OK button is now in a slightly different position.
- In Application Modeller, press Highlight again. The region was still found. Why?
- Return to Region Editor by pressing the Regions button.

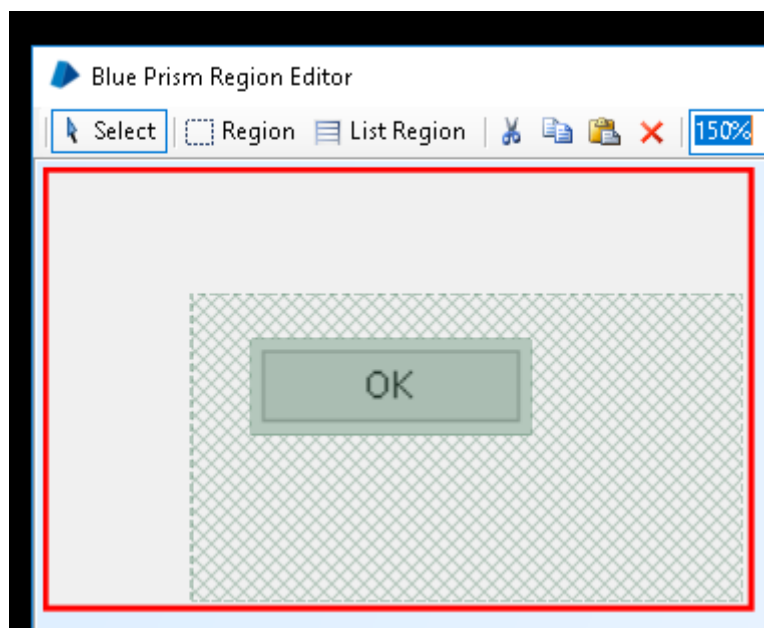


- The hatched 'padding' defines the area that Application Modeller will search in when trying to locate the region image.
- Close Region Editor and load the [Exercise 1.1.2b.png](#) image. The button is now in a markedly different location.
- Press Highlight. You should find that the image cannot be found because the button is outside the padding.



- Go back to Region Editor and change the Right and Bottom values of the Search Padding property. Set the Right values to about 70 and Bottom to about 55 so that the cross hatching extends towards the edge of the image.

Region Location	
Position	Fixed
Location Method	Image
Search Padding	20, 15, 70, 55
All	-1
Left	20
Top	15
Right	70
Bottom	55
Colour Tolerance	0
Greyscale	False

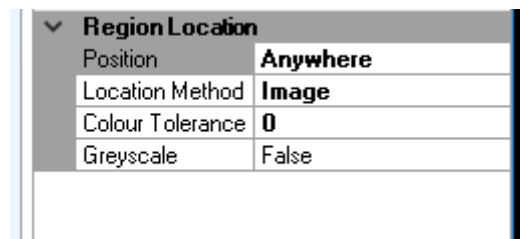


- Press OK to come out of Region Editor, select the region element in the left-hand tree, and press Highlight. You should find that the region is now found.
- Press the Previous button on Image loader to go back to the other button images to confirm that they too can be highlighted.

Exercise 1.1.3 Maximum Search Padding

The search area for an image region can also be maximised to the bounds of its parent Win32 element by a simple change of properties.

- Go back to Region editor and change Position property to Anywhere. We'll discuss the Relative option later.

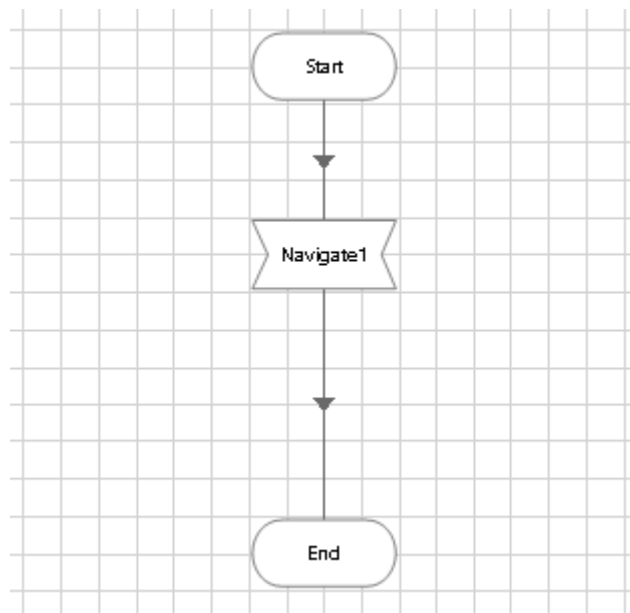


- Observe that the padding is expanded to cover the whole of the parent element.

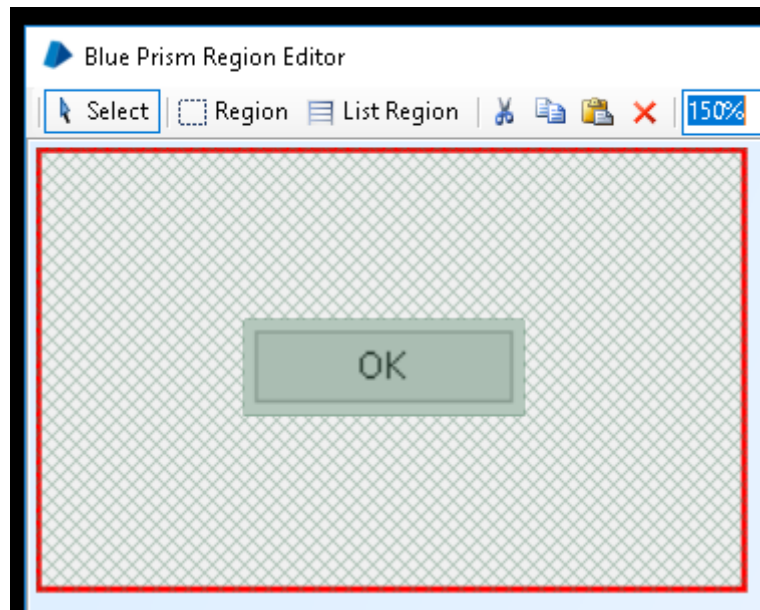
Exercise 1.1.4 Mouse clicks

Now that we have our first region, let's see how to start using it.

- Press OK to close Application Modeller.
- Add a Navigate stage to a new page and link it between the Start and End stages.



- Open the Navigate properties and set up a **Global Mouse Click Centre** action.



- Close Region Editor and confirm that Highlight will still find the button in the three images.

Key Points

- At 'run time' Blue Prism will naively assume that an image region is in its original 'design time' location.
- If it is not, then Blue Prism will look in the Search Padding area until it locates the image's current position. The search will spiral out from the 'design time' location towards the edge of the padding.
- If the image can't be found an exception will be raised, much in the same way that a 'No element found' error is thrown for other types of element.
- In general, it is recommended to keep the default padding values unless you know the element location is dynamic. Reducing the padding to zero for a static element won't improve efficiency because the search begins in the 'design time' location.

1.2 Relative Position

It's not uncommon to find the same image repeated on an application window, as in the example below where there are duplicate labels and buttons. This presents a risk of using the wrong field or pressing the wrong button.

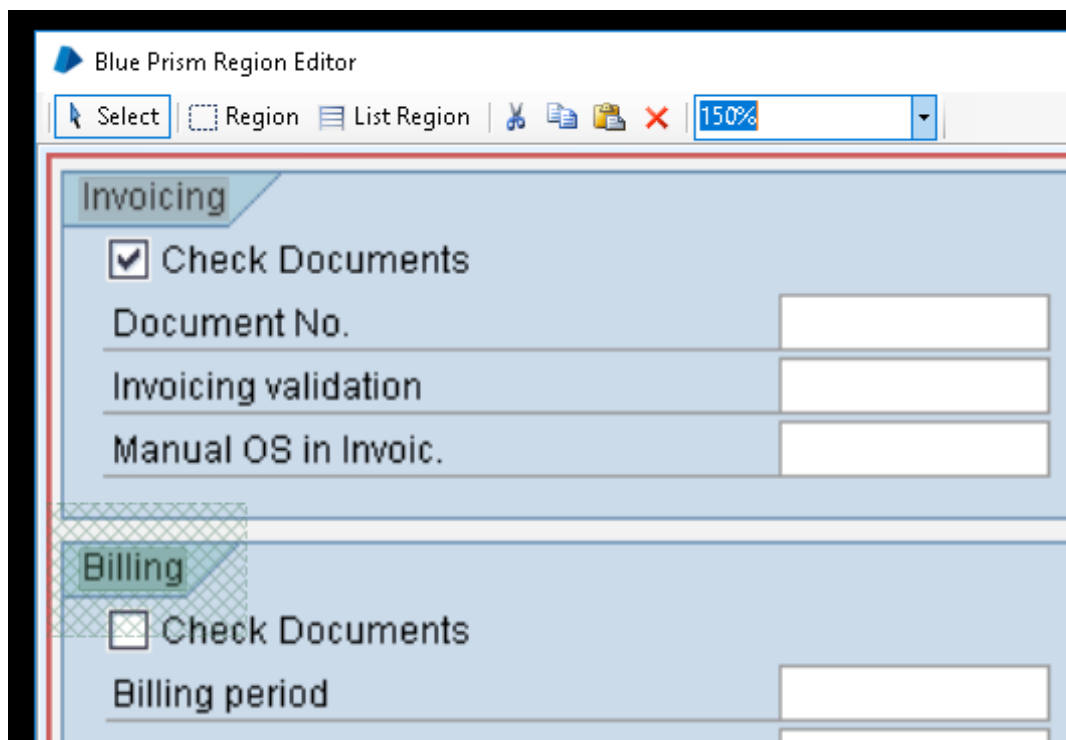
The way to handle this in Surface Automation is to make **relationships** between regions. If you were asked to 'enter the billing document number range and press the button' you would instinctively find the 'Billing' area and the fields and button to the right of the 'Document No.' label. Surface Automation takes a similar approach.

Invoicing	
<input checked="" type="checkbox"/> Check Documents	
Document No.	to
Invoicing validation	to
Manual OS in Invoic.	to

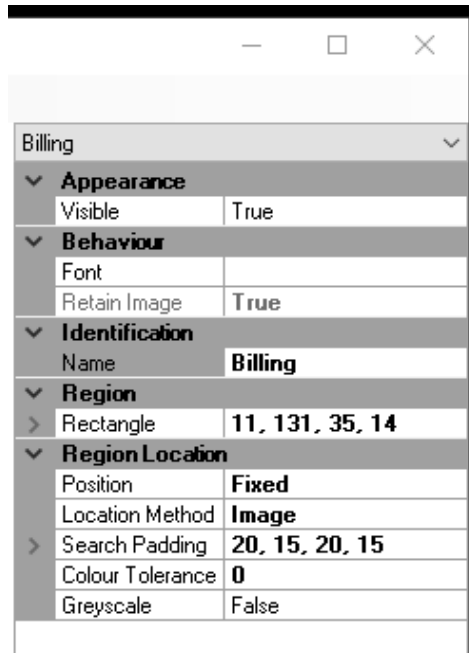
Billing	
<input type="checkbox"/> Check Documents	
Document No.	to
Contract	to
No. of billing run	to
Billing Validation	to
Manual OS in Billing	to
Company Code	to
Division	to
Meter reading unit	to

Exercise 1.2.1 Related regions

- Open the [Exercise 1.2.1.png](#) in Image Loader.
- Create 2 regions, one around the Invoicing section header and a similar one for Billing.



- Keep the default properties but change the names of the regions to Invoicing and Billing.



Billing	
Appearance	
Visible	True
Behaviour	
Font	
Retain Image	True
Identification	
Name	Billing
Region	
Rectangle	11, 131, 35, 14
Region Location	
Position	Fixed
Location Method	Image
Search Padding	20, 15, 20, 15
Colour Tolerance	0
Greyscale	False

- Confirm that both regions can be highlighted as normal.

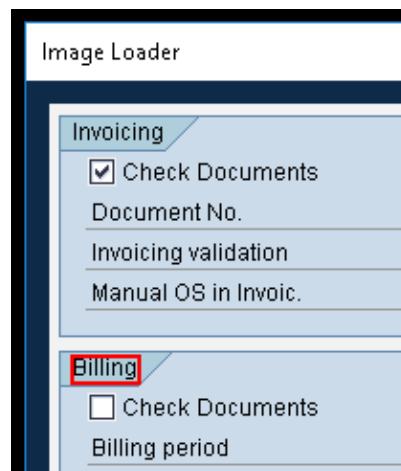


Image Loader

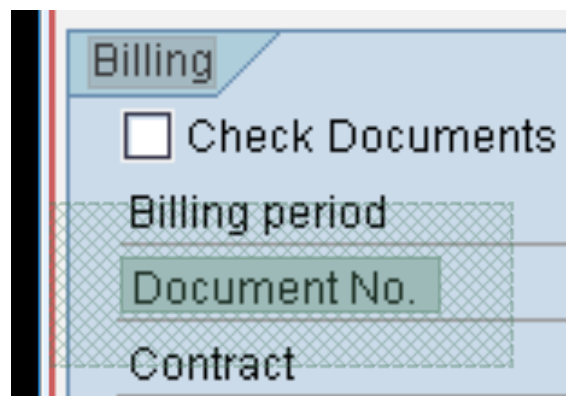
Invoicing

- ☒ Check Documents
- Document No.
- Invoicing validation
- Manual OS in Invoic.

Billing

- ☐ Check Documents
- Billing period

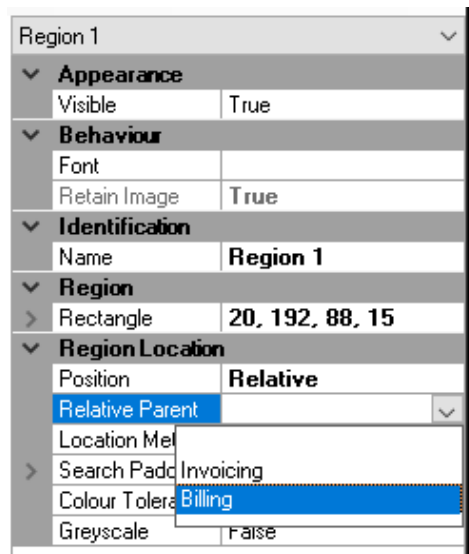
- Return to Region Editor and create new region around the Document No. label in the Billing section. Rename this region as 'Document Number'.



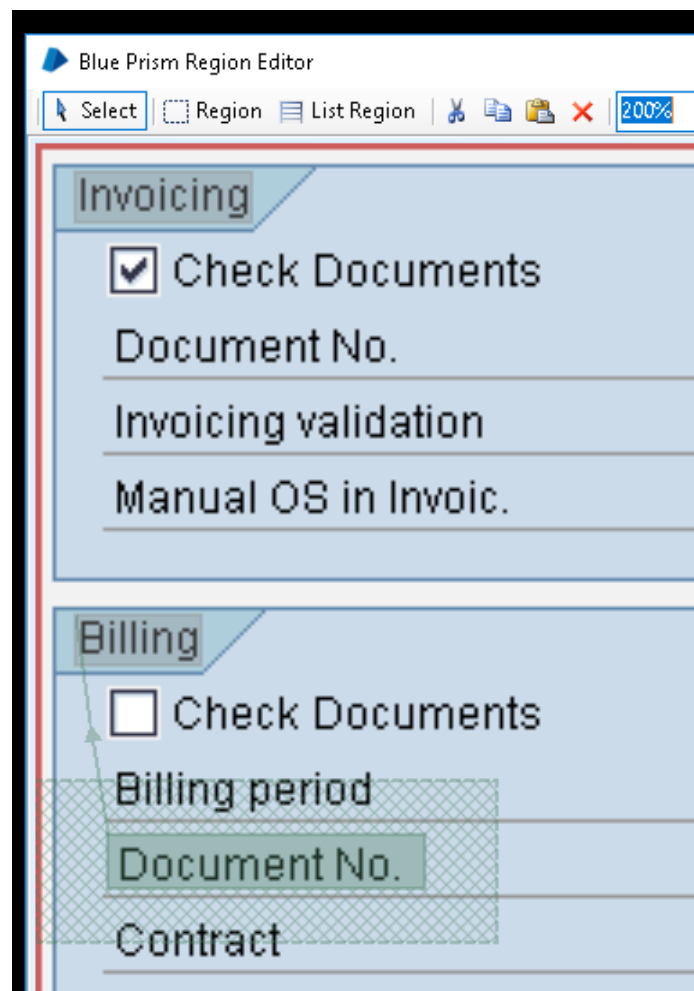
Billing

- ☐ Check Documents
- Billing period
- Document No.**
- Contract

- In the properties of the Document Number region, change the Position property to 'Relative' and observe that a new property called Relative Parent appears. Open the Relative Parent list and select 'Invoicing'.



- Now look back at the Region Editor and you should be able to see a faint arrow indicating the Document Number region is now related to the Billing region.

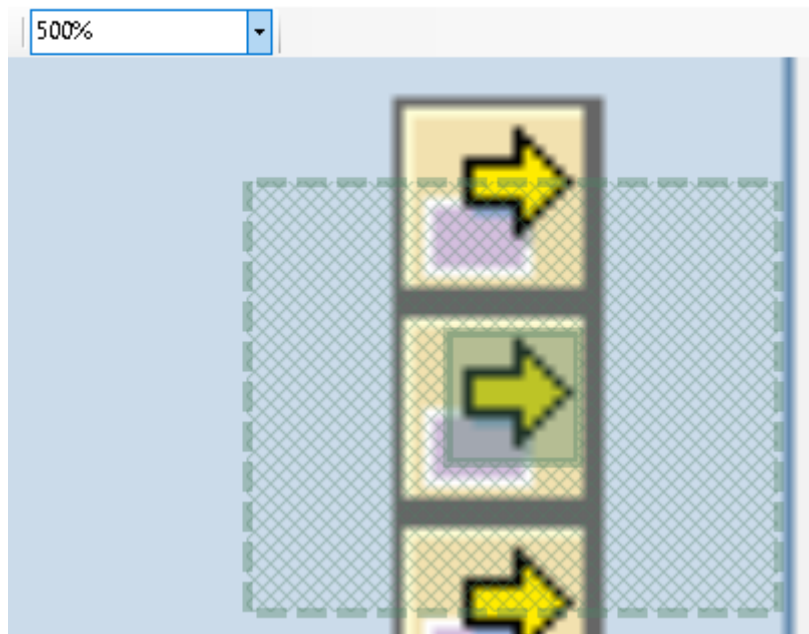


What this means is that when BP attempts to find the Document Number region it will first locate its parent, Billing.

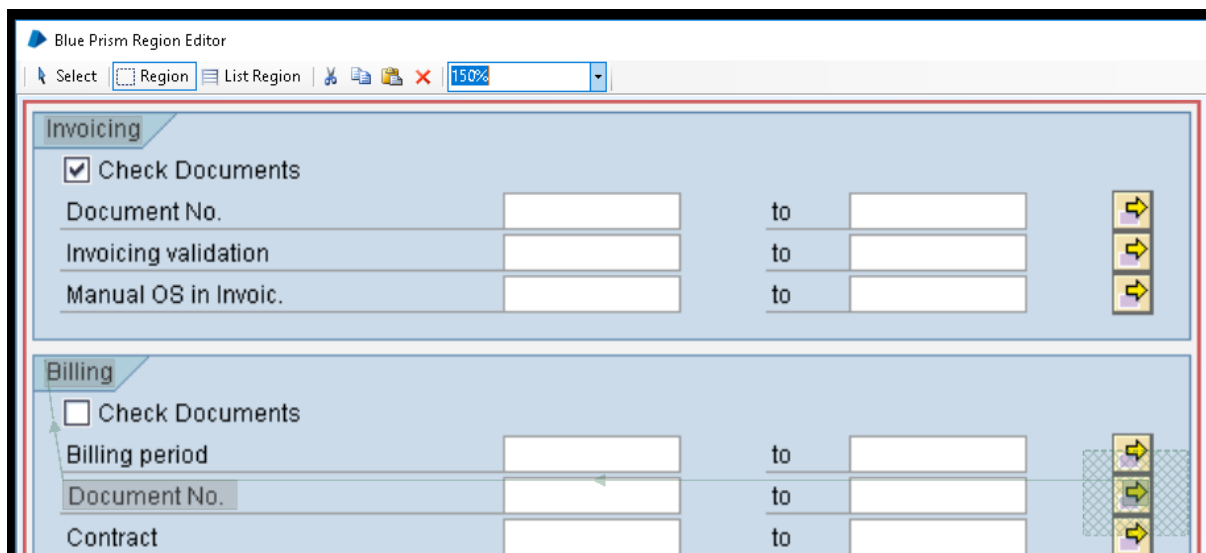
Exercise 1.2.2 Multi-level dependencies

In this exercise we'll push the relationship concept further by creating another level of dependency.

- Return to Region Editor and create a new region around the button to the right of the 'Document No.' label. Make the region fit quite tightly around the arrow, as show below.



- As before, change the properties of this region so that the Position is 'Relative' and the Relative Parent is Document Number.
- You should now see that there is now another faint line linking the button to the label.



- So now there are 2 levels of hierarchy above the button region - the parent, Document Number and the grandparent, Billing.

This means that when BP is asked to find the button, it will start by searching for Billing, then Document No, then the button. This how Surface Automation can accurately replicate business process instructions such as 'press the Billing Document Number button'

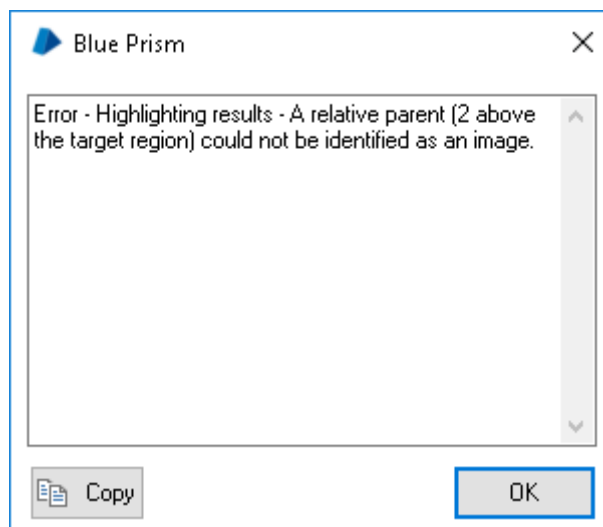
Exercise 1.2.3 Relative position with padding

In this exercise we'll learn how to manage an application that has an erratic layout. We won't concern ourselves why this behaviour occurs, we'll just accept that is how our imaginary application works.

- Open the [Exercise 1.2.3.png](#) in Image Loader. Use the Previous and Next buttons to see that it is similar to the previous image, except that some elements have moved slightly.
- Knowing that billing document number fields and the button are no longer in the same position, try to Highlight the button region.
- You should see that it is still found - why is this? There are two reasons - the dependencies are anchored on the grandparent Billing region, which hasn't moved, and the padding around 'Document No.' and the button are enough to cope with the variance in position.

Exercise 1.2.4 Relative position with maximum padding

- Now load [Exercise 1.2.4.png](#). Observe that this image is different again, with the Billing section now above Invoicing.
- Try highlighting the button. It should fail with the message below, indicating where the hierarchy broke down, namely in trying to find the Billing header.



- Open Region Editor and select the Billing Region.
- Change the Position attribute to 'Anywhere' and press OK.

Billing	
▼ Appearance	
Visible	True
▼ Behaviour	
Font	
Retain Image	True
▼ Identification	
Name	Billing
▼ Region	
> Rectangle	11, 131, 35, 14
▼ Region Location	
Position	Fixed
Location Method	Fixed
> Search Padding	Relative
Colour Tolerance	Anywhere
Greyscale	False

- Now try to highlight the button region again. You should see that it can now be found.

What we have done here is modelled the 'logical location' of the button so that we can carry out the command 'press the Billing Document Number button'. This is a trivial task for a human, but for a robot we have to break the task down, i.e. 'find the billing section, then find the document number field underneath and then press the button on the left'.

In reality we would probably not choose the Anywhere location method for the Billing label because it is potentially uneconomical in terms of computing effort to search the whole screen. Instead the padding option might be more efficient.

Key Point

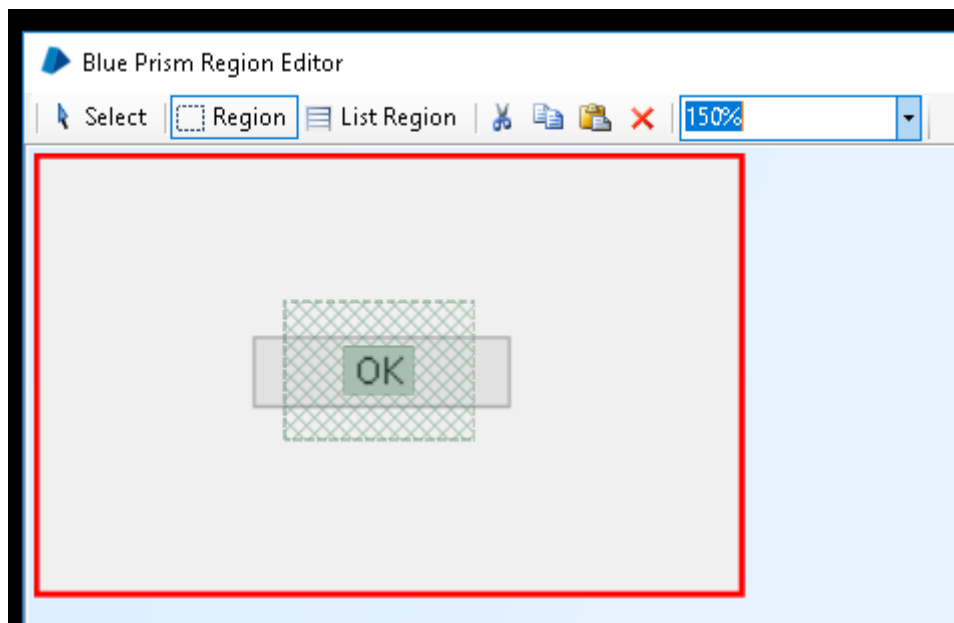
- Surface Automation is entirely dependent on the target application being clearly visible. If the application is minimized or obscured by another window then Surface Automation techniques cannot succeed. This can be difficult to achieve whilst you are developing your objects, and good tip is to avoid maximizing either the application or Object Studio, and instead try to arrange them 'side by side'.

1.3 Coordinate Location Method

Exercise 1.3.1 Creating a basic coordinate region

The previous exercises required you to use the default 'image' region. There is another, simpler type of region available which is essentially a set of coordinates. Prior to version 6, this was the only type of region available, and readers already familiar with Blue Prism version 5 or 4 will already understand how it works.

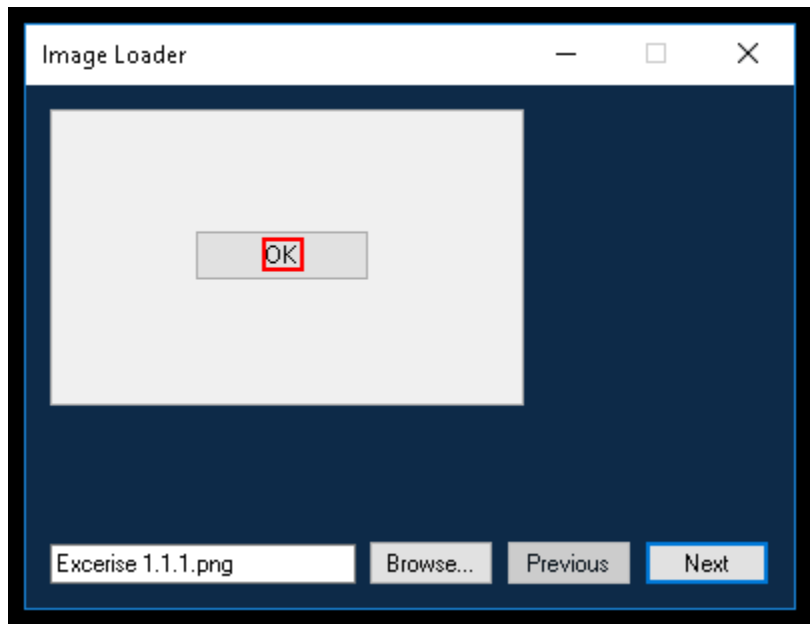
- Open the [Exercise 1.1.1.png](#) image again.
- Add a new element to Application Modeller and identify the Picture Box to open Region Editor as before.
- Create a region over the OK, like below.



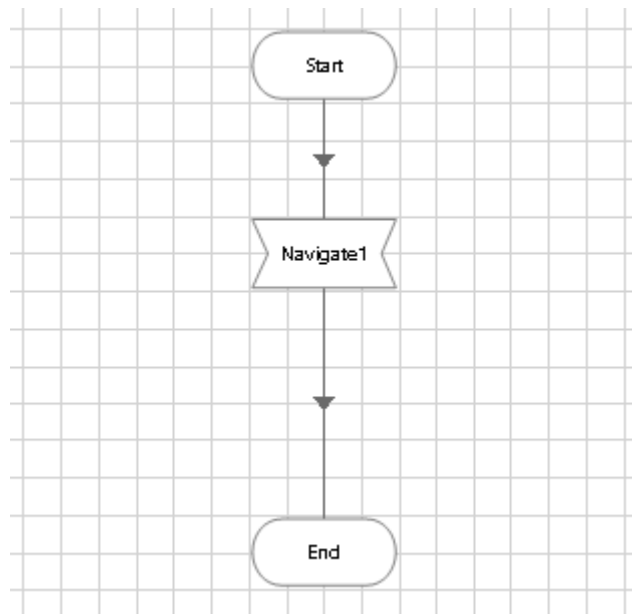
- Go to the region properties on the right and change the [Location Method](#) property to Coordinates and the [Retain Image](#) property to False. We will discuss these properties in due course but for now press OK to close Region Editor.

Region 1	
▼ Appearance	
Visible	True
▼ Behaviour	
Font	
Retain Image	False
▼ Identification	
Name	Region 1
▼ Region	
Rectangle	98, 64, 29, 17
▼ Region Location	
Position	Fixed
Location Method	Coordinates

- Press Highlight and you should see that the OK is highlighted in red.



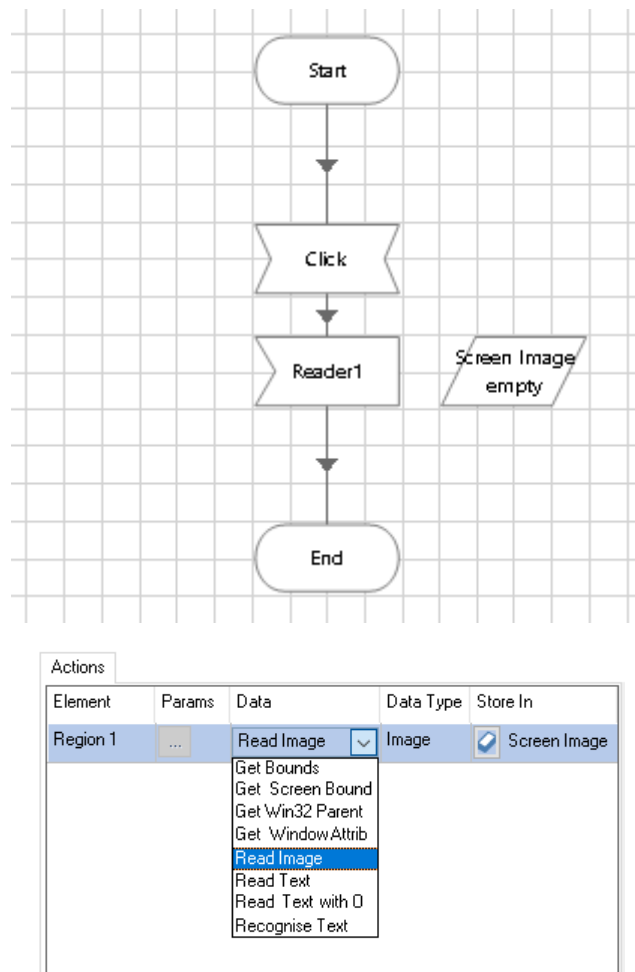
- Add a Navigate stage to a new page and link it between the Start and End stages.



- Open the Navigate properties and set up a **Global Mouse Click Centre** action.

Actions			
Element	Params	Action	Inputs Set
Region 1	...	Global Mouse Click Centre	No
		<div> Click Centre Drop Onto Global Mouse Click Global Mouse Click Centre Start Drag Verify </div>	

- Add a Read stage under the Navigate and set it up to perform Read Image.



- Run the page and you should see the mouse move to the centre of the region. And if you open the properties of your image Data Item and press View, you should see that a screenshot of the region has been captured.

Current Value: 26x14

Clear View... Import.. Export..

Visibility: ☒ Hide from other pages in the process

Initialisation: ☒ Reset to Initial Value whenever this page runs

Exercise 1.3.2 Obscuring a coordinate region

As mentioned above, Surface Automation is reliant on having the target application in view and in a previous exercise we saw that an exception is raised when an image region is obscured. Let's see what can happen with a coordinate region.

- Maximise Object Studio, run the page again and observe the click behaviour. Notice how the mouse just clicks whatever is covering the region.
- Open the properties of the image Data Item and view the image. Notice how the screenshot is taken from whatever is covering the region.

Exercise 1.3.3 False positives with a coordinate region

A coordinate region has no 'memory' of the image, it is simply a geographical area of a Win32 element. So as long as the parent Win32 element exists, the region will be found regardless of whether it is being obscured or what the image within the region currently is.

- Return Object Studio to its previous size covering part of the desktop.
- Open the [Exercise 1.3.3.png](#) image, which is completely white.
- Re-run the page and observe the click behaviour. Notice how the mouse clicks in the same place but this time there is no OK button in the image.
- Open the properties of the image Data Item and view the image. Notice how the screenshot is taken from the same place but this time there is no OK button in the image.

This limitation will be familiar to users of Blue Prism versions 4 and 5. Version 6 however brings the image region which makes Surface Automation easier and more robust.

Key Points

- A coordinate region is a fixed rectangular area of a Win32 element.
- An image region is also the child of a Win32 element, but has the ability to find a screen entity by locating a design time image inside a run time search area.
- Best Practice is to always relate a coordinate region to an image region, to minimise the prospect of a 'false positive'.

2 Data Input

Surface Automation input techniques are very limited compared to the more sophisticated methods available in other spy modes like Win32 and AA. Actions such as Write and Press are not possible because elements like textboxes and buttons do not exist on the local machine. A thin client is the projection of a remote application that has very little substance on the local machine in comparison to a thick client.

So far, we've seen how to direct a mouse click onto an application by locating coordinates or by finding a sub-image. We've also learnt that it's vital that the application is in clear view for images to be found and for mouse clicks to land; if the target application is obscured or minimised then Surface Automation is not going to work.

Another input technique available is to use keystrokes and the *Global Send Keys* method.

2.1 Send Keys

To learn the basics, we will first create an interface to Blue Prism's *Surface Automation Training* application.

Key Point

- Although we can spy the Surface Automation Training application as normal (i.e. using Win32 and AA), for these exercises we are going to pretend that we can't and work with it as a thin client application.

Exercise 2.1.1 SA Training Business Object

- Close Image Loader and your Business Object.
- Create a new Business Object called 'SA Training'.
- Use the Application Modeller wizard to describe the application in the usual way. The executable path will be wherever you have put the application (eg *C:\BluePrism\Training\Surface Automation Training.exe*) and we want to launch the application rather than attach.
- Press the Launch button and open Application Modeller.

Blue Prism Training - 17 Oct 2017

blueprism

Surface Automation Training Log In

User Name

Password

System

[Options](#) Version 1.1

- Rename the default 'Element 1' to 'Log In Window' and spy the application window in Win32 mode.

Exercise 2.1.2 Application focus

In an earlier exercise, we stipulated that Object Studio must not be maximized and that the target application must be clearly visible. We can ensure the application is in focus by using the **Activate Application** action.

- Close Application Modeller and add a new Navigate stage to the Action1 page.
- Open the Navigate properties form and drag in the Log In Window element.
- Select the Activate Application action from the drop-down list.
- Enter 0.5 in the 'Pause After Each Step' field. This field is used to slow down the execution speed of a Navigate stage. With some actions, and Activate is one of them, applications do not respond instantaneously, and it is often necessary to intersperse commands with short pauses.

Navigate Properties

Name:

Description:

Application Explorer

Filter the tree...

- SA Training
 - Log In Window

Actions

Element	Params	Action	Inputs Set
Log In Window		Activate Application	N/A

Pause After Each Step (timespan/secs)

Move Up Move Down Add Remove

Inputs

Name	Datatype	Value

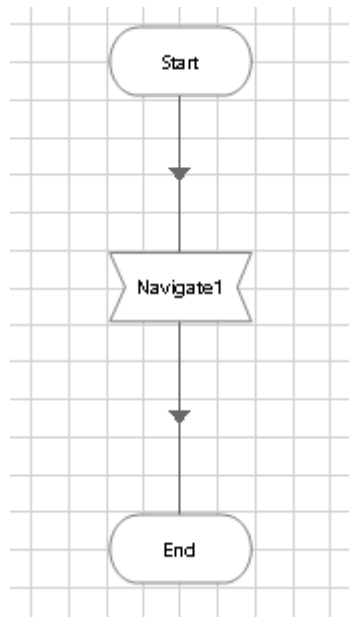
Stage logging:

Warning threshold:

Number of minutes (0 to disable)

OK Cancel

- Press OK to close the Navigate properties and link up your Start and End stages. Your page should simply look like this.



- Run the page and you should see that it brings the application into focus. Move Object Studio so that it covers the application and run the page again.

Key Point

- In Blue Prism V6 the use of 'dumb' Wait stages to create pauses after Navigate stages has been reduced by the introduction of the Pause After Each Step field in the Navigate stage properties.

Exercise 2.1.3 Global Send Keys

- Go back the Navigate properties form.
- Add a new row and drag in the very top (root) node from the Application Explorer tree.
- Select the **Global Send Keys** action from the list and enter a name of your choice as an expression in the Text input field.

Navigate Properties

Name:

Description:

Application Explorer

Filter the tree...

- SA Training
 - Log In Window

Actions

Element	Params	Action	Inputs Set
Log In Window		Activate Application	N/A
SA Training		Global Send Keys	No

Pause After Each Step (timespan/secs)

Move Up Move Down Add Remove

Inputs

Name	Datatype	Value
Text	Text	John
Interval	Number	

Stage logging:

Warning threshold: Number of minutes: (0 to disable)

OK Cancel

- Press OK and re-run the page. You should see the User Name field being populated.

Blue Prism Training - 17 Oct 2017

blueprism

Surface Automation Training Log In

User Name

Password

System

Log In Reset

[Options](#) Version 1.1

- Note that the text was entered rather quickly. Not all applications can cope with keystrokes sent at super-human speed, and as we did with Activate, it can sometimes help to slow things down a little. The Interval input provided for Global Send Keys is used to create a pause between keystrokes. For this exercise 0.5 would be too slow, so try 0.1 instead.

Inputs		
Name	Datatype	Value
Text	Text	"John"
Interval	Number	0.1

- Re-run the page (manually press Backspace to clear the field first) to see the effect of the interval. You should observe that the keystrokes are entered at a more human-like pace.

Key Point

- In Blue Prism V6 the use of 'dumb' Wait stages to create pauses between keystrokes has been reduced by the introduction of the Interval field in the Global Send Keys and Global Send Key Events actions.

Exercise 2.1.4 Field focus

In the previous exercise you may have noticed that we were only able to populate User Name field because the cursor happened to be in the right place. If you manually put the cursor into the Password field and run your page again, you'll see your Business Object naively sends keys to wherever the cursor happens to be. Evidently this needs better control.

- Create a new element in Application Modeller and spy the window again, but this time with Region mode.
- In Region Editor create a coordinate region that sits just inside the User Name field.
- Back in Object Studio, open the properties of your Navigate stage and add a new line between Activate Application and Global Send Keys.
- Drag in the region you just created and select the Global Mouse Click Centre action.

Actions			
Element	Params	Action	Inputs Set
Log In Window		Activate Application	N/A
User Name	...	Global Mouse Click Cen	No
SA Training		Global Send Keys	Yes

- Manually clear both fields in the training application and put the cursor in the Password field.
- Run the page again and you should see that the mouse clicks into the User Name field before the keys are sent.

The Password field can be populated in exactly the same way.

- Return to Region Editor by pressing the Regions button.

- Create another coordinate region for the Password field and then return to Object Studio.
- In your diagram you have the option of adding more lines to the properties of your existing Navigate stage, or you could add a second Navigate stage to handle the Password field. A single Navigate will not be any more efficient than two, it's more a question of diagram presentation. Note that there is no real need to perform Activate Application twice, once should be enough.

Now let's imagine that the User Name field is already populated with some text.

- Manually enter the name 'Denis' into the User Name field and then click in the Password field.
- Run your page again to see that it ignores the existing User Name text and just adds more. Clearly we need to guard against this.
- In the Navigate properties, introduce a new row under the Global Mouse Click Centre row.
- Set up a Global Send Keys action as before, but this time use the expression `"{HOME}{DELETE 20}"` as the input value. Leave the Interval input empty.

Element	Params	Action	Inputs Set
Log In Window		Activate Application	N/A
User Name		Global Mouse Click Centre	No
SA Training	...	Global Send Keys	Yes
SA Training		Global Send Keys	Yes

Pause After Each Step (timespan/secs)

Move Up Move Down Add Remove

Name	Datatype	Value
Text	Text	"{HOME}{DELETE 20}"
Interval	Number	

- You should be able to guess what will happen next. Run the page again to find out.

Key Point

- Although tabbing between fields can work, it is not as robust as clicking into a region that has been identified via a label. Not only is the tab order of an application susceptible to change, but there is also less certainty that the cursor will be in the right place when you start to enter a value.

Exercise 2.1.5 Paste from clipboard

Often in Surface Automation it can be useful to employ the clipboard as a means of data entry. The clipboard can be pre-set programmatically so that input can be pasted in.

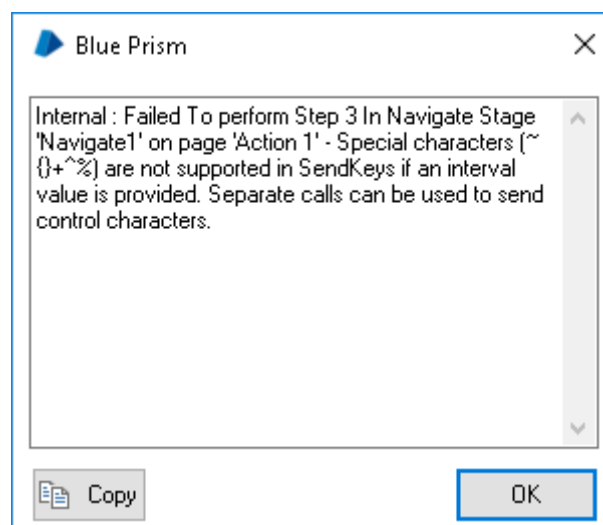
- Link in an Action stage above the Navigate stage.
- In the Action properties select the **Utility – Environment** object and the **Set Clipboard** action with the expression **“Chris”** as the input value.
- Edit the Navigate stage properties so that the Global Send Keys text is the expression **“^v”**, which is code for CTRL+v. Remove the Interval input as it’s unnecessary here. If you leave it in you’ll see an error message because when Global Send Keys contains special characters like ^, an interval cannot be used.

Inputs		
Name	Datatype	Value
Text	Text	“^v”
Interval	Number	

- Re-run the page to see the new user name value pasted in. The choice of whether to use paste or keystrokes is dependent on the target application and the choice of the Business Object developer.

Key Point

- The Interval input for Global Send Keys must be left blank when sending ‘special’ key instructions like **{HOME}** and **{DELETE}**



2.2 Mouse Click

By now you will be familiar with clicking into a region, so the following exercise is intended to stretch you a little by providing only basic guidance.

Exercise 2.2.1 Login Action

- Create a new page in your Business Object called 'Log In'.
- Create 'image location' regions for the User Name and Password labels. Don't worry about the padding, just use the default values.
- Modify your 'coordinate location' regions for the User Name and Password text boxes and make them relative to their respective label regions.
- Create 'image location' regions for the Log In and Reset buttons.
- Use Navigate stages to create the 'Log In' logic. Remember to use pauses.
- The User Name must be at least 3 characters but can be any value. The password must be the same as the User Name.
- Ignore the System drop-down and for now and for now don't worry about waiting after clicking the Log In button.

Once you are happy with your Log In page, save and close your object. Close the training application too, we'll come back to it later in the next Surface Automation course.

3 Data Output

As with data input, Surface Automation has limited options for getting data out of an application because of the paucity of information offered by a thin-client. In some situations, the clipboard can be useful, but often the only option is to analyse the image, either by OCR or by Blue Prism's own technique of **Character Matching**.

3.1 Character Matching

Character Matching ability to discern text characters from an image. In brief, the technique involves creating a template of font characters which can then be used to inspect a screenshot, and by finding matching patterns of pixels, characters and words can be deduced.

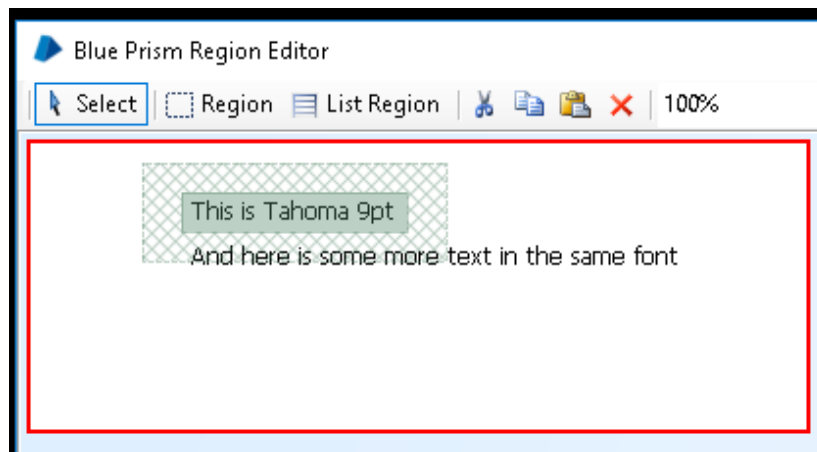
Identifying Fonts

We will now return to the Image Loader application to learn how to read image text without using the standard spying techniques.

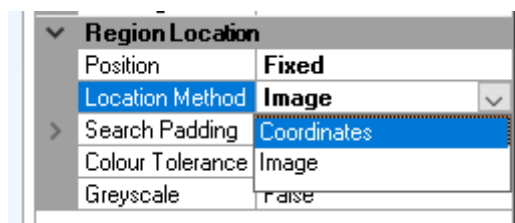
Exercise 3.1.1 Region font

In this exercise we'll associate a region with a font.

- Launch Image Loader and open the **Exercise 3.1.1.png** image. As you can see, this image is deliberately simple.
- Create a new region for the first line.

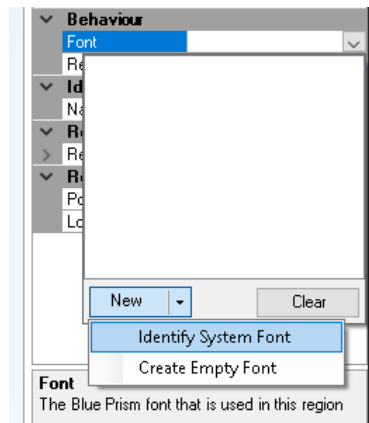


- In the region properties, change the default Location Method to 'Coordinates'.



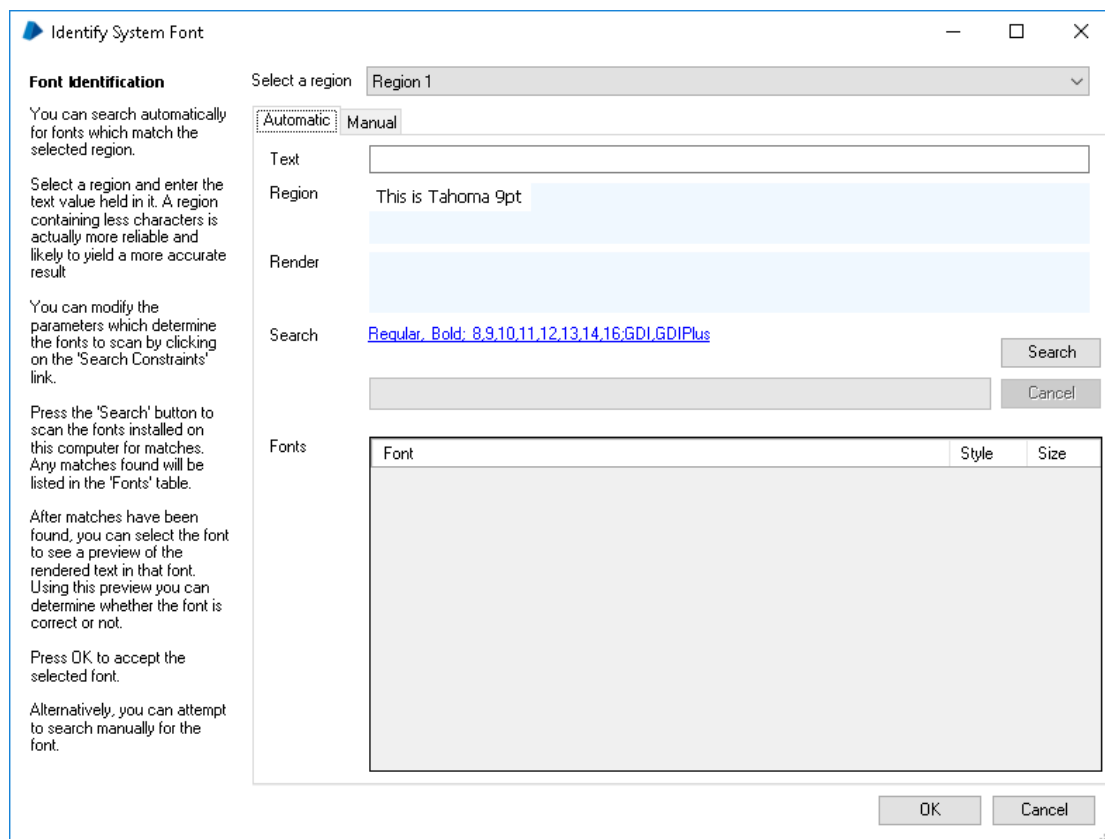
Key Point

- Regions to be used for reading text must have the Location Method property set to 'Coordinates'. The default 'Image' method makes the 'design time' image part of the region's identity and as such the region won't be discovered if the 'run time' image is different.
- Still in the region properties, select the Font drop down and then select **New -> Identify System Font**.



Key Points

- At this stage you may receive a warning message regarding **Font Smoothing**.
- Font Smoothing is a Windows setting designed to make fonts easier for people 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.
- See the Appendix at the end of this document for how to switch off Font Smoothing.
- With Font Smoothing disabled, selecting **Identify System Font** will open the Identify System Font screen, as shown below.



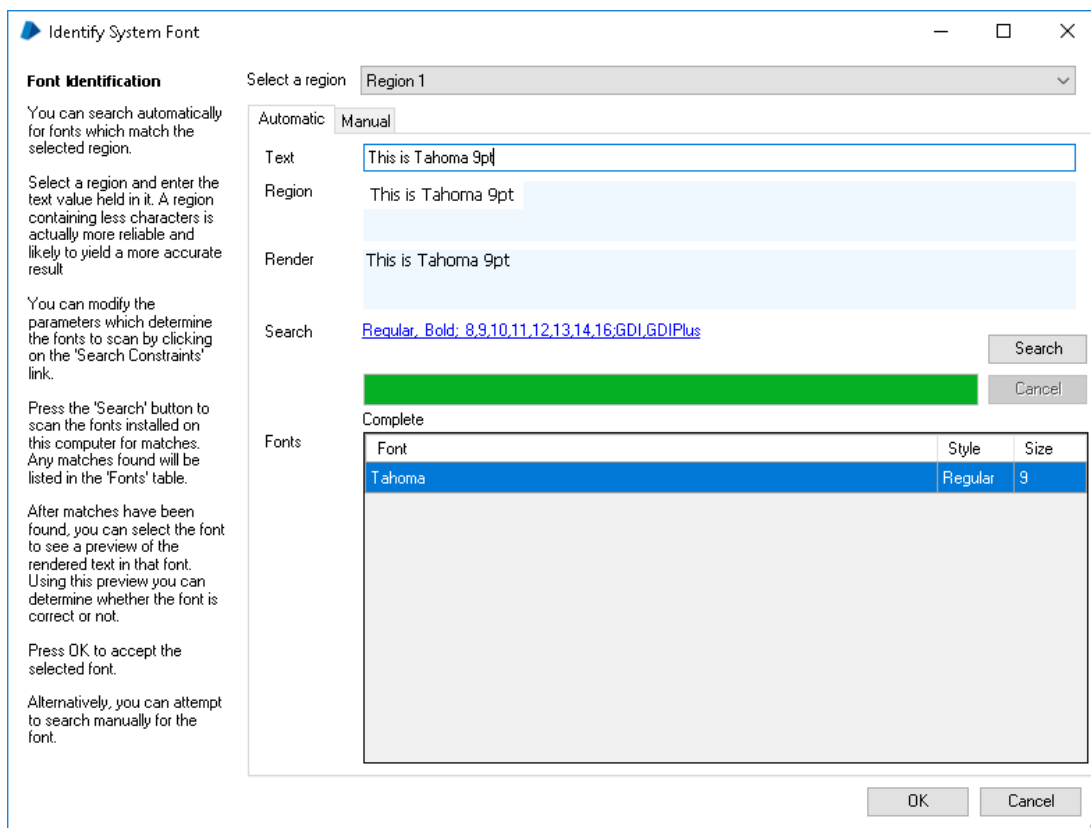
Exercise 3.1.2 Identifying a system font (Part 1)

Before we can match characters, we need to first identify a font. Read the instructions on the left of the Identify System Font window to familiarise yourself with how fonts are identified.

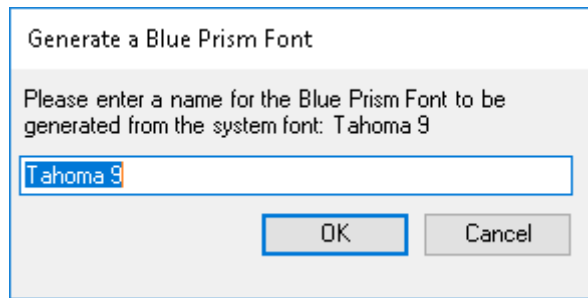
- In the Text field, type in the same characters shown in the region, i.e., ***This is Tahoma 9pt***. This tells Blue Prism what characters we are looking to match.
- Click the Search button.
- As it searches, Blue Prism creates an image of the text value as it would appear using each font, and compares each 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 the text is rendered in that font.

Key Point

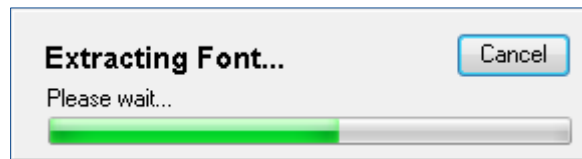
- Automatic search requires a degree of experimentation; it may not yield any results or might produce only partial matches.
- The target image has been made easy, so you should find that a single match is found, as below.



- Select the Tahoma 9 font and click OK. You can rename the font at this point but in this case, we will simply accept the default name.



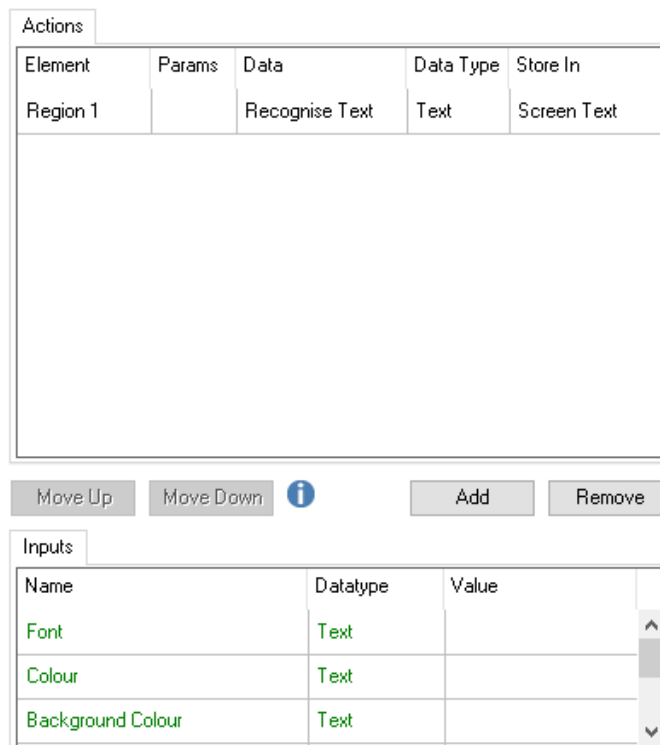
- Blue Prism will then generate a font definition and store it in the database. This will make the font definition available to other Blue Prism machines connected to the database.



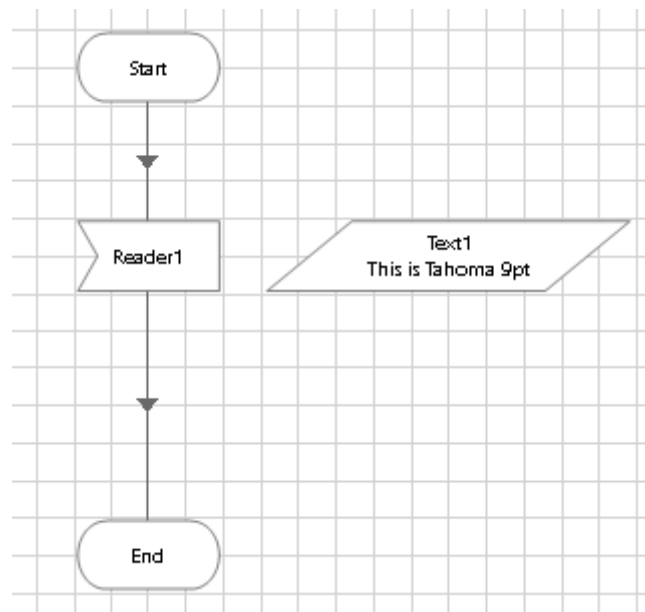
Exercise 3.1.3 Recognise Text

Now we have identified the Tahoma 9 font, we can use a Read action to interpret the text from the image.

- Close Application Modeller and create a new page.
- Add a new Read stage to the page.
- Open the Read stage properties and drag in the region from the Application Model.
- Use the **Recognise Text** action to extract text data from the region and store it in a Data Item. Ignore the optional inputs for now.



- Run the page and you should see that the text is recognised. Remember that if Image Loader is obscured by Object Studio or anything else, Surface Automation cannot succeed.



- If you are having difficulty with Recognise Text, it can help to capture the run time image in a Data Item.
- Add a second row to Reader1 and use **Read Image**.

Actions				
Element	Params	Data	Data Type	Store In
Region 1		Recognise Text	Text	Text1
Region 1	...	Read Image	Image	Image1
		Get Bounds Get Screen Bound Get Win32 Parent Get WindowAttrib Read Image Read Text Read Text with O Recognise Text		

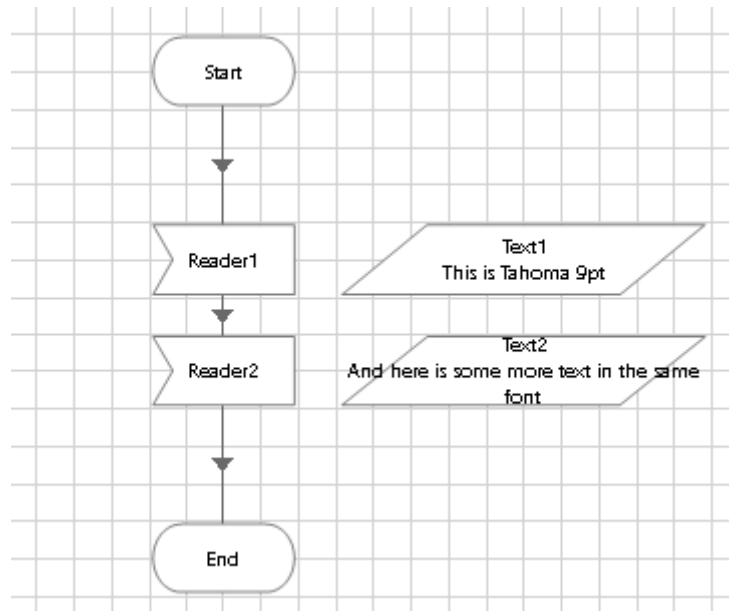
- Run the page again, open the image Data Item and press View to see the run time image that Recognise text is trying to interpret. You might be able to spot the problem, perhaps the region is incorrectly positioned.

Current Value	26x14	Clear	View...	Import..	Export...
Visibility	<input checked="" type="checkbox"/> Hide from other pages in the process				
Initialisation	<input checked="" type="checkbox"/> Reset to Initial Value whenever this page runs				

Key Point

- By looking at the properties of the image Data Item you will be able to see the image taken from the screen, and this might help you understand the problem.

- Go back into Region Editor and create a new region for the second line of text.
- Select Tahoma 9 as the region's font and press OK.
- Create another Read stage for the new region.
- Run the page again and you should see that the second line of text is recognised too.



Exercise 3.1.4 Colours and Recognise Text

By default, Blue Prism expects the text in region elements used with Recognise Text to be black. This is why in the previous exercises there was no need to provide inputs to the Read stage. You may have noticed in the Read stage properties that Recognise Text has colour input parameters, and in this next exercise we'll see how they work.

Inputs		
Name	Datatype	Value
Font	Text	
Colour	Text	
Background Colour	Text	

- Open the [Exercise 3.1.4.png](#) image and see that it is a coloured version of the previous image.
- Run the page again to confirm that Recognise Text no longer works. This is because Blue Prism is anticipating black text but can't find any.
- Open the first read stage, enter the expression `"FF0000"` in the Colour input value and press OK.
- Re-run the page and you should see that the first region is now recognised.
- Open the second read stage and enter `"0000FF"` in the Background Colour input field and press OK.
- Re-run the page and you should see that the second region is now recognised.


As you will probably have worked out, the two Read stages are now functioning again because we have added colour information. We told the first Read stage to interpret red pixels as text, and in the second Read stage we indicated that blue was the background colour and all other pixels were to be assumed as text.

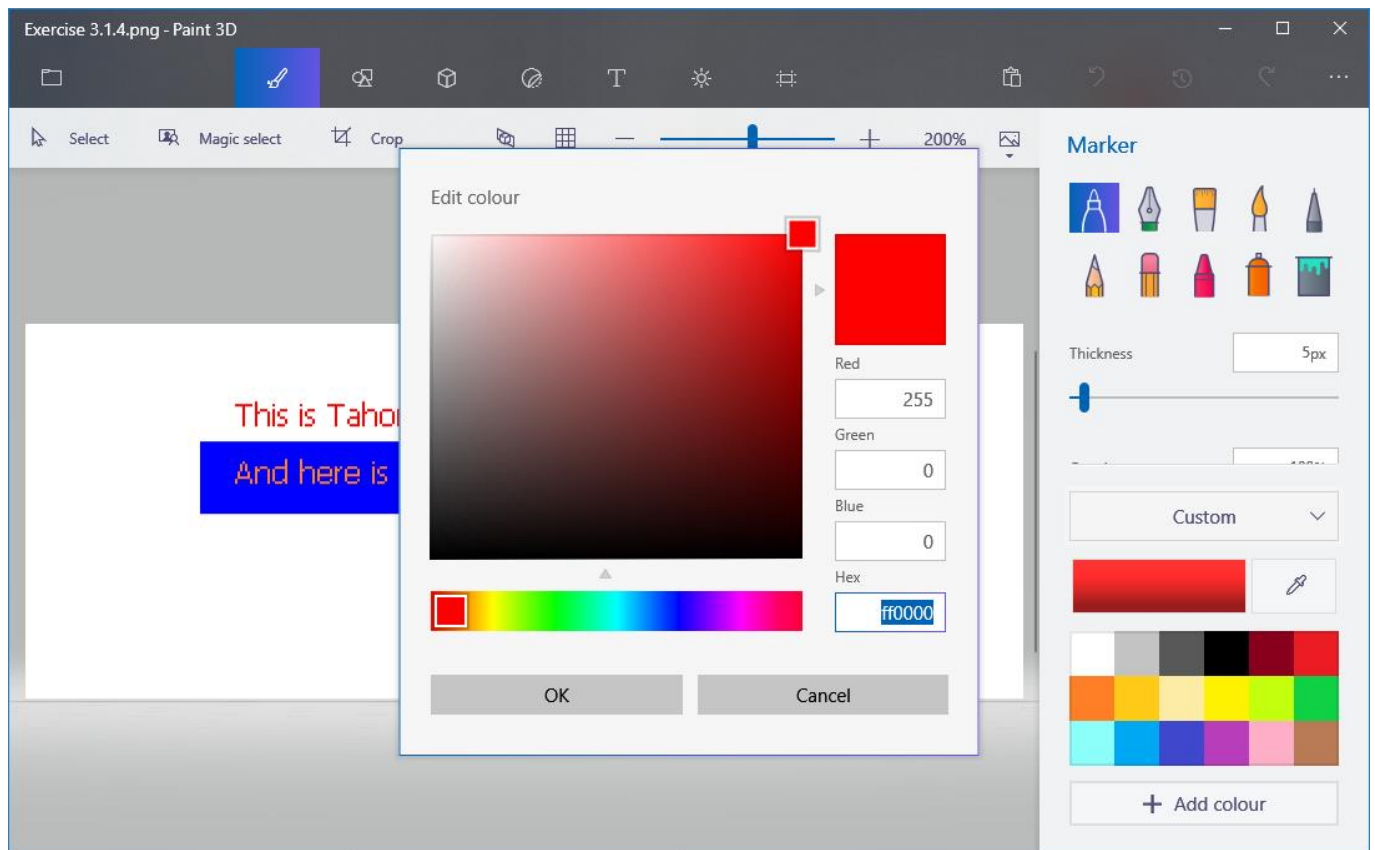
Key Point

- Currently Read stages employ a technical notation for colours, known as **hexadecimal**, or hex for short. This isn't the most user-friendly input mechanism and at the time of writing (October 2017) there are plans to introduce a colour picker.

Exercise 3.1.5 Hex codes

Microsoft's Paint 3D is an easy way to find out what the hex code of a colour is.

- Open the [Exercise 3.1.4.png](#) image in Paint 3D and zoom in to about 200%.
- Use the colour picker  to select the red text and then press the button to the left of the colour picker.
- Note the Hex field in the Edit Colour dialog box.



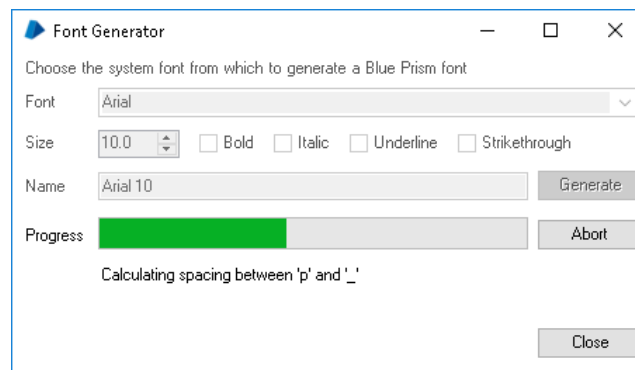
- If you don't have Paint 3D there are many useful resources online.

Exercise 3.1.6 Fonts and Recognise Text

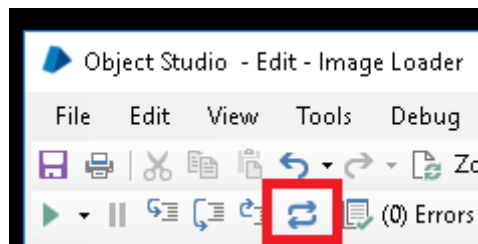
With the regions we have created so far, we've indicated what the font will be in Region Editor. This is not essential, as there is a way to specify the font in the Read stage.

- Open the [Exercise 3.1.6.png](#) image and notice that different fonts have been used.

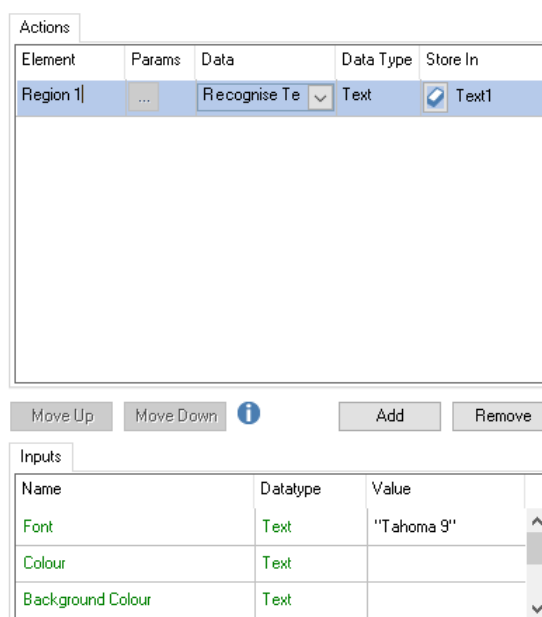
- Create new region elements for each line, remembering to set the Location Method property to 'Coordinates'. However, when it comes to the Font property, leave it blank for all three regions.
- Close Region Editor and then close Application Modeller.
- Leaving Object Studio open, go to the main Blue Prism window and navigate to **System->System->Fonts**, where you should see the Tahoma 9 font we created earlier.
- Press **New -> From System Font** and generate a font definition for Arial 10.



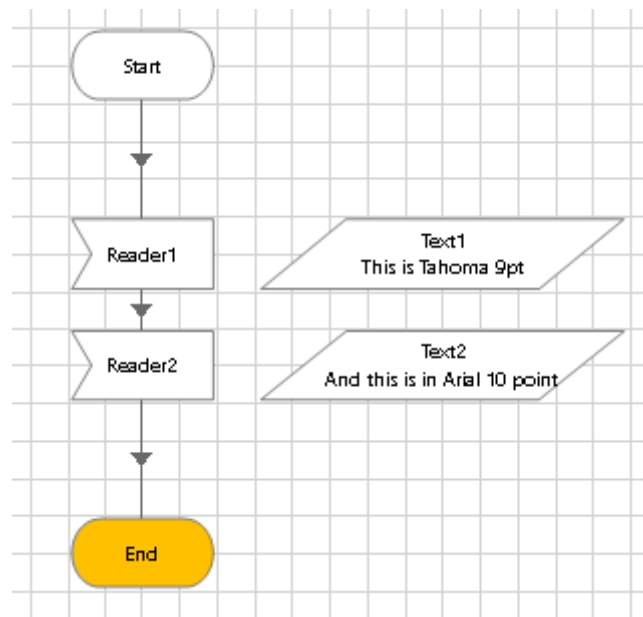
- Now return to Object Studio, create a new page and press the Reset button, shown below in red. This is to make the object reload all font definitions from the database.



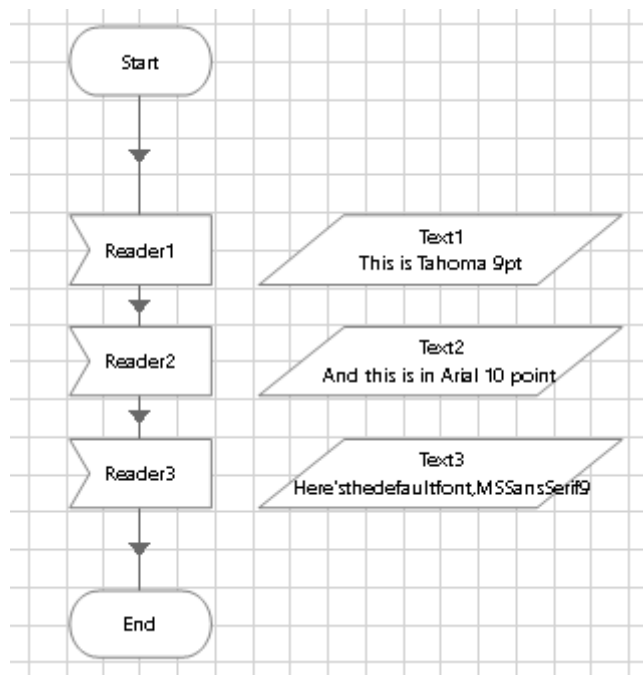
- Link two Read stages into the new page, and set them up to use Recognise Text on the first two regions. This time however, populate the Font input field with the name of the appropriate font.



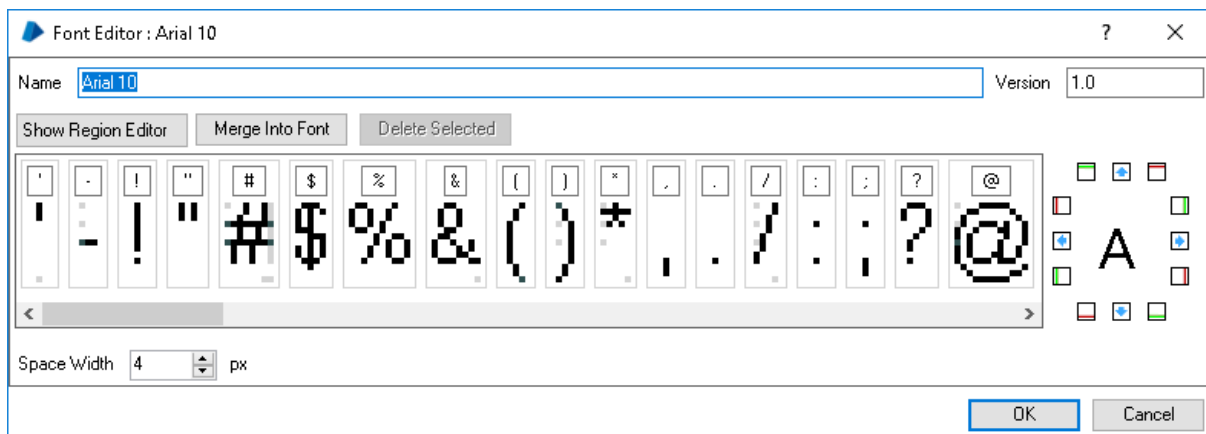
- Run the page and you should see that the first two lines of text are read correctly.



- Now add in another Read stage for the third region. This time however, don't specify any font.
- Run the page again and you should see that the third region is read, albeit not as well as the others.



MS Sans Serif 9 is the default font when no font is specified, either in Region Editor or in the Read stage. However, because the font has not been explicitly defined, Blue Prism does not know how to interpret whitespace. If you go back to **System -> Fonts** and press **Edit**, you will see that a font definition contains a **Space Width** value.



Recognise Text Input Parameters

As we have seen, the Recognise Text method has input parameters such as Font, Colour, and Background Colour values. This is a summary of their usage.

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 default font.


Colour

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

Background Colour

- This is the hex code of the background colour the Read stage should use.
- It must be left blank if a foreground colour has been specified.
- It can also be set to **“auto”** to indicate that the least prevalent colour in the region is the foreground colour. This is useful if the colour of the text in the region is changeable. Note however it cannot be used if the text is not rendered in a single colour, e.g. as a gradient colour.

Split Lines, Use Original Algorithm and Erase Blocks

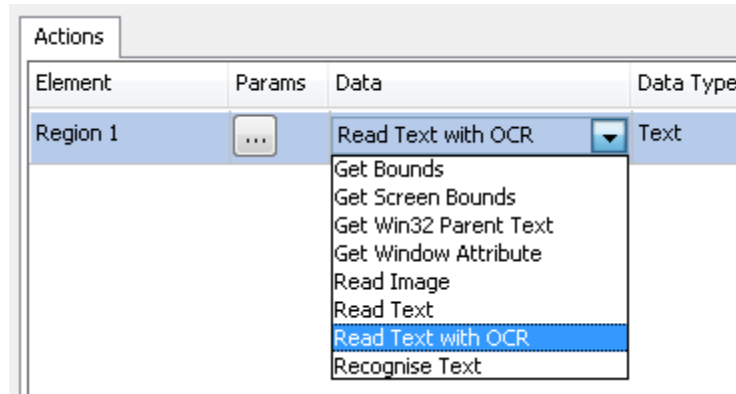
- These settings have very specific uses that we won't go into in here and for now you can ignore them. As always, the blue  button can be used to find out more.

3.2 Read Text with OCR

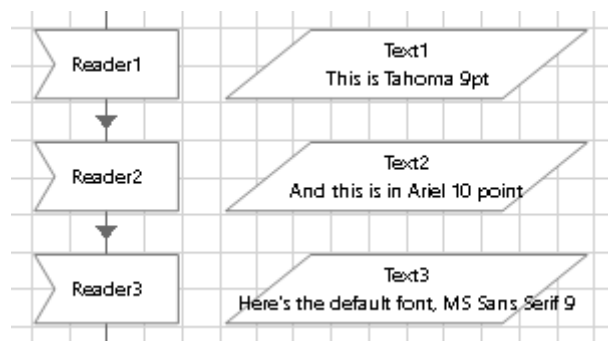
Blue Prism's **Read Text with OCR** action uses Google's **Tesseract** open source OCR (Optical Character Recognition) engine to be able to read text without identifying the font or disabling font smoothing. This document will only cover the basics and won't go into all the details of OCR; there is a guide on the Blue Prism Portal for further reading.

Exercise 3.2.1 Read Text With OCR

- Return to the three Read stages created in the previous exercise.
- Change each one so that instead of using Recognise Text they use Read Text with OCR.



- With the [Exercise 3.1.6.png](#) image showing in Image Loader, run the page again. You should see that all three lines are read, apparently successfully.
- Look again at the Data Items, is everything correct? On the author's machine the second line is wrongly interpreted as Ariel, instead of Arial. This is an example of the fallibility of OCR.



Key Points

- OCR is **heuristic** and can result in a 'false positive' or a 'false negative'. An example of a false positive is when the OCR incorrectly determines that some text value exists on the screen, when in reality it does not. A false negative is where OCR mistakenly decides that a value does not exist, when in fact it does.
- By contrast Character Matching is **deterministic**, either there is a 100% match with the font definition or there is no match.
- Care should always be taken when using any OCR technology. Quality cannot be guaranteed in advance, and only through large scale testing of your specific use case will you know if the technology is suitable for your solution. Where possible Recognise Text should always be used instead.

4 Synchronisation

As you should know from your Foundation training, the synchronisation of a Business Object with the behaviour of its target application is fundamental to the success of Blue Prism solution. Logic built on fixed, 'unintelligent' pauses results in weak and inefficient logic that is bound to fail. Without 'intelligent' Wait stages integration cannot succeed, and Surface Automation is no different.

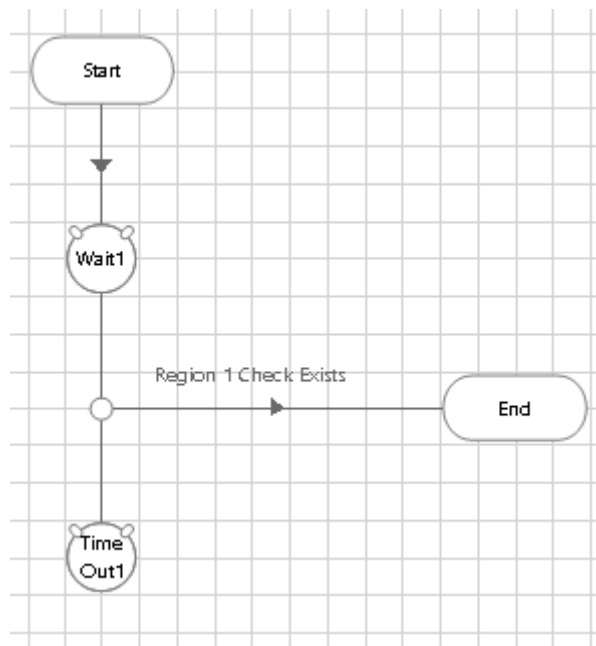
Exercise 4.1.1 Waiting for an image region

Here we will learn how to use a wait stage to tell us when an image has been found.

- Open [Exercise 4.1.1a.png](#) and then open [Exercise 4.1.1b.png](#) in Image Loader.
- Create a region around the red icon and press OK to close Region Editor.
- Select the region element and press Highlight to confirm it can be found.
- Press the Previous button on Image Loader and then press Highlight in Application Modeller. You should see that the region can no longer be found.
- Go back to Object Studio, create a new page and add a Wait stage.
- Open the Wait stage properties, drag in the region, and select the Check Exists condition.

Actions					
Element	Params	Condition	Type	Comparison	Value
Region 1		Check Exists	Flag	= (Equal)	True

- OK the properties window and link the Wait stage node to an End stage. Don't bother with a time out Exception stage for this exercise.



- Before you run the page, ensure Image loader is not obscured – remember Surface Automation cannot function unless the target application is clearly visible. Also check that it is the blue image that is on display.

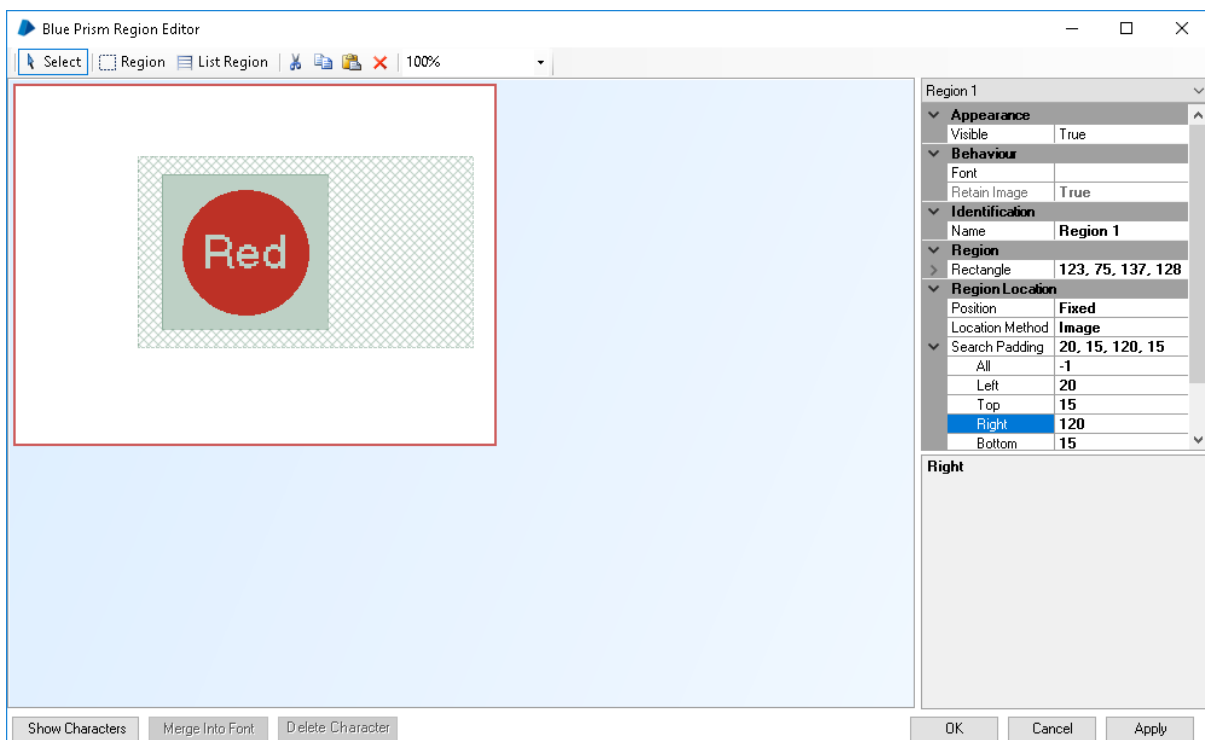
- Press play in Object Studio and note that the Wait stage hangs for 5 seconds before timing out. This is to be expected because we are checking for the red icon.
- Reset the page and run again but this time, while the wait stage is hanging, press the Next button on Image Loader. You should see that the Wait stage detects the red icon and goes to the End stage.

This exercise should enable you to see that a Wait stage using Check Exists can tell you when an image region has arrived, much like you have seen Check Exists do for a Win32 element.

Exercise 4.1.2 Waiting and Search Padding

As we have discussed previously, it's not uncommon that the image you modelled at design time is not in exactly the same place at run time. And as some of you may have already anticipated, a Wait stage can also use the Search Padding of a region to its advantage.

- Open Region Editor and expand the Search Padding property of the region. Increase the Right value to about 120, so that the cross-hatched padding extends towards the edge of the white area.



- Return to Object Studio and open [Exercise 4.1.2.png](#) before re-running your page. You should see that the increased padding has enabled the red icon to be found in its new location.

Key Point

- The Wait stage searches for images within the confines of the Search Padding area. It looks first in the original 'design time' location and then expands the search outwards.

Exercise 4.1.3 Waiting and colour tolerance

Sometimes the presentation technology (e.g. Citrix) can affect the colours of a thin-client. This difference is usually imperceptible to the naked eye but can manifest itself as run time images that no longer match the design time images. Often you will find that the colours are very slightly different, making an exact match is no longer possible. Blue Prism offers an option to deal with this problem.

- Open [Exercise 4.1.3.png](#). Although it looks similar to [Exercise 4.1.2.png](#), the red is a different shade.
- Test that the image is different by re-running your page. You should get a time out.
- Open Region Editor and set the Colour Tolerance property to 50.

Region Location	
Position	Fixed
Location Method	Image
Search Padding	20, 15, 120, 15
Colour Tolerance	50

- Run the page again and you should find that the Wait stage can find the image again.
- Go back to [Exercise 4.1.2.png](#) and check that the page runs with that image too.

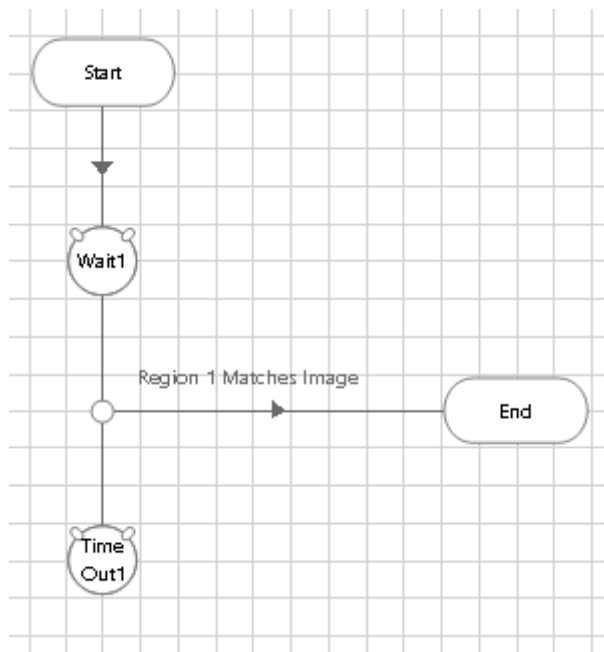
Here Blue Prism is deeming any pixels whose RGB values are within 50 units of each other to be the same. In reality the colour variance will usually be far lower than 50, more like somewhere between 1 and 10. As with all the exercises so far, the example images have been deliberately simplified to illustrate a point.

Exercise 4.1.4 Coordinate regions and Retain Image

Now that you are beginning to understand the 'image' region, let's now go back to its predecessor, the 'coordinate' region. Hopefully readers familiar with Blue Prism version 4 or 5 will appreciate that the image region is new to version 6 and offers significant improvements over the coordinate region. The coordinate region is still vital for reading text, but its rather limited benefit to Wait stages will now become less important.

Recall that in exercise 1.1.1 you were asked to set the Retain Image property of a coordinate region to False.

- Go back to the [Exercise 1.1.1.png](#) image.
- Create a region around the OK as before and set Location Method to Coordinates. This time however leave the Retain Image property as True.
- Come out of Application Modeller and create a new page.
- Add a Wait stage that uses the [Matches Image](#) condition. This condition has an Image input parameter but you can leave it blank.
- Link the Wait stage node to the End, like this.



- Run the page and see that the Wait stage will indicate that the region exists.
- Open the [Exercise 1.1.2a.png](#) image and run the page again. The wait stage should time out.

The Retain Image property does enable a comparison of the run time image with the design time image, but the inflexibility of the coordinate region means that it would be easier to use an image region instead.

5 Summary

This is the end of this introductory Surface Automation course and you should now be in a position to tackle the advanced course. Here is a summary of the main points we have learned so far.

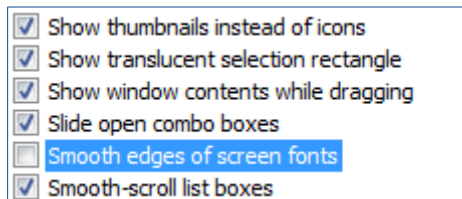
- Surface Automation is a technique for working with images. Primarily it is used on thin-clients but it can also be applied to any application with a user interface.
- Like any other type of integration technique, intelligent Wait stages are essential to producing robust Business Objects.
- Surface Automation is wholly dependent on the visibility of the target application.
- Input is limited to mouse clicks and keystrokes and, if available, pasting from the clipboard.
- Output is limited to Character Matching, OCR and, if available, copying to the clipboard.
- Character Matching can only work when Font Smoothing is switched off.
- Although very useful, no OCR is 100% reliable.
- Coordinate regions must be used for reading text. Image regions are not suitable because they are used to compare a 'design time' image with a 'run time' image.
- With image regions, Wait stages can be used to detect visual changes in application state.
- Version 6 introduces new powerful features such as Search Padding, Relative Positioning and Colour Tolerance.
- It's Best Practice to make coordinate regions dependent on an image region. For example, using an image region for a field label and a related coordinate region for the field.

6 Appendix

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.

- To switch off font smoothing in Windows 10, 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.



Further Reading

- Guide to Reading Text with OCR
- Surface Automation of Terminal Emulators
- Surface Automation - Guide to Modifying Fonts