New accessibility model for Microsoft windows

R Haverty

Accessible Technology Group, Microsoft Corporation
1 Microsoft Way, Redmond, Washington 98052, USA

www.microsoft.com/enable

ABSTRACT

Microsoft® Windows® User Interface (UI) Automation is the new accessibility framework for Microsoft Windows and is intended to address the needs of assistive technology products and automated testing frameworks by providing programmatic access to information about the user interface. UI Automation will be fully supported in the Windows platform on "Longhorn" and will be the means of enabling automated testing and accessibility for all new forms of Windows user interface, including existing legacy controls.

1. INTRODUCTION

In 2003, Microsoft Corporation commissioned Forrester Research, Inc., to conduct a study to measure the potential market of people in the United States who are most likely to benefit from the use of accessible technology for computers http://www.microsoft.com/enable/research/. Accessible technology enables individuals to adjust their computers to meet their visual, hearing, dexterity, cognitive, and speech needs. It includes both accessibility options built into products as well as specialty hardware and software products (assistive technology products) that help individuals interact with a computer. Overall results show that 57% (74.2 million)[i] of computer users in the United States are likely or very likely to benefit from the use of accessible technology due to experiencing mild to severe difficulties or impairments http://www.microsoft.com/enable/research/computerusers.aspx.

With the current technology, assistive technology vendors (ATV) are required to use many different approaches to obtain and present information about the UI to the end user, thus spending an inordinate amount of time and resources on providing this basic information. With such a large percentage of users needing accessible information it is becoming increasingly important to make it easier for an ATV to programmatically obtain information about the UI.

The new accessibility model for Windows, UI Automation, is designed to provide a single reliable source of UI information to assistive technology products and automated test scripts. It provides programmatic access that allows automated tests to interact with the UI and allows assistive technology products to provide information about the user interface to their end users. UI Automation also provides means for manipulating the UI.

UI Automation has two main audiences: UI Automation providers and UI Automation clients. UI Automation providers are applications such as Microsoft Word, Excel, or third-party applications based on the Windows operating system. UI Automation clients are assistive technology applications, such as screen readers, screen enlargers, alternative input, or others. Automated test scripts can use UI Automation for automated testing and are also considered clients in the UI Automation framework.

This document includes information on the namespaces that the UI Automation framework uses, as well as information on the following UI Automation features: automation tree, UI Automation control patterns, UI Automation properties, UI Automation events, and UI Automation Input. Also included is information on security. This documentation for UI Automation is preliminary and is subject to change.

2. UI AUTOMATION NAMESPACES

The following table lists the namespaces used in the UI Automation framework, as well as the audience that uses each namespace.

Table 1. *UI Automation Namespaces and related audiences.*

Namespace	Audience used by
System.Windows.Automation	UI Automation clients (both assistive technology and automated test scripts) for finding automation elements, registering for events and working with control patterns.
System.Windows.Automation.Provider	UI Automation providers for implementing UI Automation on "Avalon" controls or applications.
System.Windows.Automation.InteropProvider	UI Automation providers for implementing UI Automation on non-"Avalon" frameworks such as Microsoft Win32®.
System.Windows.Automation.ComInteropProvider	UI Automation providers for implementing UI Automation on COM-based controls or applications.
System.Windows.Automation.Searcher	UI Automation clients for navigating the automation tree.

3. UI AUTOMATION TREE

Standard Windows programming has always exposed the relationship between elements in the user interface in a parent/child relational structure. UI Automation clients (assistive technologies and automated testing tools) view the UI elements on the desktop as a set of automation elements which are arranged in a tree structure. Automation elements implement a common class (AutomationElement) to enable consistent information, interaction, and a navigation model. UI Automation unifies disparate UI Frameworks such as Avalon, Trident, and Win32 so that code can be written against one API rather than several.

Within the automation tree there is a root automation element which represents the current desktop and whose children represent application windows on the desktop. Each of these childe elements can contain automation elements representing the UI elements comprising their UI, such as menus, buttons, toolbars, and list boxes. Each piece of UI can contain automation elements representing their content, such as menu items, or list items. Even a button, which does not contain any items, may have child automation elements which represent the basic UI components that comprise the button, such as text and rectangles.

It is important to note that the automation tree is not a fixed structure. For performance reasons it is built on demand starting with an automation element which the UI Automation client specifies.

1.1. Views of the Automation Tree

The automation tree can be filtered to create customized views of the tree which contains only those automation elements that are relevant for a particular client. This approach allows clients to customize the structure presented through UI Automation to their particular needs. A few default views are provided by the UI Automation framework, but clients can also define custom views if they need additional control.

1.2. Raw View

The raw view of the automation tree is the full tree of elements for which the desktop is the root. The raw view closely follows the native programmatic structure of an application and therefore is the most detailed view that is available. It is also the base on which the other views of the tree are built. Because this view depends on the underlying UI framework, the raw view of an Avalon button will have a different raw view than a Win32 button.

1.3. Control View

The Control View of the automation tree simplifies the assistive technology product's task of describing the UI to the end user and helping that end user interact with the application because it closely maps to the UI structure perceived by an end user.

The control view includes all elements from the raw view that an end user would understand as interactive or contributing to the logical structure of the control in the UI. Examples of elements that contribute to the logical structure of the UI but are not interactive themselves are item containers such as list view headers, toolbars, menus, and the status bar. Non-interactive elements used simply for layout or decorative purposes will not be seen in the control view. An example is a panel that was used only to layout

Proc. 5th Intl Conf. Disability, Virtual Reality & Assoc. Tech., Oxford, UK, 2004

the controls in a dialog but does not itself contain any information. Non-interactive elements that will be seen in the control view are graphics with information and static text in a dialog.

1.4. Content View

The content view of the automation tree contains elements that convey the true information in a user interface. For example, the values in a drop-down ComboBox will appear in the content view because they represent the information being used by an end user. In the content view, a ComboBox and ListBox are both represented as a collection of items where one, or perhaps more than one, item can be selected. The fact that one is always open and one can expand and collapse is irrelevant in the content view because it is designed to show the data, or content, that is being presented to the user.

1.5. Custom Views

The UI Automation framework also allows a UI Automation client to create a custom view of the automation tree by specifying the desired match conditions and scoping information. This also allows UI Automation clients to build their own interaction models for the application using just the data that they need.

1.6. Automation Tree Structure Example

The following example compares the control view and content view of the automation tree for the same application: Wordpad.

Table 2. *Comparison of two Automation Tree structures*

Automation Tree (Control View)	**Automation Tree (Content View)**

The *Control* view of WordPad shown from the Desktop has the following automation tree structure:
• Desktop
  ○ Window “Notepad”
  ▪ TitleBar “Notepad”
  • SystemBar
  ○ MenuItem
  • Button AutomationId = “Minimize”
  • Button AutomationId = “Maximize”
  • Button AutomationId = “Close”
  ▪ MenuBar ““
  • MenuItem “File”
  • MenuItem “Edit”
  ▪ ToolBar ““
  • Button “New”
  • Button “Open”
  ▪ Text ““
  ▪ StatusBar
  • Edit
  • Edit
	The *Content* view of WordPad shown from the Desktop has the following automation tree structure:
	• Desktop
	  ○ Window “Notepad”
	  ▪ MenuBar ““
	  • MenuItem “File”
	  • MenuItem “Edit”
	  ▪ ToolBar ““
	  • Button “New”
	  • Button “Open”
	  ▪ Text ““
	  ▪ StatusBar
	  • Edit
	  • Edit

4. UI AUTOMATION CONTROL PATTERNS

UI Automation uses control patterns to express the functionality contained in each control. UI Automation differentiates between what a user would call the control and what can be programmatically done with the control by using control patterns to express only functionality, separate from the type or name of that control.

UI Automation providers implement control pattern interfaces on UI elements. For Avalon controls, control pattern interfaces are found in the System.Windows.Automation.Provider namespace and have names that include the suffix “Provider” (for example, IScrollProvider and IInvokeProvider). For non-Avalon controls, control pattern interfaces are found in the System.Windows.Automation.InteropProvider namespace and have names that include the suffix “InteropProvider” (for example, IScrollInteropProvider and IInvokeInteropProvider).

Clients access methods and properties of control pattern classes and use them to access information about a UI element, or to manipulate the UI. These control patterns classes are found in the

Systems.Windows.Automation namespace and have names that include the suffix "Pattern" (for example, InvokePattern and SelectionPattern).

1.7. Control Pattern Components

Control patterns define the structure, methods, properties, and events supported by a control:

- The structure includes the parent, child, and sibling relationships of elements for that control pattern.
- The methods provide the ability to programmatically manipulate the control.
- The properties and events provide rich information and notifications relevant for that control.

Control patterns relate to UI as interfaces relate to COM objects. In COM, you can query an object to ask what interfaces it supports, and then use those interfaces to access functionality. In UI Automation, clients can ask a control which patterns it supports and then interact with the control through the properties, methods, events, and structure of the supported control patterns. For example, providers implement IScrollProvider for a multi-line edit box. When a client detects that a UI element supports ScrollPattern, it can use the properties, methods, and events from that class to gather scroll-specific information or programmatically scroll its content to a new position.

1.8. Standard UI Controls and Their Control Patterns

Controls can support zero or more control patterns. For example:

- The image control does not support any control patterns.
- The button control supports InvokePattern to correspond to the functionality that it can be clicked.

To define the full set of functionality for a control, providers implement multiple control patterns. The following table shows more examples of standard controls and the control patterns they support.

Table 3. *Controls and their Control Patterns.*

Control Type	Relevant Control Patterns
Button	Invoke
CheckBox	Toggle
ComboBox	ExpandCollapse or Selection
Edit	Value, Text
ListBox	Selection
ListItem	SelectionItem
Tree	Selection
TreeItem	SelectionItem, ExpandCollapse

1.9 Control Patterns

Table 4 lists some of the key control patterns and their classes and interfaces. Note that interfaces are designated for "Avalon" providers (for example, IInvokeProvider) and non-"Avalon" providers (for example, IInvokeInteropProvider).

5. UI AUTOMATION PROPERTIES

UI Automation properties are a set of standard properties that expose information that is important to assistive technologies. Frequently, this information is exposed differently for each UI framework. Table 5 shows how one standard UI Automation property maps to multiple property names in other UI frameworks. By implementing UI Automation, providers map unique UI framework properties to standard UI Automation properties. When this done, it allows UI Automation clients to query for property information using one API call for a UI Automation property.

Table 4. *Control Patterns and their Classes and Interfaces*

Control Pattern	Client-Side Class	Provider-Side Interfaces
ExpandCollapse	ExpandCollapsePattern	IExpandCollapseProvider; IExpandCollapseInteropProvider
Grid	GridPattern	IGridProvider; IGridInteropProvider
GridItem	GridItemPattern	IGridItemProvider; IGridItemInteropProvider
Invoke	InvokePattern	IInvokeProvider; IInvokeInteropProvider
MultipleView	MultipleViewPattern	IMultipleViewProvider; IMultipleViewInteropProvider
RangeValue	RangeValuePattern	IRangeValueProvider; IRangeValueInteropProvider
Scroll	ScrollPattern	IScrollProvider; IScrollInteropProvider
Selection	SelectionPattern	ISelectionProvider; ISelectionInteropProvider
SelectionItem	SelectionItemPattern	ISelectionItemProvider; ISelectionItemInteropProvider
Sort	SortPattern	ISortProvider; ISortInteropProvider
Table	TablePattern	ITableProvider; ITableInteropProvider
TableItem	TableItemPattern	ITableItemProvider; ITableItemInteropProvider
Text	TextPattern	ITextProvider; ITextInteropProvider
Toggle	TogglePattern	IToggleProvider; IToggleInteropProvider
Transform	TransformPattern	ITransformProvider; ITransformInteropProvider
Value	ValuePattern	IValueProvider; IValueInteropProvider
Window	WindowPattern	IWindowProvider; IWindowInteropProvider
Zoom	ZoomPattern	IZoomProvider; IZoomInteropProvider

Table 5. *Mapping UI Automation Properties to Other UI Frameworks.*

UI Automation Control Type	UI Framework	Framework Property	UI Automation Property
Button	Avalon	Content	NameProperty
Button	Win32	Caption	NameProperty
Image	Trident/HTML	ALT	NameProperty

6. UI AUTOMATION EVENTS

UI Automation offers an event mechanism similar to WinEvents in the current Windows platform. However, unlike WinEvents, UI Automation's events are not based on a broadcast mechanism. UI Automation clients register for specific event notifications and can request that specific UI Automation properties and control pattern information be passed into their event handler. This provides a much more powerful mechanism than WinEvents because clients make fewer calls to retrieve the information they require, which results in a fewer cross-process calls, and therefore better performance. UI Automation provides event notifications for logical structure changes, control pattern changes, focus changes, property changes, and multimedia events.

7. UI AUTOMATION INPUT

In addition to using control patterns to manipulate the UI, UI Automation provides an Input class as a way to automate keyboard or mouse input. This class is a simple wrapper around SendInput that may, in a future release, be replaced by a more powerful input model (designed in conjunction with IME, Tablet, and Cicero teams).

8. SECURITY AND UI AUTOMATION

The security model for UI Automation is based on only granting access that is needed at the current time. Programmatically accessing and manipulating the user interface (UI) - functionality available in many UI Automation programming elements - requires specific security permissions. These UI Automation programming elements do not work inside a default secure execution environment. This applies to the following groups of methods:

- Methods that access information about a UI element, such as property values.
- Methods that write or modify the UI by using Control Patterns, such as AddToSelection.
- Methods that provide keyboard or mouse input. These methods are in the Input class.

UI Automation clients need to have the appropriate AutomationPermission when using these programming elements. The following table summarizes the AutomationPermission that UI Automation clients can use and provides examples of methods that would require the permission:

Table 5. *AutomationPermssion for UI Automation Clients*

Action of Method	AutomationPermissionFlag	Example
Access information about the UI	Read	GetPropertyValue
Write to or modify the UI	Write	Control pattern methods that manipulate the UI, such as AddToSelection or Invoke.
Send mouse or keyboard input	Input	Methods from the Input class, such as MoveToAndClick or SendKeyboardInput.

9. CONCLUSIONS

UI Automation is a key part of the new accessibility model for Windows, gathering information about and interacting with the UI. Adoption of this technology will improve product quality for Windows applications and reduce the time to market for assistive technology products. Additionally, by implementing UI Automation, ATVs reduce the resources invested in obtaining UI information allowing them to improve and expand on the products that they offer.

10. REFERENCES

UI Automation specific section of the Longhorn Software Development Kit (SDK):

http://longhorn.msdn.microsoft.com/?//longhorn.msdn.microsoft.com/lhsdk/accessibility/overviews/uiaccess_ovw.aspx,

Longhorn SDK Home page: http://longhorn.msdn.microsoft.com/

Longhorn Developer Center home page: http://msdn.microsoft.com/longhorn/

Microsoft research results, "The Wide Range of Abilities and Its Impact On Computer Technology":

http://www.microsoft.com/enable/research/

Microsoft Accessibility home page: http://www.microsoft.com/enable/

[i] Study Commissioned by Microsoft Conducted by Forrester Research, Inc. 2004.