

Expanding the SAP function library

Contents

Introduction	2
Pre-Requisites	2
Procedure.....	2
Editing the SAPElements file	3
Editing the Actions file	4
Functions for working with SAP controls.....	5
SAPGetProperty	5
SAPSetProperty	5
SAPInvokeMethod	6
Testing new functions.....	7
Example of adding a new action to an existing control.....	8
Appendix 1: List of Argument Data Types.....	10
Appendix 2: List of SAP Controls.....	11
Appendix 3: Useful references.....	13
Troubleshooting.....	14

Introduction

Although Blue Prism provides accessibility to most of the main controls and functions which SAP exposes via its Application Programmable Interface (API) there are some controls and functions which have not been implemented as standard within the SAP product. Instead, Blue Prism have created a method to enable controls and functions to be added to the list of exposed items which can be spied and interacted with. This document explains the procedure to expose these features and make them available.

This procedure is not expected to be attempted by the majority of customers. If you need assistance with creating and validating new methods then please raise a support request for help.

Pre-Requisites

It is assumed that the following software products are installed and configured before using this procedure:-

- Blue Prism v4.2.43 or higher
- SAP GUI for Windows (SAP Netweaver 7.1). The system ID for this is "NSP".

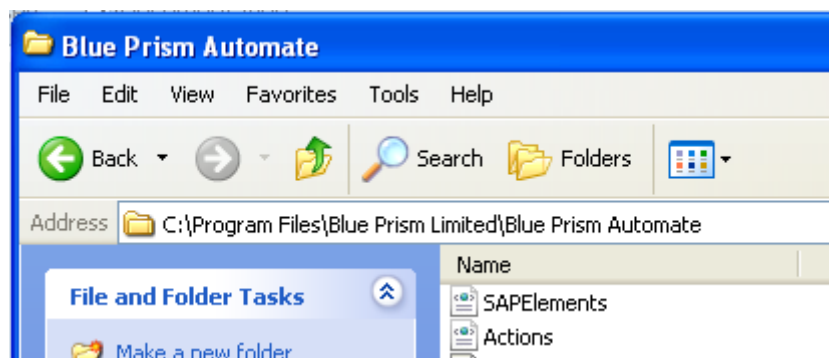
Procedure

In order to extend the functionality of how Blue Prism interacts with an SAP control it is only necessary to amend two important XML files. There are two files that define the Elements that can be worked with (i.e. the SAP controls and their component parts) and the Actions that can be performed with those controls.

The XML files are:-

- SAPElements.XML
- Actions.XML

Both of these files can be found in the default Blue Prism installation directory, e.g.



The two XML configuration files are read by the Blue Prism application when the Interactive Client is started. Once the changes have been added to these files then they will be available in the Application Modeller interface.

Editing the SAPElements file

The SAPElements XML file contains a list of SAP controls, and provides Blue Prism with information about how to interact with these controls.

A typical SAPElements file will contain a description of a number of controls. Here is the XML description of the SAP Treeview control:-

```
<element id="SAPCtrlTree">
  <apptype>SAP</apptype>
  <name>SAP Treeview</name>
  <helptext>A SAP Treeview</helptext>
  <sapidentification>GuiCtrlTree,GuiShell/Tree</sapidentification>
  <readquery action="GetSelectedNodeKey">sapgetproperty
propname=selectedNode</readquery>
  <readquery action="GetNodeText">sapinvokemethod
methodname=getNodeTextByKey arguments=$key$</readquery>
</element>
```

- **id** – a bespoke unique name used by Blue Prism to identify the SAP control
- **apptype** - will always be "SAP"
- **name** – the name for the control, as it will appear in the Application Modeller interface
- **helptext** – some text to describe the type of control
- **sapidentification** - contains a comma-separated list of SAP Component Types (as per the SAP API documentation – see Appendix 3) which should be mapped on to this element type.
 - a forward slash character '/' can be used to specify a subtype
 - an asterisk '*' means that any control that hasn't been matched elsewhere will match this one

In the above example, two *readquery* elements mean that there will be two actions available in a Read Stage for this element type. The action functions (e.g. "GetNodeText") may be ones already in use in the product, since most activities (e.g. pressing a button) are already defined and used. All currently available actions are listed in the [Actions.xml](#) file.

If a control requires a new action, it can be added to the [Actions.xml](#) file (see 'Editing the Actions file' below).

As well as **Read Stage** actions, we can define actions for **Write** and **Navigate** Stages. A Navigate stage will use an *actionquery* and a Write stage will use a *writequery* XML element.

*NOTE: Aside from **writequery**, of which there can be only one, there can be multiple instances of **actionquery** or **readquery** elements.*

This example using a Radio Button control illustrates the *actionquery* and *writequery* elements:-

```
<element id="SAPRadioButton">
  <apptype>SAP</apptype>
  <name>SAP Radio Button</name>
  <helptext>A SAP radio button.</helptext>
  <sapidentification>GuiRadioButton</sapidentification>
  <actionquery action="Select">sapinvokemethod
methodname=select</actionquery>
```

```

    <readquery action="GetWindowText">sapgetproperty
    propname=text</readquery>
    <writequery>sapsetproperty propname=selected
    arguments=$NewText$</writequery>
</element>

```

The default data type for all elements is **text**. If this needs to be changed, it can be specified using a 'datatype' XML element. See '[List of Argument Data Types](#)' for more details.

Editing the Actions file

The **Actions.xml** file contains a list of the programmed actions which can be used with the various elements listed in the **SAPElements.xml** file.

The Actions.xml file is located in the Blue Prism installation folder, typically:

```
C:\Program Files\Blue Prism Limited\Blue Prism Automate\Actions.xml
```

The actions specified in this file are generic actions which can be implemented against any elements which can theoretically use those actions. The scope of such actions is defined in the **SAP GUI Scripting API document** released by SAP (see Appendix 3: Useful references).

An **action** is defined by the following essential parameters:-

- **id** – a unique identifier name for the action
- **name** – the name of the action (which Blue Prism will display)
- **helptext** – helpful text describing the action's function
- **returntype** – the data type returned by the action

The action may optionally require an 'argument' – a parameter which can be passed into the function.

An **argument** can be defined by the following parameters:-

- **id** – the unique identifier for the argument as it will appear in the function's parameter list
- **name** – the name of the argument
- **description** – a description of what the argument's parameter represents
- **datatype** – the data type which determines the data being passed via the argument

Here is an example of the 'DoubleClickNode' action's definition:

```

<action id="DoubleClickNode">
  <name>Double Click Node</name>
  <helptext>Simulate double clicking the tree node</helptext>
  <returntype>text</returntype>
  <argument id="newtext">
    <name>Text</name>
    <description>The text of the node</description>
    <datatype>text</datatype>
  </argument>
</action>

```

This action equates to the published SAP function 'doubleClickNode()':

Function doubleClickNode (nodeKey As String)

This function emulates double clicking a node.

Note: If Item Selection is enabled, double clicking a node can only be performed by double clicking on the Folder/Leaf Symbol of the node.

Functions for working with SAP controls

The following functions are available to work with SAP controls:-

- **sapgetproperty** – for obtaining property data from an SAP element
- **sapsetproperty** – for defining property data for an SAP element
- **sapinvokemethod** – for calling an existing SAP function

SAPGetProperty

This function returns the properties which an SAP control may expose to be consumed. This function is typically called as part of a <readquery> request.

The required parameters of this function are:-

- **id** – the unique identifier of the SAP component
- **propname** – the name of the property

A sapgetproperty() call which specifies these parameters will return a **string** return value.

Usage example:

```
<readquery action="GetWindowText">sapgetproperty  
propname=text</readquery>
```

This action will return a string value containing the text of a Window title.

The following optional properties may be used:-

- **retproc** – specifies additional processing on the returned value
- **colcount** – if specified then it is assumed that the target SAP component is a GuiCollection, and the count of the component's columns is returned as a **number**
- **targetprop** - specifies the name of a property of the identified component on which the action will actually be performed
- **arguments** – any arguments required to access the property, as a comma-separated list. This must be the correct number of arguments required by the property's definition.

SAPSetProperty

This function sets the properties which an SAP control may require. This function is typically called as part of the <writequery> request.

The required parameters of this function are:-

- **id** – the unique identifier of the SAP component
- **propname** – the name of the property
- **arguments** – any arguments required to access the property, as a comma-separated list. This includes the value(s) being set.

Usage example:

```
<writequery>sapsetProperty propName=text  
arguments="$newtext$"</writequery>
```

This action will set a property called "text" to be the value of the string contained in the "newtext" variable which is specified at runtime.

The following optional properties may be used:-

- **targetprop** - specifies the name of a property of the identified component on which the action will actually be performed

SAPInvokeMethod

This function calls an existing SAP method which an SAP control may expose to be consumed. This function is typically called as part of an <actionquery> request.

The required parameters of this function are:-

- **id** – the unique identifier of the SAP component
- **methodname** – the name of the method

A sapinvokeMethod() call which specifies these parameters will not return any return value unless the colcount property is added.

Usage example:

```
<actionquery action="Press">sapinvokeMethod  
methodname=press</actionquery>
```

This action will invoke the control's 'press' method, thereby pressing the control (e.g. a button). This will be exposed within Blue Prism as an action called "Press".

The following optional properties may be used:-

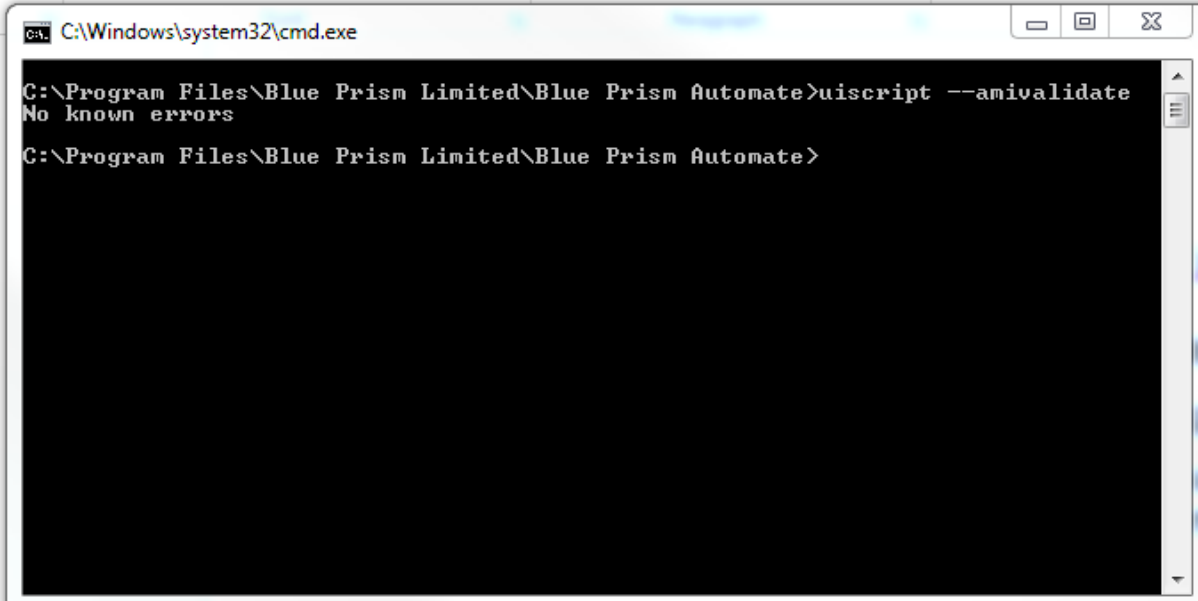
- **retproc** – specifies additional processing on the returned value
- **colcount** – if specified then it is assumed that the target SAP component is a GuiCollection, and the count of the component's columns is returned as a **number**
- **targetprop** - specifies the name of a property of the identified component on which the action will actually be performed
- **arguments** – any arguments required to invoke the method, as a comma-separated list. This must be the correct number of arguments.

Testing new functions

Use the **UIScript** tool with an **-AMValidate** switch inside a command prompt in order to validate and test the new functions and actions. The UIScript tool is available from your Blue Prism installation folder.

```
UIScript.exe --amivalidate
```

For example:

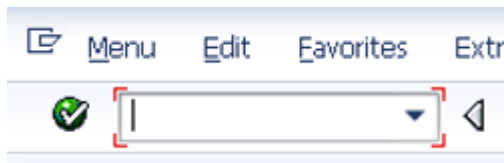


```
C:\Windows\system32\cmd.exe
C:\Program Files\Blue Prism Limited\Blue Prism Automate>uiscript --amivalidate
No known errors
C:\Program Files\Blue Prism Limited\Blue Prism Automate>
```

If you cannot resolve any issues with these functions then please seek the assistance of the Blue Prism Customer Support team by emailing support@blueprism.com.

Example of adding a new action to an existing control

Here is an example of adding the action **Set Focus** to the **SAPOKCode** field. The SAPOKCode field is the Transaction Code text field used in the top left-hand corner of the SAP main screen:



This field is used to enter an SAP transaction code which SAP will then run.

The action 'Set Focus' already exists as a definition within the Actions.XML file:

```
<action id="SetFocus">
  <name>Set Focus</name>
  <helptext>Places the focus for events on to a specific control.</helptext>
</action>
```

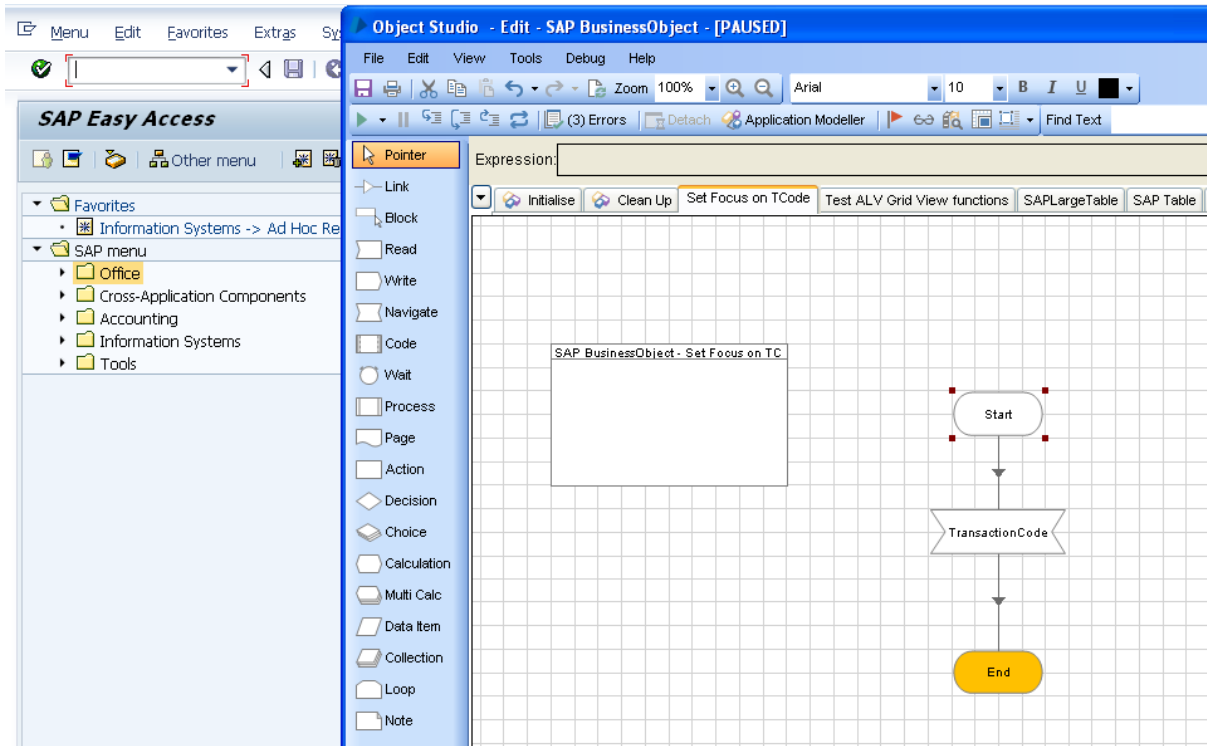
To implement Set Focus against the SAPOKCode field, the Set Focus action needs to be added to the Control's definition in the SAPElements.XML file:

```
<element id="SAPOkCodeField">
  <apptype>SAP</apptype>
  <name>SAP OK Code</name>
  <helptext>A SAP OK Code Field.</helptext>
  <sapidentification>GuiOkCodeField</sapidentification>
  <readquery action="GetWindowText">sapgetproperty propName=text</readquery>
  <writequery>sapsetproperty propName=text arguments="$newtext$"</writequery>
  <actionquery action="SetFocus">sapinvoke method methodname=SetFocus</actionquery>
</element>
```

In this instance the actions is an "actionquery" type of action, and the definition was simply copied from another control which had already got this method implemented.

Once the files have been saved, then Blue Prism can be launched. When launched Blue Prism will read the Actions and SAPElements XML files to establish the functions and methods available.

The new action can be tested with an appropriate Stage (e.g. a Navigate stage uses 'Set Focus'):



NOTE: The SAP transaction Code field now has the focus after the Navigate stage sets it.

Appendix 1: List of Argument Data Types

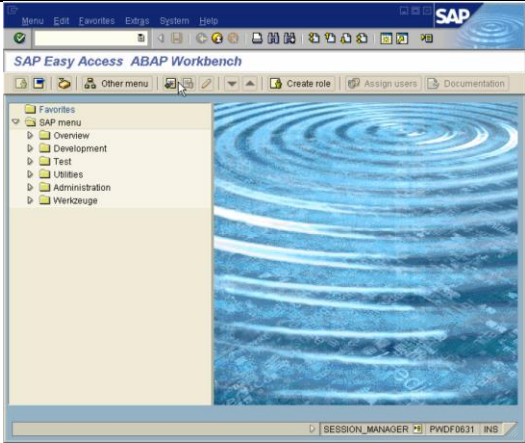
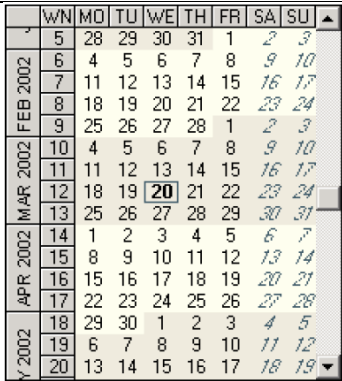
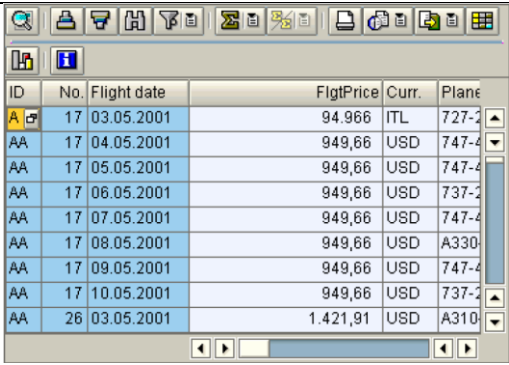
When passing arguments (parameters, or variables) to a function it is necessary to specify what data type those arguments represent. Each argument passed must be enclosed in the symbol which defines their data type.

The default data type (if unspecified) will be text. Here is the list of argument data type symbols:-

Data type	Symbol	Example
Text	\$	\$text\$
Number	#	#\$number\$
Date	@	@\$date\$

Appendix 2: List of SAP Controls

This list is the currently recognised list of SAP controls. This can be expanded by adding new controls to the SAPElements.xml file.

SAP element name	Blue Prism element ID	Appearance
GuiStatusBar	SAPStatusBar	
GuiTitlebar	SAPTitleBar	
GuiMainWindow	SAPMainWindow	
GuiShell/Calendar	SAPCalendar	
GuiShell/GridView, APOGrid, GuiCtrlGridView	SAPGridView	
GuiPasswordField	SAPPasswordField	
GuiCTextField, GuiTextField	SAPTextBox	
GuiComboBox	SAPComboBox	
GuiOkCodeField	SAPOkCodeField	
GuiButton	SAPButton	

GuiCheckBox	SAPCheckBox	
GuiRadioButton	SAPRadioButton	
GuiTabStrip	SAPTabStrip	
GuiTab	SAPTab	
GuiCtrlTree,GuiShell/ Tree	SAPCtrlTree	
GuiUserArea	SAPUserArea	
GuiSplitterShell	SAPSplitter	
GuiContainerShell	SAPGuiContainerShell	
GuiToolbar	SAPToolbar	
GuiMenubar	SAPMenubar	
GuiLabel	SAPLabel	
GuiTableControl	SAPTable	
GuiScrollbar	SAPScrollbar	
GuiScrollContainer	SAPScrollContainer	
GuiHTMLViewer,Gui Shell/HTMLViewer	SAPHTMLViewer	

NOTE: Any control not recognised as a specific SAP element type will be recognised as a generic “SAP Component” element.

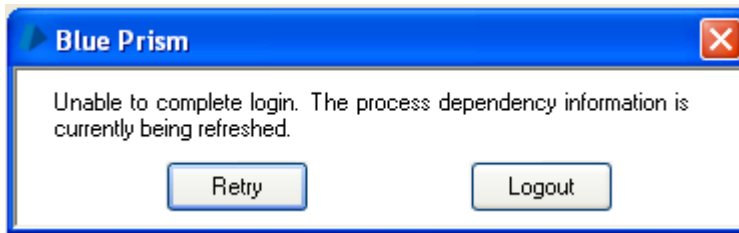
Appendix 3: Useful references

This section contains a list of useful resources that can be used to add more detail to the SAP Controls, their capabilities, and how they are used.

Reference	Description	Hyperlink
SAP GUI Scripting API Reference	The SAP reference guide to scriptable controls and functions.	SAP GUI Scripting API
Keyboard Controls for SAP GUI Elements	A guide to using keystrokes	Keyboard controls for SAP GUI elements.pdf
SAP GUI Scripting API demonstration	A slide presentation in PDF form which explains the background to, and general information about, SAP GUI Scripting.	SAP GUI Scripting API
Enabling SAP Spy Mode via GUI Scripting Settings	A quick guide to the process of enabling SAP GUI scripting in the SAP GUI interface.	Blue Prism Knowledge Base article
SAP GUI Scripting Security Guide		Security Guide

Troubleshooting

“The process dependency information is currently being refreshed”.



When logging into Blue Prism after upgrading your version of the software you get the error “The process dependency information is currently being refreshed”. This does not go away when you Retry and the only option is to log out.

This can occur when there has been a change to the Actions.XML file. Blue Prism is expecting a specific action to be in the XML file, but it cannot be found.

To determine which action is causing the problem run this command within a command prompt from the Blue Prism Automate directory:

```
C:\Program Files\Blue Prism Limited\Blue Prism Automate>automatec /user admin ad  
mini /refreshdependencies force
```

NOTE: Use the correct admin login credentials for this command

This will reveal the problem area:

```
Error processing action: refreshdependencies.  
BluePrism.BPCoreLib.NoSuchElementException: AMI did not recognise the action typ  
e 'SetCurrentCell'  
   at BluePrism.AutomateProcessCore.clsNavigateStep.FromXML(XmlElement e, clsApp  
licationTypeInfo appInfo)  
   at BluePrism.AutomateProcessCore.Stages.clsNavigateStage.FromXML(XmlElement e  
)  
   at BluePrism.AutomateProcessCore.clsProcess.ParseXMLInternal(ProcessXmlDocume  
nt doc, Boolean fullProcess)  
   at BluePrism.AutomateProcessCore.clsProcess.FromXML(clsGroupObjectDetails ext  
ernalObjs, String xml, Boolean editable, Boolean fullProc)  
   at BluePrism.AutomateAppCore.clsServer.RebuildDependencies(Boolean force)  
   at AutomateC.AutomateC.Run(String[] args)
```