INTERNATIONAL STANDARD

ISO/IEC 13066-1

First edition
2011-05-15

Information technology — Interoperability with assistive technology (AT) —

Part 1: Requirements and recommendations for interoperability

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

Partie 1: Exigences et recommandations pour l'interopérabilité

Reference number
ISO/IEC 13066-1:2011(E)

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
Web www.iso.org

Published in Switzerland

Contents

Page

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.

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

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

— *Part 1: Requirements and recommendations for interoperability*

The following parts are under preparation:

— *Part 2: Windows accessibility API* [Technical Report]

— *Part 3: I-Accessible-2 accessibility API* [Technical Report]

Introduction

Interoperability involves the ability to use assistive technology (AT) to add to or augment existing components of information technology (IT) systems. Interoperability between AT and IT is best facilitated via the use of standardized, public interfaces for all IT components.

This part of ISO/IEC 13066 provides a basis for designing and evaluating interoperability between IT and AT. It formalizes the layered architecture of hardware-to-hardware, hardware-to-software, and software-to-software connections that have long been implicit in the IT definitions of ISO/IEC JTC 1. It also recognizes the central role that accessibility application programming interfaces (accessibility APIs) play in aiding this interoperability.

This part of ISO/IEC 13066 identifies a variety of APIs that are described further in other parts of ISO/IEC 13066. These APIs can be used as frameworks to support IT–AT interoperability.

Information technology — Interoperability with assistive technology (AT) —

Part 1:
Requirements and recommendations for interoperability

1 Scope

This part of ISO/IEC 13066 defines the responsibilities of different information technology (IT) and assistive technology (AT) functional units in supporting interoperability. It recognizes that AT can be provided both as functional units that are installed or otherwise connected to a system or can be utilized by being provided as a service which is accessed via communications connections. It bases these responsibilities on fundamental IT definitions of major types of functional units. It focuses on the utilization of standard, public interfaces for functional units and on the provision of accessible documentation of their capabilities.

This part of ISO/IEC13066 recognizes that IT is implemented both in conventional computer systems and as a major component of other systems within the wider scope of information and communications technology (ICT). This part of ISO/IEC 13066 recognizes the fundamental role of operating systems and application programming interfaces (APIs), in managing interoperability, and in providing guidance to developers of other functional units. It also recognizes that different operating systems will have their own standardized methods of supporting interoperability.

This part of ISO/IEC 13066 does not define or require specific technology, commands, APIs, or hardware interfaces. It defers to other existing standards and supports the development of new standards in these areas.

It identifies a variety of common accessibility APIs that are further described in other parts of ISO/IEC 13066.

2 Terms and definitions

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

2.1
accessibility API
set of programming interfaces designed specifically to provide accessibility services

NOTE An accessibility API is a special instance of an API.

2.2
accessibility services
services provided by an operating system or other platform software, commonly in the form of APIs (application programming interfaces) that are used by software to expose information about the user interface and events to assistive technologies and that provide two-way communication with assistive technologies, including exposing information about objects and events

NOTE Accessibility services might provide additional information used by assistive technologies, e.g. about operating system status.

2.3
application programming interface
API
collection of invocation methods and associated parameters used by one piece of software to request actions from another piece of software

[ISO/IEC 18012-1, definition 3.1.1]

2.4
application software
software that is specific to the solution of an application problem

[ISO/IEC 2382-1, definition 10.04.01]

EXAMPLE A spreadsheet program.

2.5
assistive technology
AT
hardware or software that is added to or incorporated within a system that increases accessibility for an individual

EXAMPLES Braille displays, screen readers, screen magnification software and eye-tracking devices.

[ISO 9241-171, definition 3.5]

NOTE 1 Assistive technology can be helpful to individuals with disabilities or other specialized needs.

NOTE 2 Within this document, where assistive technology (and its abbreviation AT) is used, it is to be considered as both singular and plural, without distinction. If it is to be used in the singular only, it will be preceded by the article “an” (i.e. an assistive technology). If it is to be used in the plural only, it will be preceded by the adjective “multiple” (i.e. multiple AT).

2.6
compatibility
capability of a functional unit to meet the requirements of a specified interface without appreciable modification

[ISO/IEC 2382-1, definition 10.06.11]

2.7
computer
functional unit that can perform substantial computations, including numerous arithmetic operations and logic operations, without human intervention

[ISO/IEC 2382-1, definition 10.03.03]

NOTE A computer can consist of a stand-alone unit or several interconnected units.

2.8
computer system
system
one or more computers, peripheral equipment, and software that perform data processing

[ISO/IEC 2382-1, definition 10.01.20]

2.9
connectivity
capability of a system or device to be attached to other systems or devices without modification

[ISO/IEC 2382-1, definition 10.03.27]

2.10
device driver
software component that permits a system to control and communicate with a peripheral device

[IEEE Std. 610.10-1994, IEEE Std. 610.12-1990, definition 3.542]

2.11
function
defined objective or characteristic action of a system or component

[IEEE Std. 610.12-1990, unnumbered definition]

EXAMPLE A system has inventory control as its primary function.

2.12
functional unit
entity of hardware or software, or both, capable of accomplishing a specified purpose

[ISO/IEC 2382-1, definition 10.01.40]

2.13
hardware
all or part of the physical components of an information processing system

[ISO/IEC 2382-1, definition 10.01.07]

EXAMPLES Computers and peripheral devices.

2.14
information/communication technology
ICT
technology for gathering, storing, retrieving, processing, analysing and transmitting information

[ISO 9241-20, definition 3.4]

EXAMPLE A computer system.

2.15
interface
shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical interconnections, signal exchanges, and other characteristics, as appropriate

[ISO/IEC 2382-1, definition 10.01.38]

2.16
interoperability
capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units

[ISO/IEC 2382-1, definition 10.01.47]

2.17
operating system
OS
software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input-output control, and data management

NOTE Although operating systems are predominantly software, partial hardware implementations are possible.

[ISO/IEC 2382-1, definition 10.04.08]

2.18
operation
process of running a computer system in its intended environment to perform its intended functions

[IEEE Std. 610.12-1990, unnumbered definition]

2.19
peripheral equipment
device that is controlled by and can communicate with a particular computer

[ISO/IEC 2382-1, definition 10.03.07]

EXAMPLES Input–output units and external storage.

2.20
platform software
collection of software components that runs on an underlying software or hardware layer, and that provides a set of software services to applications that allow them to be isolated from the underlying software or hardware layer

NOTE A particular software component might play the role of a platform in some situations and not in others. Platforms can include such things as internet browsers, operating systems, plug-ins to internet browsers or other software applications, and, under some situations, byte-code interpreted virtual environments and other "programming within another programming" environments.

2.21
service
functionality made available to a user electronically

[ISO/IEC 24752-1, definition 4.27]

EXAMPLES Airline reservation service, currency translation services, weather forecasting, and restaurant recommendations.

2.22
role
semantic association which allows tools to present and support interaction with the object in a manner that is consistent with user expectations about other objects of that type

EXAMPLES Checkbox, menu item, list item, and table column header.

2.23
software
all or part of the programs, procedures, rules, and associated documentation of an information processing system

NOTE Software is an intellectual creation that is independent of the medium on which it is recorded.

[ISO/IEC 2382-1, definition 10.01.08]

2.24
support software
software that aids in the development, maintenance, or use of other software or provides general application-independent capability

[ISO/IEC 2382-1, definition 10.04.03]

EXAMPLES Compilers and database management systems.

2.25
system software
application-independent software that supports the running of application software

[ISO/IEC 2382-1, definition 10.04.02]

EXAMPLE An operating system, a Web browser, or a programming environment (e.g. Java) can be used as a platform for application software.

NOTE Platform software (2.20) is similar to but not always the same as system software.

2.26
user interface element
user interface object
entity of the user interface that is presented to the user by the software

[ISO 9241-171 definition 3.38]

NOTE 1 User interface elements may or may not be interactive.

NOTE 2 Both entities relevant to the task and entities of the user interface are regarded as user interface elements. Different user interface element types are text, graphics and controls. A user interface element may be a representation or an interaction mechanism for a task object (such as a letter, a sales order, electronic parts, or a wiring diagram) or a system object (such as a printer, hard disk, or network connection). It may be possible for the user to directly manipulate some of these user interface elements.

EXAMPLE 1 User interface elements in a graphical user interface include such things as basic objects (such as window title bars, menu items, push buttons, image maps, and editable text fields) or containers (such as windows, grouping boxes, menu bars, menus, groups of mutually-exclusive option buttons, and compound images that are made up of several smaller images).

EXAMPLE 2 User interface elements in an audio user interface include such things as menus, menu items, messages, and action prompts.

EXAMPLE 3 User interface elements in tactile interfaces include such things as tactile dots, tactile bars, surfaces, knobs, and grips.

2.27
boundary
⟨user interface element⟩ physical display area occupied by a particular user interface element when outputting it on a display

3 Conformance

3.1 Applying the requirements

This part of ISO/IEC 13066 contains requirements and recommendations for a variety of different products. Where a requirement does not identify a particular type of product, it is expected to apply to all types of ICT products.

All requirements in Clauses 5–9 shall be implemented by the products to which they apply.

3.2 Applying the recommendations

Individual recommendations in Clauses 5–9 should be evaluated for their applicability to the particular product. The applicable recommendations shall be implemented.

NOTE This has the effect of transforming applicable recommendations into additional requirements.

3.3 Evaluation of products

If a product is claimed to conform to this part of ISO/IEC 13066 then the procedures used to establish the product's requirements (as identified in 3.1 and 3.2), and to evaluate the product based on these requirements, shall be specified. The level of detail of the specification is a matter of negotiation between the involved parties.

4 Framework for IT-AT interoperability

4.1 Assistive technology

AT connects to ICT hardware or software components to modify, duplicate, or replace the user interface functionalities of those components.

EXAMPLE 1 A glare filter is physically attached to a display to modify the way in which a user is able to see the information on a visual display.

EXAMPLE 2 A single switch is used with an on-screen keyboard to replace the functionality of a standard keyboard.

EXAMPLE 3 A voice recognition program can provide the user with either an alternate or a duplicate method of entering data into a computer that is equipped with a microphone.

NOTE AT can be provided as a service without the need of being installed on an individual system

4.2 Interconnection

4.2.1 Introduction to interconnection

AT typically make use of standard connections between ICT components. For interconnection to take place, it is important that standard interfaces that are expected by AT be available to them.

The use of standard interfaces means that AT do not have to interoperate with ICT in unsupported, undocumented, and non-standard methods.

NOTE Unsupported, undocumented, and non-standard methods often lead to incompatibilities.

EXAMPLE 1 A Braille display is connected to a computer via a Universal Serial Bus (USB) interface.

EXAMPLE 2 Output from an application program is processed by screen magnification software prior to its being sent to the display driver in an operating system which forwards the magnified information to a video display device.

EXAMPLE 3 On-screen keyboard software is connected by the operating system and provided to an application for use instead of a physical keyboard.

4.2.2 Types of standard connections

Connections can be classified as:

a) Hardware-to-hardware connections;

b) Hardware-to-software connections;

c) Software-to-software connections.

4.2.3 Hardware-to-hardware connections

Hardware to hardware connections involve physical and/or logical interfaces that support the transfer or communication of information between the connected hardware or between the user and the hardware.

External hardware-to-hardware connections are intended to provide easy connection of peripheral devices (including AT) to a computer or another device. These connections benefit from being standardized. Hardware to hardware connections can be subdivided into:

a) wired connections (e.g. monitor, USB, speaker, microphone, Ethernet);

b) wireless connections (e.g. WI-FI, Bluetooth, infra-red);

c) non-electronic connections (e.g. the ability to place a glare filter over a display screen, the ability to use a key guard).

NOTE Internal hardware-to-hardware connections within a computer are primarily intended for connecting parts of a computer to one another (e.g. the connection used for a laptop screen to the processing capabilities of the laptop). Because of their internal nature, internal hardware to hardware connections are not necessarily standardized.

4.2.4 Hardware-to-software connections

Hardware to software connections are provided by device drivers within the ICT's system software interacting with the system's external hardware to hardware connections.

While it can be possible for other programs, besides device drivers, to provide instructions directly to external hardware, this practice generally results in interconnectivity problems and is discouraged.

4.2.5 Software-to-software connections

The standard method of software to software connection with platform software is via an Application Programming Interface (API) that has been defined by the platform software.

NOTE 1 The term "platform software" can be used to refer to any system or application software that provides services to other software from the underlying layers. It could be the operating system or it could be a browser or any application runtime environment (which might also be considered as application or support software).

NOTE 2 According to ISO/IEC 2382-1 there are three classes of software:

a) System software, which includes the operating system and other instance of platform software;

b) Application software;

c) Support software.

It is important to recognize the ISO/IEC 2382-1 definition of "application-independent software that supports the running of application software" is not specific to operating systems and is not specific to software provided by operating system vendors. This definition is consistent with the common use of the term "platform software".

4.2.6 AT-IT Connectivity

AT can be connected to ICT in the following ways:

a) An AT device can connect to another device that is connected to a computer or to ICT.

b) An AT device can replace another device that is connected to a computer or to ICT.

c) AT software can connect to the operating system.

d) AT software can replace some of the functionality provided by system software.

e) AT software can connect to an application program.

f) AT software can connect to a utility software program.

4.2.7 Introduction to interoperability

Interoperability involves more than the provision of standard connections. Interoperability involves the ability to effectively communicate between AT and the component(s) to which it is connected.

4.2.8 Goals for interoperability

The goals of AT-IT interoperability include:

a) The number of individuals who can operate ICT with (or without) standard AT is maximized.

b) The efficient use of all of the features offered by the ICT products by all individuals, including those with disabilities.

c) Individuals are enabled to use all relevant functions to operate ICT via AT.

d) Various types of AT can be easily and simply connected to ICT.

e) Standard methods are available to interface AT with ICT systems that use standard control commands.

f) Compatibility is maintained, even when version changes occur in either the AT or ICT products.

4.2.9 Hardware-to-hardware interoperability

Hardware-to-hardware interoperability is achieved by the effective communications of commands and data across hardware to hardware connections.

Hardware-to-hardware interoperability is enabled by using standard sets of hardware commands and interconnections. Hardware commands are often specific to a type of device.

Interoperability is enhanced by various devices sharing a common set of hardware commands.

Common sets of hardware commands are generally defined based on the type of devices that they apply to.

EXAMPLE ETSI TS 102 511 identifies current and potential AT Commands for Assistive Mobile Device Interfaces.

4.2.10 Hardware-to-software interoperability

Hardware-to-software interoperability is achieved by device drivers effectively translating between hardware commands and data and software commands and data.

Each operating system provides its own standard for how device drivers operate and typically provides a basic set of generic device drivers.

Manufacturers of peripheral devices can utilize these generic device drivers or can provide specialty device drivers to be installed within the system software, which provide advanced functionality for controlling features of their devices.

Interoperability is enhanced by various device drivers sharing common functionalities and common sets of hardware and software commands.

4.2.11 Software-to-software interoperability

Software-to-software interoperability is achieved through pre-defined application programming interfaces (or API's). Each operating system provides its own standard for how this is done and the specific set of API's. Additionally, APIs can be provided by other forms of system software (e.g. Web browsers or programming environments) which run on operating systems and which are in turn used to provide a platform for running application software.

5 Requirements and recommendations on hardware-to-hardware interoperability

5.1 Responsibilities of ICT manufacturers

5.1.1 Presence of hardware interfaces

ICT systems shall include hardware interfaces that support connection of industry standard user input and output devices.

NOTE 1 While directly providing currently available interfaces is preferable, this requirement can be met by providing interfaces to which currently available adapters can be connected, where these adapters can connect to standard user input and output devices.

NOTE 2 The particular hardware interfaces that an ICT system includes will vary based on current technology and on the physical characteristics of the ICT.

NOTE 3 While USB 2.0 interfaces are most common at the time of developing this standard, some other technology may replace it in the future.

NOTE 4 While computer systems intended for permanent locations include a wider variety of specialized interfaces, small, highly portable computers (e.g. Personal Digital Assistants), and other ICT systems often use a single USB connection to perform all types of hardware interfacing.

5.1.2 Following hardware interface standards

Standard hardware interfaces (e.g. USB ports, PC Card interfaces) shall comply precisely with applicable specifications.

NOTE AT products rely on standard hardware interfaces and assume they conform to a given standard. Many times, a hardware manufacturer will step outside of the standards and make their own enhancements/changes to these hardware interfaces which subsequently cause it to become incompatible with AT.

EXAMPLE A computer that ships with its own low-power USB mouse pointer that doesn't require a lot of power still supplies the standard prescribed power to its USB ports, so that an AT can be plugged into that port replacing the mouse, with the assumption the power will be there.

5.2 Responsibilities of device manufacturers

5.2.1 Use of standard hardware interfaces

Peripherals should make use of standard hardware interfaces, wherever possible.

NOTE Where device manufacturers choose to utilize non-standard hardware interfaces (for performance or other reasons), they are encouraged to also produce alternate models of the device that can utilize standard hardware interfaces. Where alternate models cannot be provided, the manufacturers can at least provide discuss how non-standard the components of the hardware interface map to a standard hardware interface and/or API.

5.2.2 Support of standard functionality and commands

Where applicable hardware command standards exist, device functionality shall be accessible via standard hardware commands.

NOTE 1 Where device manufacturers choose to utilize non-standard commands to replace standardized commands (for performance or other reasons), it is important for them to also support the standardized commands that they have chosen to replace.

NOTE 2 It is especially important that accessibility functionalities are either unaltered or enhanced in a manner consistent with providing the same service as if they were unaltered.

5.2.3 Providing information about non-standard functionality of devices

Manufacturers of devices with functionalities that are not accessible via standardized commands, shall provide accessible documentation that describes their non-standard functionalities and the hardware commands required to make use of these functionalities.

NOTE 1 This information can be made available on the manufacturers Web site, on machine readable media that are provided with the device, or through some other accessible mechanism.

NOTE 2 ISO/IEC 24756 provides a useful approach to specifying this information in a manner that can be electronically processed.

6 Requirements and recommendations on hardware-to-software interoperability

6.1 Responsibilities of ICT system manufacturers

6.1.1 Providing external control of system functions

Functions shall be available to be invoked via the accessibility services provided or supported by an ICT system to control major IT system functionalities from an external device.

NOTE 1 It is desirable that proprietary commands and new functions be standardized as soon as possible.

NOTE 2 Major IT system functionalities include controlling:

a) downloading and installing applications on the ICT;

b) starting the running of an application on the ICT;

c) operating the application on the ICT;

d) closing down the application on the ICT.

6.1.2 Information about system functionalities

Accessible information shall be provided by ICT systems, on which functionalities are accessible through standardized commands.

6.2 Responsibilities of operating system manufacturers

6.2.1 Provision of device drivers and APIs

An operating system shall provide a set of device drivers and APIs to support the needs of assistive technologies for providing alternative input and output mechanisms.

NOTE 1 It is recognized that each operating system has its own distinct needs in this area.

NOTE 2 It is recognized that additional accessibility services exist and can interoperate with some operating system APIs.

EXAMPLE 1 Drivers and APIs to the audio subsystem for generating text to speech for screen reading for blind users or people with print impairments; or for playing audio for people with speech impairments.

EXAMPLE 2 Drivers and APIs to the video device for screen magnification.

6.2.2 Information about device drivers and accessibility services

Operating system manufacturers shall provide sufficient information about the functionalities of their device drivers and accessibility services so that device manufacturers and software developers can properly interface with them. Where necessary, operating system manufacturers may allow device manufacturers and software developers to or create customized versions of device drivers.

NOTE It is desirable for OS manufacturers to present information on their APIs for potential international standardization.

6.2.3 Operating system installations

Operating System installation procedures shall ensure that all standard device drivers (i.e. mouse, keyboard, video, sound, printer, etc.) that are provided by the operating system are included in the installation.

NOTE Partial and selective installations of operating systems can lead to drivers that AT is counting on not being there.

EXAMPLE 1 While an Other Equipment Manufacturer (OEM) system that comes with its own pointing device with a custom mouse driver, the standard mouse driver might still needed.

EXAMPLE 2 Installing/upgrading the OS when USB is disabled in the BIOS can lead to a system that's missing USB or HID drivers, causing a problem if the user enables USB at a future date.

6.2.4 Device Driver Management

If the operating system allows deletion or replacement of device drivers, the operating system should allow assistive technology to register their use of device drivers. In such cases, the operating system should warn the user of the potential to create interoperability problems when deleting or replacing any of the device drivers that are used by assistive technology.

6.3 Responsibilities of device driver developers

6.3.1 Avoiding device driver incompatibilities

Specialized versions of device drivers shall conform precisely to the device driver specifications of the OS manufacturer and/or applicable international standards.

NOTE 1 This involves extending existing functionalities and commands rather than replacing them.

NOTE 2 This is especially critical for video drivers.

NOTE 3 It is important to keep performance enhancements in hardware, rather than drivers. This retains the possibility of compatibility of AT software with video displays.

6.3.2 Providing information about device driver functionalities

Developers of device drivers shall provide accessible information about which functionalities are available through standardized commands that are currently supported by their device driver.

NOTE 1 This information can be made available on the developers Web site, on machine readable media that are provided with the device, or through some other accessible mechanism.

NOTE 2 ISO/IEC 24756 provides a useful approach to specifying this information in a manner that can be electronically processed.

6.4 Responsibilities of device manufacturers

6.4.1 Providing information about specialized device drivers

Manufacturers of devices with functionalities, that require specialized device drivers, shall provide accessible information that describes any non-standard functionalities and any API extensions required to make use of these functionalities.

NOTE 1 This information can be made available on the manufacturers Web site, on machine readable media that are provided with the device, or through some other accessible mechanism.

NOTE 2 ISO/IEC 24756 provides a useful approach to specifying this information in a manner that can be electronically processed.

7 Requirements and recommendations on software-to-software interoperability

7.1 Responsibilities of all software developers

7.1.1 Extending existing functionality

New features/functions required by the software being installed shall append existing installed features, not replace them.

NOTE Installation of software that makes use of shared hardware or software components or shared applications can inadvertently overwrite existing settings and features used by AT.

7.1.2 Installation of new versions of application software

a) The user should be given the option of retaining or replacing the old version at the end of the installation procedure.

NOTE 1 This allows the user to determine whether or not any existing customizations have been rejected by the new version, due to incompatibilities between versions.

b) If a user chooses to retain the old version to check compatibility with AT, the user should be able at a later time to uninstall the old version.

NOTE 2 This can allow a user to retain the old version until the compatibility of the new version with AT has been tested.

c) Where new versions of an application are installed, customizations applied to the old version should be applied to the new version unless the installation procedure asks users to decide whether or not they wish to apply these customizations.

d) The installation procedure should alert the user to any customizations which cannot be applied to the new version.

7.1.3 Installation of alternate versions of application software

Where multiple alternate versions of an application are installed, customizations applied to one version shall not affect the capabilities of previously installed versions.

NOTE This can be handled by the installation software checking for existing settings before installing new settings over top of them and by each version of application software maintaining its own set of settings.

7.1.4 Limited ability to uninstall application software

The operating system should ensure that uninstalling application software does not remove software components shared with other applications.

7.1.5 Use of standard platform accessibility services

Software that provides user interface components shall either use the accessibility services provided by platform software or use other services to cooperate with AT, when such services allow the software to meet the accessibility provisions of this standard.

NOTE 7.1.6 and 7.1.7 provide further guidance on meeting this requirement.

EXAMPLE 1 A Windows application uses MSAA and IAccessible2 to enable interoperability with Windows assistive technology.

EXAMPLE 2 A Windows 7 application uses User Interface Automation to enable interoperability with assistive technology.

EXAMPLE 3 A UNIX application uses the UNIX Accessibility Framework to enable interoperability with assistive technology.

EXAMPLE 4 A software developer works directly with an AT vendor to develop other services to provide optimized access to the document object model of the application.

EXAMPLE 5 A third party works directly with an AT vendor to develop an interoperability interface where platform accessibility services do not yet exist.

7.1.6 Making functionality available to accessibility services

Software should make its functionality available via the accessibility services that might be found on the software platform(s) on which it runs.

NOTE 1 The use of calls to standard operating system libraries can make functionality related to these calls available to accessibility services in the Operating System.

NOTE 2 Web browsers and/or specific programming environments (e.g. Java) can act as a platform running upon another platform (i.e. the operating system). The use of standard functions on these platforms can make functionality available to the various accessibility services of the immediate platform and/or the operating system.

NOTE 3 Annex A provides information on the major accessibility services.

7.1.7 Making information available about user interface elements

Software shall provide AT with information about user interface elements, including but not limited to:

a) role, state(s), boundary, name, and description of the user interface element;

b) current value and any minimum or maximum values, if the user interface element represents one of a range of values;

c) text contents, text attributes, and the boundary of text rendered to the screen;

d) the relationship of the user interface element to other user interface element(s), including but not limited to:

   1) in a single data value, whether this user interface element is a label for another user interface element or is labelled by another user interface element,

   2) in a table, the row and column that it is in, including headers of the row and column if present,

   3) in a hierarchical relationship, any parent containing the user interface element, and any children contained by the user interface element.

NOTE This involves having interactive elements encoded in data operated on by the software associated with sufficient information to determine a role, state(s), name, and description for the interactive element, that the software can obtain as readily as it can obtain the interactive element itself.

7.1.8 Available actions on user interface elements

Software shall programmatically expose a list of available actions on a user interface element and allow assistive technology to programmatically execute any of those actions.

7.1.9 Focus and selection attributes of user interface elements

Software shall programmatically expose information necessary to track and modify: focus, text insertion point (where applicable), and selection attributes of user interface elements.

7.1.10 Changes related to user interface elements

Software shall programmatically expose notification of events relevant to user interactions, including but not limited to:

a) changes in the user interface element value,

b) changes in the name of the user interface element,

c) changes in the description of the user interface element,

d) changes in the boundary of the user interface element,

e) changes in the hierarchy of the user interface element.

7.1.11 Programmatic modifications of states, properties, values and text

Software shall allow states, properties, values, and text that can be modified by users to be programmatically modified by applications acting as assistive technology.

7.2 Responsibilities of operating system and platform software developers

a) Operating system software shall provide access to a set of accessibility services that enable applications running on the platform to interact with assistive technology sufficient to enable compliance with provisions 7.1.5 through 7.1.10.

b) Software toolkits and platform software shall support provisions 7.1.5 through 7.1.10 either by making the underlying platform accessibility services available to their client software, or using other services to cooperate with assistive technologies on the underlying platform.

8 Support of assistive technology

8.1 Provision of AT-specific documentation

AT-specific documentation related to ICT functional units (including peripherals, devices, computers, and software components) should be a part of the end-user documentation of the ICT functional unit.

NOTE Given the evolving nature of this information, this requirement can be satisfied by the product documentation pointing to a web site where the latest information is available.

EXAMPLE 1 An appendix to the ICT documentation lists which AT products have been tested with the ICT product, and any special considerations that might need to come to the user's attention.

EXAMPLE 2 The product documentation provides information on how to configure settings so that it is optimized to work with a particular screen reader.

8.2 Provision of accessible help

Help files should meet standards for accessible electronic content.

8.3 Avoiding monopolizing devices

Care should be taken not to monopolize the ICT's shared devices to the extent that they are not available to other software, especially AT.

NOTE 1 This is especially relevant for sound devices.

NOTE 2 Multiple channel sound technology can be used to ensure other sound channels are available to AT.

NOTE 3 AT applications often require the sound device. Applications which take exclusive control of the Sound Device can block sound output required by AT.

9 Expectations of assistive technology

9.1 AT responsibilities regarding the functional units they represent / replace

AT shall be responsible to comply, to the extent possible, with requirements of the functional units that they replace or assist.

9.2 Utilizing platform accessibility services

Assistive technology shall utilize platform accessibility services, to the extent that the information is appropriate to the AT function, in order to provide the alternative user interface(s) to individuals with disabilities.

NOTE AT sometimes needs to go beyond what is in the platform accessibility services.

Annex A
(informative)

Survey of accessibility application programming interfaces (accessibility APIs)

A.1 Introduction

A.1.1 Purpose

This annex provides an introduction to and survey of six (6) major accessibility API platforms that are currently used in industry and elsewhere.

In addition to identifying the APIs, each API is discussed in terms of the following four (4) primary topics:

a) Platforms

b) Versions

c) Features

d) Compatibility and relationships

This annex makes no judgments or recommendations about which API should be used to achieve the best interoperability. Rather, it provides the necessary information to allow developers to investigate the most appropriate implementations based upon their specific project requirements.

NOTE The information in this annex, while current at the date of publication, is expected to change over time. Readers of this annex are advised to obtain the latest information on applicable APIs directly from the sources of those APIs.

A.1.2 Identification information

Each accessibility API is identified with:

a) "Name" is used in this annex to refer to a given API. The name is used to commonly recognize individual APIs and to distinguish between those APIs. There is no guarantee that the name used is the official or current name being used by the creators, developers, and/or owners of the API.

b) "Source" is used to identify the organization that is responsible for developing and/or owning the API. This includes the name and contact information of the source. Where possible, a current Web address is included to help the reader to find additional information.

A.1.3 Platform information

Basic platform information for all APIs includes:

a) "Platform(s)" which identify the various platforms where the API can be used. This can include different operating systems, different versions of the same operating system, or other major differences that are implementation dependent. It is also used to identify any major differences between implementations on different platforms.

Additional platform information that is included where appropriate includes:

b) "User Agent" is used to identify various web content rendering software (e.g., web browsers) where the API may be supported.

c) "Application" is used to identify various user applications where the API may be supported.

d) "Library" is used to identify development libraries where the API can be used in the development of applications, etc.

A.1.4 Version information

Basic version information for all APIs includes:

a) "Version(s)" is used to identify the current and other currently active versions of the API. While it is recognized that this information will become dated, it is useful to serve as a frame of reference. It is also used to identify any major differences between active versions.

Additional version information that is included where appropriate includes:

b) "Native" refers to an API that, when compiled and run, is not under the control of an intermediary runtime infrastructure (e.g., virtual machine, CLR). Native code compiles directly to machine code, and the developer is responsible for virtually all aspects of programming constructs (e.g., pointers, freeing memory, etc.).

c) "Managed" refers to an API that, when compiled and run, is under the control of an intermediary runtime infrastructure, like a virtual machine. Microsoft's Common Language Runtime (CLR) and the Java Virtual Machine (JVM) are examples of runtime infrastructures. Managed code is compiled into an intermediate language construct (e.g., byte code) and the runtime infrastructure handles the compilation into machine code. The runtime infrastructure handles programming constructs like memory management.

A.1.5 Feature information

Feature information for APIs in this informative annex is organized in terms of:

a) "Feature" is used to identify common, major features of the stated accessibility APIs. It is used in providing an objective basis for comparing the APIs discussed in this Annex A. It is possible that not all features are listed for a given accessibility API.

b) "Feature Example" is used to provide an example of a given common feature specific to the accessibility API being discussed.

c) "Feature Example Description" is used to describe, in more detail, the purpose and/or use of the feature example.

For each feature, the name used in the API feature description might not be the official name used by the API to describe the feature. Instead, the name chosen is used to commonly recognize the feature across accessibility APIs. Clauses within Annex A describing a particular accessibility API will use these names to structure the features that are claimed to be provided by the API, and provide the official name of the feature in the API only as further information.

The description of features is intended to provide a generic description of functionality and, as such, does not favour any individual API. The discussion of features describing particular APIs is limited to identifying whether or not the given API has some form of the functionality of the feature. It does not include judgment on the quality of this functionality, or compare it to the functionality of other accessibility APIs. Quality determinations are outside the scope of Annex A.

Common features that are found in accessibility APIs include:

a) "UI Elements": Specialized user interface (UI) objects, graphical or otherwise, for accessible content

b) "Relations": Relations or associations between a given accessible object and other accessible objects

c) "Roles": Specialized roles for accessible objects often indicating their behaviour

d) "Interaction": Specialized interaction with an accessible object or between accessible objects

e) "Communication": Mechanisms to obtain information from the UI and provide information to the UI.

A.1.6 Compatibility and relationship information

"Compatibility" identifies how a diverse set of AT and IT systems and applications work together via APIs to create a solution for a target customer. Compatibility might occur directly or through a bridging technology that allows communication between two APIs.

"Relationships" identifies whether or not the stated accessibility API is known to be directly or indirectly related to other accessibility APIs, either through design or influence.

A.2 UI Automation (UIA) accessibility API

A.2.1 Introduction to the UI Automation accessibility API

UI Automation is the API in the Microsoft Accessibility Framework. UI Automation provides programmatic access to many user interface elements, enabling assistive technology (AT) products to provide information about the UI, and to manipulate the UI by means other than common standard input.

Source information on the UIA API is contained in Table A.1.

Table A.1 — Source information for the Windows UIA accessibility API

Name	Source
UI Automation (UIA)	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399 +1 (800) 642-7676 http://msdn.microsoft.com/en-us/accessibility/

NOTE The ISO/IEC number 13066-2 has been reserved for a future Technical Report on the Windows UIA accessibility API.

A.2.2 Platforms for the UI Automation accessibility API

UI Automation is available to developers through both native and managed APIs. The lower level, native APIs are currently the predominant set of APIs found on various Windows platforms. The managed APIs, available on all operating systems that support the .NET Framework, can access features in the native APIs through interoperability technologies (e.g., COM Interop). More information on the native APIs can be found at

http://msdn.microsoft.com/en-us/library/ms726294.aspx

and the managed APIs at

http://msdn.microsoft.com/en-us/library/ms753388.aspx.

A bridge is being developed to allow UI Automation applications to interoperate with UNIX efforts such as the UNIX Accessibility Toolkit (ATK).

Platform information on the UI Automation API is contained in Table A.2.

Table A.2 — Platform information for the UI Automation accessibility API

Platform	Versions
Microsoft Windows	Windows Vista, Windows XP, Windows Server 2003 and Windows Server 2008 (managed APIs are dependent on having at least the .NET Framework 3.0 installed. The current version of the .NET Framework is 3.5).
Linux	Through an adapter (or bridge) currently being developed by the Mono project. See http://www.mono-project.com/Accessibility for current platforms where Mono is available.

A.2.3 Versions of the UI Automation accessibility API

Developers can utilize either the managed UI Automation API or the native UI Automation API; the native API is prominent between the two.

The managed UI Automation API is a part of the overall .NET Framework. It was included starting in version 3.0 of the .NET Framework. The current version of the .NET Framework is 3.5.

The native UI Automation API (provider) is included as part of the Windows Vista and Windows Server 2008 SDK. The native UI Automation API is also distributed with the .NET Framework. The native UI Automation client API is currently being developed, and will be available for the Windows platform in the future.

A.2.4 Features of the UI Automation accessibility API

The main features of the UI Automation API are identified in Table A.3.

Table A.3 — Feature information for the UI Automation accessibility API

Feature	Feature Example(s)	Feature Example(s) Description
Communication	Provider APIs	A set of interface definitions that provide information about UI elements.
Communication	Client APIs	A set of types for managed code that enables client applications to obtain information about the UI.
UI Elements	Automation Elements	Represents a portion of the user interface (UI) in the logical hierarchy of UIA. An automation element may represent a document piece or an individual control part of the UI.
Relations	Automation Element Properties	Describes UIA properties that represent relationships among other elements (e.g. ControllerFor and LabeledBy properties).
Roles	Control Types	Indicates what kind of control a particular element represents (e.g., button).
Interaction	Control Patterns	Describes attributes and functionality of a UI element (e.g. can an element be check toggled?)
Interaction	Text Services Framework	Allows advanced user text interactions (e.g. entering or manipulating sentences)
Interaction	Text Patterns	Allows access to rich-text information.

A.2.5 Compatibility and relationships of the UI Automation accessibility API

A.2.5.1 Compatibility and relationships with Microsoft Active Accessibility (MSAA)

Before UI Automation, Microsoft Windows accessibility was supported by Microsoft Active Accessibility (MSAA). Since there are still MSAA based applications in existence, bridges are used to allow communication between UI Automation and MSAA applications. So information can be shared between the two APIs, an MSAA-to-UI Automation Proxy and UI Automation-to-MSAA Bridge were developed. The former is a component that consumes MSAA information and makes it available through the UI Automation client API. The latter enables client applications using MSAA access applications that implement UI Automation.

A.2.5.2 Compatibility and relationships with the UNIX Accessibility Toolkit (ATK)

While not completed yet, there was an announcement that an open source adapter (i.e., bridge) will be developed allowing the UI Automation framework to work with Linux accessibility efforts, including the UNIX Accessibility Toolkit (ATK). This would allow UI Automation to be used beyond Windows. There is more information at http://www.mono-project.com/Accessibility.

A.2.5.3 Compatibility and relationships with Accessible Rich Internet Applications (ARIA)

The UI Automation AriaRole and AriaProperties property can provide access to the ARIA attribute values corresponding to an HTML element (which can be exposed as an automation element by user agents like web browsers). General mapping from ARIA attributes to both UI Automation and MSAA properties is also available.

A.3 IAccessible2 Accessibility API

A.3.1 Introduction to the IAccessible2 accessibility API

The IAccessbile2 accessibility API is an alternative to UI Automation and to the UNIX Accessibility Toolkit. IBM initially developed IAccessbile2. It was donated to the Free Standards Group (FSG) in late 2006; the FSG is now part of the Linux Foundation. IAccessible2 leverages the work of Microsoft Corporation's work on the Microsoft Active Accessibility (MSAA) while also providing assistive technology (AT) access to various types of applications, including rich Internet applications (RIA) through an API that exposes the functionality of the Accessible Rich Internet Application (ARIA) markup specification.

Source information on the IAccessible2 accessibility API is contained in Table A.4.

Table A.4 — Source information for the IAccessible2 accessibility API

Name	Source
IAccessible2	The Linux Foundation 1796 18th St. Suite C San Francisco, CA 94107 USA +1 (415) 723-9709 http://www.linuxfoundation.org/en/Accessibility/IAccessible2

NOTE The ISO/IEC number 13066-3 has been reserved for a future Technical Report on the IAccessbile2 API.

A.3.2 Platforms for the IAccessible2 accessibility API

IAccessible2 is an open specification. IAccessible2 originally provided an alternative to Microsoft's work on the Microsoft Active Accessibility specification (MSAA), so it has its root of support on Microsoft Windows. IAccessible2 has many interfaces that are similar to those provided by the UNIX Accessibility Toolkit. Therefore, Windows applications that implement IAccessible2 can be easily ported to run on UNIX.

Platform information on the IAccessible2 API is contained in Table A.5.

Table A.5 — Platform information for the IAccessible2 accessibility API

Platform	Versions
Windows	Versions of Windows supported by Mozilla Firefox, for example, which currently implements IAccessible2.

Some of the specific applications that support IAccessible2 are noted in the Table A.6. A more complete list is located at http://www.linuxfoundation.org/en/Accessibility/IAccessible2/SoftwareDirectory.

Table A.6 — Software that supports the use of the IAccessible2 accessibility API

Software (User Agent/Library/Application)	Versions
Firefox	Mozilla's open source web browser. IAccessible2 is implemented in version 3 for Windows.
Lotus Symphony	An office productivity suite including a word processor, spreadsheet and presentation program. IAccessible2 is implemented in version 1.2 for Windows..

A.3.3 Versions of the IAccessible2 accessibility API

The current reference version of the IAccessible2 accessibility API is 1.02. There are no previous versions that are currently supported.

A.3.4 Features of the IAccessible2 accessibility API

The main features of the IAccessible2 API are identified in Table A.7.

Table A.7 — Features of the IAccessible2 accessibility API

Feature	Feature Example(s)	Feature Example(s) Description
UI Elements	Text Controls	Presents textual information on the display (e.g., on buttons). There are interfaces for both non-editable and editable text.
UI Elements	Tables	Provides access to a two-dimensional table. While the range of valid coordinates is implementation dependent, the indexing begins at 0,0.
UI Elements	Hyperlinks and Hypertext	Represents hyperlinks associated with a single substring of text or a single non-text object, and represents hypertext in a document that references another document.
Relations	Accessible Object Relations	Through a single interface and some constants, provides access to an accessible object's set of relations (e.g., relation types and targets).
Roles	Roles	Through an enumerator, defines a set of accessible roles of objects implementing the IAccessible2 interface (e.g., footnote).

The description of the programming interfaces describing the above features are at http://accessibility.freestandards.org/a11yspecs/ia2/docs/html.

A.3.5 Compatibility and relationships of the IAccessible2 accessibility API

A.3.5.1 Compatibility and relationships with the UI Automation accessibility API

Access is available to the native IAccessible2 accessible object tree from UI Automation, and vice-versa.

A.3.5.2 Compatibility and relationships with Microsoft Active Accessibility (MSAA):

IAccessible2 is an extension of the MSAA interfaces, specifically the IAccessible interface found in MSAA.

A.3.5.3 Compatibility and relationships with the UNIX Accessibility Toolkit (ATK):

Many of the IAccessible2 interfaces are based directly off the ATK interfaces, from name to functionality, and the header files derive directly from the OpenOffice.org UNO Accessibility API, which itself is a derivative of the Java Accessibility API.

A.3.5.4 Compatibility and relationships with Accessible Rich Internet Applications (ARIA)

The interfaces of IAccessible2 have a similar look to that of ARIA, allowing the mapping of web (e.g., web 2.0) accessible applications to, for example, Windows accessible applications.

A.4 UNIX Accessibility Toolkit (ATK) and Assistive Technology Service Provider Interface (AT-SPI)

A.4.1 Introduction to the UNIX Accessibility Toolkit and Assistive Technology Service Provider Interface (AT-SPI)

The UNIX Accessibility Toolkit (ATK) describes a set of open source accessibility interfaces that user interface components must implement in order for them to be operable via assistive technologies. Assistive technologies receive information from applications via the open source accessibility inter-process communication layer named the Assistive Technology Service Provider Interface (AT-SPI) which simply providing the information contained within ATK to applications and/or assistive technologies running in a separate process from the application that is being accessible.

UNIX ATK and AT-SPI were developed in parallel with development of the open source GNOME 2 graphical desktop for UNIX systems. Since the start of this work in 2001, Sun Microsystems, Inc. staff have been the maintainers of these APIs and also the primary staff implementing these APIs on the GNOME GTK+ libraries, as well as in applications for UNIX such as the open source GNOME help system, the Evolution e-mail and calendar application, the UNIX edition of the Mozilla family of applications (Firefox, Thunderbird, etc.), and StarOffice/OpenOffice.org. Sun staff are also the maintainer of Java Access Bridge to GNOME, which bridges the Java Accessibility API information to UNIX AT-SPI.

Source information on the UNIX Accessibility Toolkit is contained in Table A.8.

NOTE The ISO/IEC number 13066-4 has been reserved for a future Technical Report on the UNIX Accessibility Toolkit (ATK) and Assistive Technology Service Provider Interface (AT-SPI).

Table A.8 — Source information for the UNIX ATK/AT-SPI

Name	Source
Accessibility Toolkit	The GNOME Project / The GNOME Foundation PO Box 101 Groton, MA 01450 USA http://www.gnome.org http://library.gnome.org/devel/atk/ http://projects.gnome.org/accessibility/

NOTE The ISO/IEC number 13066-6 has been reserved for a future Technical Report on the UNIX Accessibility Toolkit (ATK).

A.4.2 Platforms for the UNIX Accessibility Toolkit and Assistive Technology Service Provider Interface

The interfaces of the UNIX Accessibility Toolkit (ATK) are toolkit-independent. Thus, implementations of the UNIX Accessibility Toolkit can be (and have been) written for a variety of user interface component sets.

An ATK implementation is included by default in the GNOME Desktop called GAIL (GNOME Accessibility Implementation Library). GAIL maps to the GTK widget toolkit, also included in the GNOME Desktop. The GNOME Desktop runs on a variety of UNIX distributions, and has been compiled on Windows.

Platform information on the UNIX ATK/AT-SPI is contained in Table A.9. Because of its toolkit approach, it is consistent with all current versions.

Table A.9 — Platform information for the Linux Accessibility Toolkit

Platform
Fedora (Linux)
Foresight Linux
Gentoo (Linux)
Mandriva (Linux)
openSuse (Linux)
Ubuntu (Linux)
FreeBSD
Solaris
OpenSolaris

ATK is either implemented directly by user interface component sets, or implemented via the use of a bridge that bridges some other accessibility API to ATK or AT-SPI. Work is underway to bridge the KDE/Qt library to ATK.

The main user interface components of the ATK are identified in Table A.10.

Table A.10 — Use interface component sets that implement ATK

User Interface Component set / library	Versions
GTK+	All versions since GNOME 2
XUL	All versions since version 1.9
UNO	All versions since version 2.0 either via the Java Accessibility API and the gnome-java-bridge or directly implementing ATK
Java Foundation Classes (aka “Swing”)	All versions, via the gnome-java-bridge
EWT – Oracle library for creating Java applications	All versions, via the gnome-java-bridge
JEWT – Oracle library for creating Java applications	All versions, via the gnome-java-bridge
WAI ARIA	The emerging standard (not yet at version 1.0) via the Firefox web browser for UNIX as of Firefox version 3.

As ATK is implemented on the main user interface component libraries of the GNOME desktop, as well as a number of user interface libraries commonly used in UNIX environments, a very large number of applications support ATK and are accessible via ATK/AT-SPI by the assistive technologies available for graphical UNIX environments.

Examples of key applications that support the ATK are identified in Table A.11.

Table A.11 — Some key applications that support ATK/AT-SPI

Software Application	Versions
Firefox	All versions since version 1.5
StarOffice / OpenOffice.org	All versions since OpenOffice.org 2.0 / StarOffice 8, either via the Java Accessibility API and the gnome-java-bridge or directly implementing ATK
Lotus Symphony	An office productivity suite including a word processor, spreadsheet and presentation program. IAccessible2 is implemented in version 1.2.

The GNOME desktop includes several assistive technologies and developer tools that utilize ATK/AT-SPI. All of the platforms listed in the table above either include by default the assistive technologies and tools listed below, or make them available as a free-to-install option.

Examples of AT and tools that utilize the ATK are identified in Table A.12.

Table A.12 — Assistive technologies / tools that utilize ATK/AT-SPI

Assistive technology / tools	User need	Versions
Orca screen reader/magnifier	Blindness via speech & Braille. Low vision via magnification	All versions
GOK – GNOME On-screen Keyboard	Wide range of severe physical impairments, including switch & dwell selection (compatible with head trackers & USB switches)	All versions
Dasher alternate text entry system		All versions
Accerciser test tool	Tool for examining / testing applications via ATK/AT-SPI. Includes a scripting interface for creation of highly sophisticated tests.	All versions
at-poke test tool	Tool for examining / testing applications via ATK/AT-SPI.	All versions (largely superseded by Accerciser)
Glade	Tool for creating GTK+ based GUIs	All versions since version 2.4

A.4.3 Versions of ATK/AT-SPI

The latest, stable versions of both ATK and AT-SPI are 1.26. However, past stable versions are still in use and are available; they include 1.9.0, 1.10.3, 1.11.4, 1.12.4, 1.18.0, 1.20.0, 1.22.0 and 1.24.0.

A.4.4 Features of the UNIX Accessibility Toolkit

The UNIX Accessibility Toolkit contains basic API calls that are used in applications to ensure compatibility with assistive technologies. Also, included are more advanced API calls to allow, for example, development of custom widgets that are accessible.

The main features of the UNIX Accessibility Toolkit (ATK) are identified in Table A.13.

Table A.13 — Features of the UNIX Accessibility Toolkit

Feature	Feature Example(s)	Feature Example(s) Description
Relations	Relations	Describes to an assistive technology how one user interface element (e.g., a label) is related to another (e.g. a text entry field). Note: this does not include hierarchy relationships, which are conveyed via the Context interface which all user interface elements provide.
Relations	Context	Contains the core set of accessibility information about every user interface element, including hierarchical relationships. This is further defined below under “UI Elements”.
Communication	State	Describes an accessible component’s state (e.g., checked). In ATK, user interface elements typically are in multiple States, and ATK will provide a full “State set”
Communication	Component	Describes information about the boundary of the user interface, provides for “hit testing”, and also programmatic movement of the element.
Interaction	Action	Enumerates and describes a set of actions that encompass they ways an object can be interacted with programmatically (but see value, text, and hypertext below)
Interaction	Value	Describes the current, minimum, and maximum values of a user interface element, as well as provides an interface for setting that value (as distinct from the Action interface).
Interaction	Selection	Describes the objects that are selectable/selected inside the user interface element (e.g. Items in a list box), and provides an interface for setting/clearing the selection programmatically (as distinct from the Action interface).
Interaction	StreamableContent	Provides an interface for retrieving and parsing the underlying stream of information behind a user interface element, via the standard UNIX streams interface (e.g. the “text/html” stream behind a web page) so that assistive technologies that way to have full access to the underlying data are able to get it.
UI Elements	Text	Accessible components that contain text content, both editable and non-editable. This interface exposes attribute information (font name, size, style, attributes) and provides a variety of ways of retrieving segments of text and discerning attribute runs. It also provides ways of retrieving and setting the text caret location and the text selection range.
	Table	Accessible components that contain row/column information.
	Links and HyperText	Allows accessible components to manipulate hyperlinks.