
**Information technology —
Interoperability with assistive
technology (AT) —**

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

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

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

PROOF/ÉPREUVE

Reference number
ISO/IEC TR 13066-2:2015(E)



iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/b8c39e81-3a29-4611-bd30-ce6b3394044d/iso-iec-tr-13066-2-2016>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword	v
Introduction	vi
1 Scope	1
2 Terms and definitions	1
3 General description and architecture of the Microsoft Windows Automation API	7
3.1 General description.....	7
3.1.1 Microsoft Active Accessibility overview.....	7
3.1.2 UI Automation overview.....	8
3.1.3 IAccessibleEx interface.....	9
3.2 Architecture.....	10
4 Using the API	12
4.1 Using the Microsoft Active Accessibility API.....	12
4.1.1 Types of Microsoft Active Accessibility support.....	13
4.1.2 Retrieving an accessible object.....	13
4.1.3 The WM_GETOBJECT message.....	13
4.1.4 Special values of Object Identifier.....	14
4.2 Using the UI Automation API.....	15
4.2.1 UI Automation model.....	15
4.2.2 UI Automation tree.....	16
4.2.3 UI Automation control patterns, control types, properties, and events.....	16
4.3 Using the IAccessibleEx interface.....	21
4.3.1 The IAccessibleEx interface implementation.....	21
5 Exposing User Interface Element Information	24
5.1 General.....	24
5.2 Exposing UI Elements with Microsoft Active Accessibility.....	25
5.2.1 How an MSAA Server exposes relevant properties.....	25
5.2.2 Provide support for the Accessible Object structure.....	26
5.2.3 Support hit testing.....	27
5.2.4 Generate appropriate WinEvents.....	27
5.2.5 Object identifier.....	27
5.2.6 How MSAA clients access exposed UI elements.....	28
5.3 Exposing UI Elements with UI Automation.....	28
5.3.1 Types of providers.....	28
5.3.2 UI Automation provider concepts.....	28
5.3.3 Provider interfaces.....	29
5.3.4 Property values.....	30
5.3.5 Provider navigation.....	30
5.3.6 Provider reparenting.....	31
5.3.7 Provider repositioning.....	31
5.3.8 How UI Automation clients access exposed UI Elements.....	32
6 Exposing UI Element actions	33
6.1 Exposing UI Element actions in MSAA.....	33
6.2 Exposing UI Element actions in UI Automation.....	33
6.2.1 UI Automation control pattern components.....	33
6.2.2 Control patterns in providers and clients.....	34
6.2.3 Dynamic control patterns.....	34
6.2.4 Control patterns and related interfaces.....	34
7 Keyboard focus	36
7.1 MSAA keyboard focus and selection.....	36
7.1.1 Focus and selection properties and methods.....	36
7.1.2 Events triggered in menus.....	37
7.2 UI Automation keyboard focus and selection.....	37

7.2.1	Focus.....	37
7.2.2	Selection.....	38
8	Events.....	44
8.1	WinEvents.....	44
8.1.1	USER's role in WinEvents.....	44
8.1.2	Receiving event notifications.....	45
8.1.3	Sending events.....	45
8.1.4	The allocation of WinEvent IDs.....	45
8.2	UI Automation events.....	46
8.2.1	How providers raise events.....	47
8.2.2	How clients register for and process events.....	48
9	Programmatic modifications of states, properties, values, and text.....	48
9.1	UI Automation specifications.....	48
9.1.1	Introduction.....	48
9.1.2	UI Automation elements.....	49
9.1.3	UI Automation tree.....	49
9.1.4	UI Automation properties.....	50
9.1.5	UI Automation control patterns.....	50
9.1.6	UI Automation control types.....	50
9.1.7	UI Automation events.....	50
10	Design considerations.....	51
10.1	UI Automation design considerations.....	51
10.1.1	UI Automation clients.....	51
10.1.2	UI Automation providers.....	54
10.1.3	Coexistence and interoperability with Microsoft Active Accessibility.....	57
10.2	IAccessibleEx design considerations.....	58
10.2.1	Design consideration for providers before implementing the IAccessibleEx interface.....	58
10.2.2	IAccessibleEx interface for providers.....	58
10.2.3	IAccessibleEx interface for clients.....	59
11	Further Information.....	63
11.1	Microsoft Active Accessibility and Extensibility.....	63
11.2	UI Automation extensibility features.....	63
11.2.1	Registration of custom UI Automation properties, events, and control patterns.....	63
11.2.2	How clients and providers support custom control patterns.....	64
	Annex A (informative) Microsoft Active Accessibility to Automation Proxy.....	65
	Annex B (informative) UI Automation to Microsoft Active Accessibility Bridge.....	72
	Annex C (informative) UI Automation for W3C Accessible Rich Internet Applications (ARIA) Specification.....	77
	Annex D (informative) Other Useful APIs for Development and Support of Assistive Technologies.....	81
	Bibliography.....	88

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: [Foreword — Supplementary information](#).

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 35, *User interfaces*.

This second edition cancels and replaces the first edition (ISO/IEC/TR 13066-2:2012), which has been technically revised.

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)*
- *Part 3: IAccessible2 accessibility application programming interface (API)*
- *Part 4: Linux/UNIX graphical environments accessibility API*
- *Part 6: Java accessibility application programming interface (API)*

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 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 13066 is to provide information and application programming interfaces (APIs) needed to use these frameworks. A primary goal of this part of ISO/IEC 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)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/b8c39e81-3a29-4611-bd30-ce6b3394044d/iso-iec-tr-13066-2-2016>

Information technology — Interoperability with assistive technology (AT) —

Part 2: Windows accessibility application programming interface (API)

1 Scope

This part of ISO/IEC 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 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 13066 is to facilitate extensibility and interoperability by enabling implementations by multiple vendors on multiple platforms.

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

2 Terms and definitions

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 1 to entry: 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 1 to entry: 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 1 to entry: Assistive Technology can be manifested through both hardware and software.

2.6

Accessibility Toolkit

Linux Accessibility Toolkit

ATK

programming support accessibility features in their applications

2.7

automation

replacement of manual operations by computerized methods

Note 1 to entry: With respect to this part of ISO/IEC 13066, automation is a way to manipulate an application's user interface from outside the application.

2.8

automation element

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

Note 1 to entry: 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.

2.9

Audio Video Interleaved

AVI

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

2.10

C#

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

Note 1 to entry: C#, which is an evolution of C and C++, is type safe and object-oriented.

Note 2 to entry: 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 1 to entry: 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 1 to entry: 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 to entry: The CLI provides a specification for executable code and the execution environment in which it runs

Note 2 to entry: At the centre of the CLI are 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 1 to entry: 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 1 to entry: 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 1 to entry: 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 1 to entry: 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 1 to entry: 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

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

Note 1 to entry: 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 1 to entry: 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

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

2.28

Multiple Document Interface

MDI

A 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 1 to entry: 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 1 to entry: 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 1 to entry: 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

Note 1 to entry: 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 1 to entry: 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 1 to entry: 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 1 to entry: 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 to entry: 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 to entry: 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 to entry: TSF can be enabled in applications, or as a TSF text service.

Note 2 to entry: 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 to entry: The user interface provides input mechanisms, allowing users to manipulate a system.

Note 2 to entry: 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

2.41

virtual machine

VM

computer within a computer, implemented in software

Note 1 to entry: 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 to entry: Each operating system runs in its own isolated software partition.

2.42

Visual Basic

VB

visual programming environment from Microsoft based on the BASIC programming language

2.43

Web Accessibility Initiative

WAI

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

A 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 [Clause 8](#).
- `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`).