

---

---

**Information technology — Interoperability  
with Assistive Technology (AT) —**

Part 2:

**Windows accessibility application  
programming interface (API)**

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

*Technologies de l'information — Interopérabilité avec les technologies  
d'assistance —*

*Partie 2: Interface de programmation d'applications (API) d'accessibilité  
Windows*

[ISO/IEC TR 13066-2:2012](https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012)

[https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-  
d6c7e58e3122/iso-iec-tr-13066-2-2012](https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012)

## iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC TR 13066-2:2012](https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012)

<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>



### **COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.org](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Published in Switzerland

# Contents

Page

|   |           |
|---|-----------|
| Foreword .....  | iv        |
| Introduction.....   | v         |
| <b>1 Scope .....</b>  | <b>1</b>  |
| <b>2 Terms and definitions .....</b>  | <b>1</b>  |
| <b>3 General Description and Architecture of the Microsoft Windows Automation API .....</b>                         | <b>7</b>  |
| <b>3.1 General Description .....</b>  | <b>7</b>  |
| <b>3.2 Architecture .....</b>   | <b>10</b> |
| <b>4 Using the API .....</b>  | <b>12</b> |
| <b>4.1 Using the Microsoft Active Accessibility API.....</b>  | <b>12</b> |
| <b>4.2 Using the UI Automation API .....</b>  | <b>15</b> |
| <b>4.3 Using the IAccessibleEx Interface.....</b>   | <b>20</b> |
| <b>5 Exposing User Interface Element Information .....</b>  | <b>24</b> |
| <b>5.1 Exposing UI Elements with Microsoft Active Accessibility .....</b>   | <b>25</b> |
| <b>5.2 Exposing UI Elements with UI Automation.....</b>   | <b>28</b> |
| <b>6 Exposing User Interface Element Actions.....</b>   | <b>33</b> |
| <b>6.1 Exposing User Interface Element Actions in MSAA .....</b>  | <b>33</b> |
| <b>6.2 Exposing User Interface Element Actions in UI Automation .....</b>   | <b>33</b> |
| <b>7 Keyboard Focus .....</b>   | <b>36</b> |
| <b>7.1 MSAA Keyboard Focus and Selection.....</b>   | <b>36</b> |
| <b>7.2 UI Automation Keyboard Focus and Selection .....</b>   | <b>38</b> |
| <b>8 Events.....</b>  | <b>45</b> |
| <b>8.1 WinEvents .....</b>  | <b>45</b> |
| <b>8.2 UI Automation Events .....</b>   | <b>47</b> |
| <b>9 Programmatic Modifications of States, Properties, Values and Text .....</b>                                    | <b>49</b> |
| <b>9.1 UI Automation Design Considerations.....</b>   | <b>49</b> |
| <b>10 Design Considerations .....</b>   | <b>52</b> |
| <b>10.1 UI Automation Design Considerations.....</b>  | <b>52</b> |
| <b>10.2 IAccessibleEx Design Considerations .....</b>   | <b>60</b> |
| <b>11 Further Information .....</b>   | <b>66</b> |
| <b>11.1 Microsoft Active Accessibility and Extensibility.....</b>   | <b>66</b> |
| <b>11.2 UI Automation Extensibility Features.....</b>   | <b>66</b> |
| <b>Annex A (informative) Microsoft Active Accessibility to Automation Proxy .....</b>                               | <b>69</b> |
| <b>Annex B (informative) UI Automation to Microsoft Active Accessibility Bridge .....</b>                           | <b>78</b> |
| <b>Annex C (informative) UI Automation for W3C Accessible Rich Internet Applications (ARIA) Specification .....</b> | <b>83</b> |
| <b>Annex D (informative) Other Useful APIs for Development and Support of Assistive Technologies.....</b>           | <b>87</b> |
| <b>Bibliography.....</b>  | <b>94</b> |

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide to publish a Technical Report. A Technical Report is entirely informative in nature and shall be subject to review every five years in the same manner as an International Standard.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 13066-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 35, *User interfaces*.

ISO/IEC TR 13066 consists of the following parts, under the general title *Information technology — Interoperability with Assistive Technology (AT)*:

- *Part 1: Requirements and recommendations for interoperability*
- *Part 2: Windows accessibility application programming interface (API)* [Technical Report]
- *Part 3: IAccessible2 accessibility application programming interface (API)* [Technical Report]
- *Part 4: Linux/UNIX graphical environments accessibility API* [Technical Report]<sup>1</sup>
- *Part 6: Java accessibility API* [Technical Report]

---

<sup>1</sup> To be published.

## Introduction

Individuals with a wide range of functional disabilities, impairments, and difficulties require specific technology to enable computers and software to be accessible to them. This part of ISO/IEC TR 13066 provides information about the Microsoft® Windows® Automation Frameworks, including Microsoft Active Accessibility, User Interface (UI) Automation, and the common interfaces of these accessibility frameworks including the IAccessibleEx interface specification.

The intent of this part of ISO/IEC TR 13066 is to provide information and application programming interfaces (APIs) needed to use these frameworks. A primary goal of this part of ISO/IEC TR 13066 is to ensure that accessible software applications can be written in such a way that they are fully compatible with the Microsoft Accessibility APIs available on the Microsoft Windows operating system.

## iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC TR 13066-2:2012](https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012)

<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>

**iTeh STANDARD PREVIEW**  
**(standards.iteh.ai)**

ISO/IEC TR 13066-2:2012

<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>

# Information technology — Interoperability with Assistive Technology (AT) —

## Part 2: Windows accessibility application programming interface (API)

### 1 Scope

This part of ISO/IEC TR 13066 specifies services provided in the Microsoft Windows platform to enable assistive technologies (AT) to interact with other software. One goal of this part of ISO/IEC TR 13066 is to define a set of application programming interfaces (APIs) for allowing software applications to enable accessible technologies on the Microsoft Windows platform. Another goal of this part of ISO/IEC TR 13066 is to facilitate extensibility and interoperability by enabling implementations by multiple vendors on multiple platforms.

This part of ISO/IEC TR 13066 is applicable to the broad range of ergonomics and how ergonomics apply to human interaction with software systems.

### 2 Terms and definitions

<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-4607c50c9122/iso-iec-tr-13066-2-2012>

For the purposes of this document, the following terms and definitions apply.

#### 2.1 application programming interface API

standard set of documented and supported routines that expose operating system programming interfaces and services to applications

**NOTE** An API is usually a source code interface that an operating system, library, or service provides to support requests made by computer programs.

**EXAMPLE** Examples of operating system services that are exposed by APIs include administration and management, diagnostics, graphics and multimedia, networking, security, system services, user interfaces, and accessibility.

#### 2.2 accessibility

degree to which a computer system is easy to use by all people, including those with disabilities

#### 2.3 accessible object

part of user interface object that is accessible by Microsoft Active Accessibility

**NOTE** An accessible object is represented by an IAccessible interface and a ChildId identifier.

2.4

**Accessible Rich Internet Applications**

**ARIA**

accessibility framework from W3C that exposes web content to assistive technologies such as screen readers and speech commanding programs

2.5

**Assistive Technology**

**AT**

technology designed to provide accessibility support to individuals with physical or cognitive impairments or disabilities

NOTE Assistive Technology can be manifested through both hardware and software.

2.6

**Accessibility Toolkit (Linux)**

**ATK**

programming support accessibility features in their applications

2.7

**automation**

replacement of manual operations by computerized methods

NOTE With respect to this part of ISO/IEC 13066, automation is a way to manipulate an application's user interface from outside the application.

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

2.8

**automation element**

object or element that is accessible by the UI Automation object model

NOTE Similar to accessible objects in Microsoft Active Accessibility, an automation element in UI Automation represents a piece or a part of the user interface, such as button, window, or desktop.

ISO/IEC TR 13066-2:2012  
http://standards.iteh.ai/catalog/standards/sist/540-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012

2.9

**Audio Video Interleaved**

**AVI**

format that enables both audio and video data in a file container

2.10

**C#**

a programming language designed for building applications that run on the .NET Framework

NOTE 1 C#, which is an evolution of C and C++, is type safe and object oriented.

NOTE 2 Because it is compiled as managed code, it benefits from the services of the common language runtime, such as language interoperability, security, and garbage collection.

2.11

**callback function**

function or procedure that third party or client code supplies to a component, often by passing a function pointer through the component's API

NOTE The component may then call this code at specific times. This technique is often used by components to signal client code that some event has taken place, or to request client code to perform some specific task.



## 2.12 clients

component that uses the services of another component

NOTE In this part of ISO/IEC 13066, client refers more specifically to a component that uses the services of Microsoft Active Accessibility or UI Automation, or both, to access, identify, or manipulate the user interface (UI) elements of an application.

## 2.13 Common Language Runtime CLR

Microsoft's commercial implementation of the Common Language Infrastructure (CLI) specification

NOTE 1 The CLI provides a specification for executable code and the execution environment in which it runs

NOTE 2 At the center of the CLI is a unified type system, a virtual execution system, and a specification for multiple programming languages to share the type system and compile into a common intermediate language.

## 2.14 Component Object Model COM

object-oriented programming model that defines how objects interact within a single process or between processes

NOTE In COM, clients have access to an object through interfaces implemented on the object.

## 2.15 content view

subset of the control view of the UI Automation tree

NOTE The content view contains UI items that convey the actual information in a user interface, including UI items that can receive keyboard focus and some text that is not a label on a UI item.

## 2.16 control pattern

<UI Automation> design implementation that describes a discrete piece of functionality for a control

NOTE This functionality can include the visual appearance of a control and the actions it can perform.

## 2.17 control view

subset of the raw view of the UI Automation tree

NOTE The control view includes the UI items that provide information to the user or enable the user to perform an action.

## 2.18 enumerator

object that iterates through its associated collection

NOTE An enumerator can be thought of as a movable pointer to any element in the collection.

## 2.19 Global Unique Identifier GUID

unique reference number used as an identifier in computer software

**2.20  
HWND**

unique long integer value that is assigned by Microsoft Windows to the current window, where a window is a primitive of Windows' UI management

**2.21  
in-process**

<UI Automation> Microsoft Accessibility code that is executed in a target application's process

**2.22  
Java Accessibility Application Programming Interface  
JAAPI**

accessibility framework for the Java SE platform that exposes Java applications to assistive technologies such as screen readers and speech commanding programs

**2.23  
Java Development Kit  
JDK**

collection of programming tools

**2.24  
Java Virtual Machine  
JVM**

environment in which Java bytecode can be executed

**2.25  
managed API**

API that, when compiled and run, is under the control of an intermediary runtime infrastructure, like a virtual machine

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

EXAMPLE Microsoft's Common Language Runtime (CLR) and the Java Virtual Machine (JVM) are examples of runtime infrastructures.

ISO/IEC TR 13066-2:2012  
<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>

NOTE Managed code is compiled into an intermediate language construct (for example, byte code) and the runtime infrastructure handles the compilation into machine code. The runtime infrastructure handles programming constructs like memory management.

**2.26  
Microsoft Active Accessibility**

COM-based technology that improves the way accessibility aids work with applications running on Microsoft Windows

NOTE Microsoft Active Accessibility provides dynamic-link libraries (DLLs) that are incorporated into the operating system, as well as a COM interface and application programming elements that provide reliable methods for exposing information about user interface elements.

**2.27  
MSDN**

Microsoft Developer Network, which is a technical information resource for developers who are using Microsoft technologies

**2.28  
Multiple Document Interface  
MDI**

document interface that allows a window to reside under a parent window

**2.29****native API**

API that, when compiled and run, is not under the control of an intermediary runtime infrastructure such as a virtual machine or CLR

NOTE Native code compiles directly to machine code, and the developer is responsible for most aspects of programming constructs (for example, pointers, freeing memory, and so on). Also known as a native API.

**2.30****out-of-process**

<UI Automation> Microsoft Accessibility code that is executed in a different process from the target application's process

**2.31****providers**

<UI Automation> components that expose information about UI elements

EXAMPLE Such components can be applications, DLLs, and so on

NOTE These include any control, module, or application which implements UI Automation provider interfaces.

**2.32****raw view**

full tree of UI Automation element objects in the UI Automation tree for which the desktop is the root

NOTE The raw view closely follows the native programmatic structure of an application and, therefore, is the most accurate view of the UI structure. It is also the base on which the other views of the tree are built

**2.33****root element**

element of the UI Automation tree that represents the current desktop and whose child elements represent application windows <https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>

NOTE Each of these child elements can contain elements representing pieces of UI such as menus, buttons, toolbars, and list boxes.

**2.34****servers**

components of Microsoft Active Accessibility that have UI elements and expose information about the UI elements and/or allow them to be manipulated

EXAMPLE Such components can be applications, DLLs, and so on.

NOTE A Microsoft Active Accessibility server has the same role as a UI Automation provider.

**2.35****simple element**

<Microsoft Active Accessibility> element that shares an IAccessible object with other peer elements

NOTE A simple element relies on the shared IAccessible object (typically its parent in the object hierarchy) to expose its properties.

**2.36****Services Control Manager****SCM**

system process that maintains a database of installed services and driver services, and provides a unified and secure means of controlling them

NOTE The database includes information on how each service or driver service should be started. It also enables system administrators to customize security requirements for each service and thereby control access to the service.

**2.37**  
**system service**

application conforming to the interface rules of the Service Control Manager (SCM)

NOTE 1 It can be started automatically at system boot, by a user through the Services control panel applet, or by an application that uses the service functions

NOTE 2 Services can execute even when no user is logged on to the system. File services, indexing service, memory management, power management, and remote desktop services are examples of services.

**2.38**  
**Text Services Framework**  
**TSF**

simple and scalable framework for the delivery of advanced text input and natural language technologies

NOTE 1 TSF can be enabled in applications, or as a TSF text service

NOTE 2 A TSF text service provides multilingual support and delivers text services such as keyboard processors, handwriting recognition, and speech recognition.

**2.39**  
**user interface**  
**UI**

mechanisms by which a person interacts with a computer system

NOTE 1 The user interface provides input mechanisms, allowing users to manipulate a system

NOTE 2 It also provides output mechanisms, allowing the system to produce the effects of the users' manipulation.

**2.40**  
**User Interface Automation**  
**UI Automation**  
**UIA**

accessibility framework that exposes applications to software automation or to assistive technologies such as screen readers and speech commanding programs

ISO/IEC TR 13066-2:2012  
<https://standards.iteh.ai/catalog/standards/sist/e71b8c2e-5e40-449a-b4fc-d6c7e58e3122/iso-iec-tr-13066-2-2012>

**2.41**  
**virtual machine**  
**VM**

computer within a computer, implemented in software

NOTE 1 A virtual machine emulates a complete hardware system, from processor to network card, in a self-contained, isolated software environment, enabling the simultaneous operation of otherwise incompatible operating systems.

NOTE 2 Each operating system runs in its own isolated software partition.

**2.42**  
**Visual Basic**  
**VB**

generally visual programming environment from Microsoft based on the BASIC programming language

**2.43**  
**Web Accessibility Initiative**  
**WAI**

an effort to improve the accessibility of the World Wide Web

**2.44**  
**WinEvents**

mechanism that allows servers and the Windows operating system to notify clients when an accessible object changes

## 2.45

**World Wide Web Consortium****W3C**

standards organization for the World Wide Web

### 3 General Description and Architecture of the Microsoft Windows Automation API

#### 3.1 General Description

The Microsoft® Windows® Automation API consists of two accessibility frameworks — Microsoft Active Accessibility and User Interface Automation (UI Automation). The IAccessibleEx interface specification integrates the two accessibility frameworks.

Although Microsoft Active Accessibility and UI Automation are two different frameworks, the basic design principles are similar. The purpose of both is to expose rich information about the UI elements used in Windows applications. Developers of accessibility tools can use this information to help make applications more accessible to people with vision, hearing, or motion disabilities.

##### 3.1.1 Microsoft Active Accessibility Overview

Microsoft Active Accessibility is based on the Component Object Model (COM), which defines a common way for applications and operating systems to communicate. The goal of Microsoft Active Accessibility is to allow custom controls to expose basic information, such as name, location on screen, or type of control, and state information such as visibility, enabled, or selected.

The *accessible object* is the central object of Microsoft Active Accessibility and is represented by an IAccessible COM interface and an integer ChildId. It allows applications to expose UI elements as a tree structure that represents the structure of the UI. Each element of this tree exposes a set of properties and methods that allow the corresponding UI element to be manipulated. Microsoft Active Accessibility clients can access the programmatic UI information through a standard API. The following sections describe the main parts of Microsoft Active Accessibility, including accessible objects, the WinEvents mechanism, the Microsoft Active Accessibility runtime (Oleacc.dll), and Microsoft Active Accessibility clients and servers.

##### 3.1.1.1 Microsoft Active Accessibility Components

Microsoft Active Accessibility contains the following main components:

- Accessible Object – A logical UI element (such as a button) that is represented by an IAccessible COM interface and a ChildId value.
  - The IAccessible interface has properties and methods for obtaining information about and manipulating UI elements.
  - ChildId is an identifier for an accessible object that is used together with an IAccessible instance to refer to a specific UI element.
- WinEvents – An event system that allows servers to notify clients when an accessible object changes. For more information, see Events.
- Oleacc.dll – A run-time dynamic-link library that provides the Microsoft Active Accessibility API and the accessibility system framework. Oleacc.dll also provides proxy objects for the Windows operating system standard controls.

### 3.1.1.2 Oleacc.dll

The following APIs and functions are included in Oleacc.dll:

- Client APIs – APIs that clients use to request an IAccessible interface pointer (for example, AccessibleObjectFromX).
- Server APIs– APIs that servers use to return an IAccessible interface pointer to a client (for example, LresultFromObject).
- APIs for getting localized text for the role and state codes (for example, GetRoleText and GetStateText).
- Helper APIs (for example, AccessibleChildren).
- Proxies– Code that provides the default implementation of an IAccessible interface for standard USER and COMCTL controls. Because these controls implement the IAccessible interface on behalf of the system controls, they are known as *proxies*.

### 3.1.1.3 Microsoft Active Accessibility Clients

Microsoft Active Accessibility helps accessibility aids, called *clients*, interact with standard and custom UI elements of other applications and the operating system. Clients can use Microsoft Active Accessibility to access, identify, and manipulate an application's UI elements. Clients include accessibility aids, automated testing tools, and computer-based training applications.

Clients must know when the server's UI changes so that information can be conveyed to the user. They are notified about changes in the server UI by registering to receive notifications of specific changes through a mechanism called *Window Events*, or *WinEvents*. For more information, see *Events*.

To learn about and manipulate a particular UI element, clients use a pair consisting of an IAccessible interface and a ChildId.

### 3.1.1.4 Microsoft Active Accessibility Servers

Applications that interact with and provide information to clients are called *servers*. Servers include any control, module, or application that implements Microsoft Active Accessibility. A server uses Microsoft Active Accessibility to provide information about its UI elements to clients.

## 3.1.2 UI Automation Overview

UI Automation provides programmatic access to UI elements on the desktop, enabling assistive technology products such as screen readers to provide information about the UI to end users and to manipulate the UI by means other than standard input. UI Automation also allows automated test scripts to interact with the UI. The UI Automation Specification is designed so that it can be supported across platforms other than Microsoft Windows.

UI Automation is broader in scope than just an interface definition. UI Automation provides:

- A set of classes that make it easy for client applications to receive events, retrieve property values, and manipulate UI elements.
- A core infrastructure for doing fetch, find, and similar operations efficiently across process boundaries.
- A set of interfaces for providers to express the UI as a tree structure, along with some general properties.

- A set of interfaces that providers use to express other properties and functionality specific to the control type. These are the control pattern interfaces.

To improve on Microsoft Active Accessibility, UI Automation aims to address the following goals:

- Enable efficient *out-of-process* clients, while continuing to allow *in-process* access.
- Expose more information about the UI in a way that allows clients to be out-of-process.
- Co-exist with and use Microsoft Active Accessibility, but do not inherit problems that exist in Microsoft Active Accessibility.
- Provide an alternative to `IAccessible` that is simple to implement.

The Microsoft Windows implementation of UI Automation features COM-based interfaces and managed interfaces that are included with the Microsoft .NET Framework.

### 3.1.2.1 UI Automation Components

UI Automation has four main components, as shown in the following table.

| Component            | Description   |
|----------------------|---|
| Provider API         | A set of COM interfaces that are implemented by UI Automation providers. UI Automation providers are objects that provide information about UI elements and respond to programmatic input.                          |
| Client API           | A set of COM interfaces that enable client applications to obtain information about the UI and to send input to controls.   |
| UiAutomationCore.dll | The run-time library, sometimes called the UI Automation core, that handles communication between providers and clients.  |
| Oleacc.dll           | The run-time library for Microsoft Active Accessibility and the proxy objects. The library also provides proxy objects used by the Microsoft Active Accessibility to UI Automation Proxy to support Win32 controls. |

UI Automation can be used to create support for custom controls by using the provider API, and to create client applications that use the UI Automation core to communicate with UI elements.

### 3.1.2.2 UI Automation Model

UI Automation exposes every element of the UI to client applications as an object represented by the `IUIAutomationElement` interface. Elements are contained in a tree structure, with the desktop as the root element. Clients can filter the raw view of the tree as a control view or a content view. Applications can also create custom views.

A UI Automation element exposes properties of the control or UI element that it represents. One of these properties is the control type, which defines the basic appearance and functionality of the control or UI element as a single recognizable entity, for example, a button or check box.

In addition, a UI Automation element exposes one or more control patterns. A control pattern provides a set of properties that are specific to a particular control type. A control pattern also exposes methods that enable client applications to get more information about the element and to provide input to the element.

UI Automation provides information to client applications through events. Unlike WinEvents, UI Automation events are not based on a broadcast mechanism. UI Automation clients register for specific event notifications and can request that specific properties and control pattern information be passed to their event handlers. In