Chapter 1

The UI Automation Environment

Intended for interoperable implementations by other companies, Microsoft's UI Automation (UIA) Community Promise is a specification that provides information about Microsoft's accessibility frameworks, including Active Accessibility (MSAA), UI Automation (UIA), and its shared implementations. In this chapter, we provide a summary of descriptions from the UIA Community Promise to show how the components of UIA fit together to enable accessibility.

UIA provides programmatic access to UI controls on the desktop, enabling assistive technology (AT) products, such as screen readers, to provide information about the UI to end users. ATs enable the user to manipulate the UI by means other than the standard mouse and keyboard, such as through speech recognition.

UIA improves upon Microsoft's legacy accessibility framework, MSAA, by aiming to address the following goals:

- Enable efficient access and security over MSAA's architecture
- Expose more robust information about the UI
- Offer interoperability with MSAA implementations
- Provide developers the option of using either native interfaces or managed interfaces

For demonstration purposes, examples are in native code (unmanaged interfaces based on COM); however, the same principles and techniques are applied to managed practices (the programming model of the Microsoft .NET Framework). Whether you will use native or managed code depends upon your framework and preferences.

Go further: For more information on the UIA Community Promise, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Providers and Clients

In UIA, applications, such as word processing programs, are called *Providers*. ATs, such as screen readers, are called *Clients*. Providers expose properties and features of the UI by implementing UIA interfaces. Clients can then obtain information about the UI through a client interface from the UIA framework.

Providers communicate to Clients through UIA Events. Events are crucial for notifying Clients of changes to the UIA Tree (discussed later in this chapter), UI states, or UI controls. Unlike

WinEvents used in MSAA, UIA Events use a subscription mechanism, rather than a broadcast mechanism, to obtain information. UIA Clients register for UIA Events for specific user interfaces or even parts of the UI and can also request that some UIA Properties and Control Pattern information be cached along with registration for better performance.

Figure 1-1 is a simplified illustration of a UIA Provider and Client.

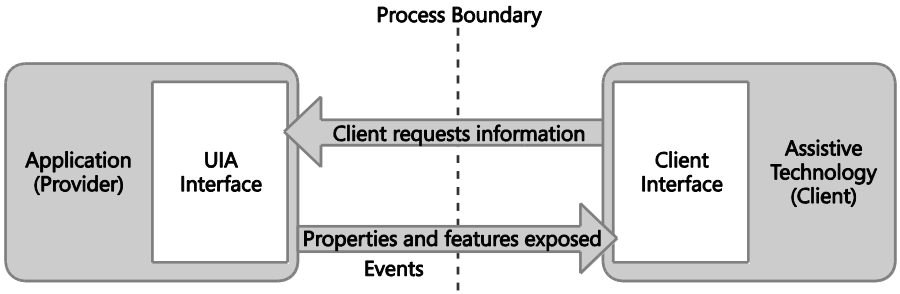

FIGURE 1-1 Simplified illustration of a UIA Provider and Client

Providers

An application may support UIA through one of two ways:

- Designing the UI based on standard framework controls and libraries that support UIA
- Implementing the UIA Provider interfaces

The following are just some of the common actions performed by UIA Providers:

- Expose UI controls by describing their functionality through Control Patterns, Properties, and Methods
- Expose the relationships of UIA Elements through the UIA Tree
- Report changes and actions related to the UI by raising UIA Events

Clients

UIA Clients can perform many different actions. The following are just some of the common actions performed:

- Search for elements within the UIA Tree
- Navigate among UIA Elements

- Subscribe to UIA Events
- Manipulate the UI by using UIA Control Patterns

Main Components

Now that you have a general sense of how UIA works, let's talk further about the main components of the framework: the Automation Elements and the UIA Tree.

Automation Elements

UIA exposes every component of the UI to Client applications as an Automation Element. Elements are contained in a tree structure, with the desktop as the root element.

Automation Elements are associated with pairs of Properties and Control Patterns, representing the functionality of an element in the UI. One of these properties is the UIA Control Type, which defines its basic appearance and functionality as a single recognizable entity, such as a button or check box. Table 1-1 lists a few Control Types and Patterns associated with a typical Automation Element.

TABLE 1-1 Example set of Control Types and Patterns associated with a typical Automation Element

Name	Control Type	Control Pattern
OK	`Button`	`Invoke`
Open	`ComboBox`	`Value, Expand/Collapse`
Installed Programs	`List`	`Selection, Scroll`

The UIA Tree

The UIA Tree allows UIA Clients to navigate through the structure of the UI. The root element of the Tree is the desktop, whose child elements are programs running on it, such as an application or the operating system's UI. Each of the child elements can contain elements representing parts of the UI, such as menus, buttons, toolbars, and lists. These elements in turn can also contain sub-elements, such as items in a list.

The UIA Tree is not a fixed structure and is seldom seen in its totality, because it might contain thousands of elements. Parts of it are built as they are needed, and it can undergo changes as elements are added, moved, or removed. UIA enables reparenting and repositioning, so that an element can move to another part of the tree, despite the hierarchy imposed by ownership of the underlying architecture.

Navigation in the UIA Tree is hierarchical: from parents to children and from one sibling to the next. UIA Providers support the UIA Tree by implementing navigation among items within a fragment, which consists of its root and sub-elements. Simple parts of the UI, however, do not need navigation implemented. The UIA framework manages navigations between fragments based on the underlying architecture.

A simple UIA Provider can be seen in Figure 1-2. Created on a Win32 framework, the Email Address window contains two child elements: the Email text label and its corresponding edit box. The Email text label and the edit box are siblings and would be positioned next to each other in the fragment of the UIA Tree. In Chapter 2, "Designing the Logical Hierarchy," we discuss in more detail why correctly mapping sibling relationships is important for navigation and giving users of AT context about the UI.

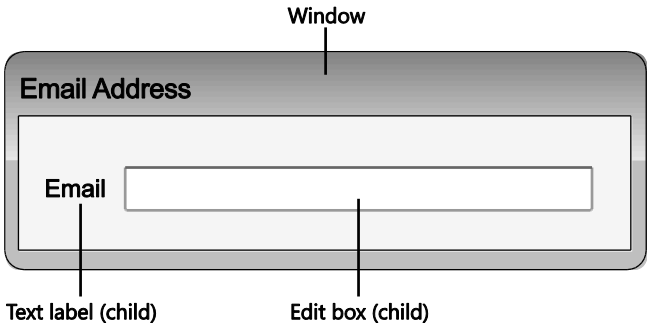

FIGURE 1-2 UIA Provider with two child elements: the Email text label and its corresponding edit box

UIA offers three default views of the UIA Tree for Clients. Clients can customize the view by defining new conditions for the UIA Properties.

- **Raw view** The raw view is a UIA Tree with no filtering. All elements are available in this view.
- **Control view** The control view of the UIA Tree simplifies the AT product's task of describing the UI to the end user and helping that end user interact with the application. The view maps to the UI structure perceived by an end user. It includes all Automation Elements that an end user would understand as interactive or contributing to the logical structure of the control in the UI. Examples of UI items that contribute to the logical structure of the UI, but are not interactive themselves, are list view headers, toolbars, menus, and the status bar. Non-interactive items used simply for layout or decorative purposes will not appear in the control view. An example would be a panel that is used only to lay out the controls in a dialog box, decorative graphics, and static text in a dialog box. UIA Providers can specify the elements appearing in control view by setting the UIA `IsControlElement` Property to True.

- **Content view** The content view of the UIA Tree is a subset of the control view. It contains UI items that convey the true information in a UI, including UI items that can receive keyboard focus and some text that are not labels for other UI items nearby. For example, the values in a drop-down combo box will appear in the content view because they represent the information being used by an end user. UIA Providers can specify the elements appearing in content view by setting the UIA `IsContentElement` Property to True.

Control Patterns

Control patterns represent common UI behaviors (such as invoking a button) and support the properties, methods, and events. Each UIA Control Pattern is its own interface with properties and methods that provide a way to categorize and expose a control's functionality, independent of the UIA Control Type or the appearance of the control. Table 1-2 provides examples of the functionality represented by different UIA Control Patterns.

TABLE 1-2 Examples of functionality for different Control Patterns

Functionality	Control Pattern
Ability to share three states of on / off / indeterminate	`Toggle`
Ability to support a numeric value within a range	`RangeValue`
Ability to support a string value	`Value`
Ability to move / resize / rotate	`Transform`

Go further: For more information on UIA Control Patterns, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Control Types

UIA Control Types are well-known identifiers that can be used to indicate what kind of con trol a particular element represents, such as a Button, Check Box, Combo Box, Data Grid, Document, Hyperlink, Image, ToolTip, Tree, or Window. Each Control Type has a set of conditions, which include specific guidelines for the UIA Tree, Property values, Control Patterns, and Events that a control *must* meet to use a Control Type defined in the UIA Specification.

Having a well-known identifier makes it easier for Client programs to determine what kinds of controls they must interact with in the UI. The Control Types included with UIA offer a clearer identification for the controls than ones defined by MSAA's `accRole` property.

Controls do *not* have to set a Control Type, however. If there is no Control Type that represents your control well, set the Control Type to "custom," and expose your control properly through the patterns and properties (including the `LocalizedControlType` property) that makes the most sense for your control. The UIA Specification defines required, recommended, or prohibited control patterns and properties. Custom controls can implement additional Control Patterns or Properties while being mapped to a specific Control Type.

Go further: For more information on UIA Control Types, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Properties

In UIA, there are two kinds of properties that provide information about a UI element:

- **Automation Element Properties** Properties that are applicable to most elements. For example, two properties that apply to all Automation Elements are the `Name` and `AutomationId` properties. Having these properties properly filled is highly recommended because most Clients use these properties for every Automation Element, but there may be times when the `Name` property may be blank for valid reasons. For example, elements used solely for layout purposes are often kept nameless, but interactive controls should not be left with a blank `Name` property.
- **Control Pattern Properties** Properties specific to the functionality represented in the Control Pattern interfaces. For instance, the UIA Value Pattern will support the Value property to represent the context of controls such as a progress bar or calendar.

To ensure that you are providing the right information for clients to consume, be sure to adhere to the Specification. Certain properties have very strict requirements set. At other times, sometimes leaving the default property values is the right course of action.

Go further: For more information on UIA Properties, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Events

UIA Events correspond to activities occurring in the UI and are crucial pieces of information for UIA Clients. As mentioned, UIA uses a subscription model for UIA Events; a UIA Provider will not process an Event unless a Client is listening for them. Table 1-3 lists the four different types of UIA Events.

TABLE 1-3 UIA Events

Event	Description
Property change	Raised when a UIA property changes. For example, if a Client needs to monitor an application's check box control, it can register to listen for a Property change Event on the `ToggleState` property of the Toggle Pattern. When the check box control is checked or unchecked, the property change Event for the Property gets raised.
Element action	Raised when an action is made in the UI, often related to UIA Control Patterns. For example, when an item is selected, an `ElementSelected` Event gets raised.
Structure change	Raised when the structure of the UIA Tree changes. The structure changes when new UI items become visible, hidden, or removed on the desktop.
General event	Raised when actions of global interest to the Client occur, such as when the focus shifts from one element to another, or when a window closes.

Go further: For more information on UI Automation Events, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Custom Control Patterns, Properties, and Events

UIA features several Control Patterns, Properties, and Events, but the Windows implementation of UIA also offers further extensibility by registration of custom control patterns, properties, and events. As of today, this functionality is not available for managed applications of both UIA Providers and Clients.

New custom control patterns, properties, and events are only necessary if the standard UIA Control Patterns, Properties, and Events are not sufficient. Because of the extraordinary costs associated with creating new custom control patterns, properties, and events, you should avoid doing so whenever possible.

Go further: For more information on UIA Custom Control Patterns, Properties, and Events and future interoperable specifications, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Planning Your Hierarchy

Now that we have covered how each of the components of UIA fit together and enable programmatic access, you are ready to learn how to design a navigational tree, called the *logical hierarchy,* for your product. In the next chapter, we walk you through the steps for designing a logical hierarchy, using an employee timecard application as an example.