TECHNICAL REPORT

ISO/IEC TR 13066-6

First edition
2014-07-15

Information technology — Interoperability with assistive technology (AT) —

Part 6: Java accessibility application programming interface (API)

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

Partie 6: Interface de programmation d'applications (API) d'accessibilité Java

Reference number
ISO/IEC TR 13066-6:2014(E)

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.

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*.

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*

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

— *Part 3: IAccessible2 accessibility application programming interface (API)*

— *Part 4: Linux/UNIX graphical environments accessibility application programming interface (API)*

— *Part 6: Java accessibility application programming interface (API)*

Introduction

Assistive technology (AT) is specialized information technology (IT) hardware or software that is added to or incorporated within a system that increases accessibility for an individual. In other words, it is special purpose IT that interoperates with another IT product enabling a person with a disability to use the IT product.

Interoperability involves the ability to add or replace Assistive Technology (AT) to 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 describes the Java accessibility API that can be used as a framework to support software to software IT-AT interoperability on the multiple computing platforms. It also describes the Java Access Bridge for Windows – for enabling AT on Windows to interoperate with accessible Java applications on the Microsoft Windows platform – and the Java Access Bridge for GNOME – for enabling AT on UNIX and GNU/Linux platforms running the GNOME graphical desktop to interoperate with accessible Java applications on UNIX and GNU/Linux environments.

NOTE 1 GNOME is both a common and accessible graphical desktop for Linux / UNIX graphical environments, as well as an open source project delivering a collection of software libraries and applications. It was formerly an acronym meaning “GNU Network Object Model Environment”.

NOTE 2 The code examples contained in this document are illustrative in nature. With rare exception, they do not include error checking or exception handling, and should be treated more like pseudo-code than as cookbook templates that can use directly in applications or assistive technologies.

Information technology — Interoperability with assistive technology (AT) —

Part 6: Java accessibility application programming interface (API)

1 Scope

This part of ISO/IEC 13066 provides an overview to the structure and terminology of the Java accessibility API

It will provide:

— A description of the overall architecture and terminology of the API;

— Further introductory explanations regarding the content and use of the API beyond those found in Annex A of ISO/IEC 13066-1;

— An overview of the main properties, including of:

  — user interface elements;

  — how to get and set focus;

  — of communication mechanisms in the API;

  — a discussion of design considerations for the API (e.g. pointers to external sources of information on accessibility guidance related to using the API);

  — information on extending the API (and where this is appropriate);

  — an introduction to the programming interface of the API (including pointers to external sources of information).

  — an introduction to the Java Access Bridge for Windows and the Java Access Bridge for GNOME

It will provide this information as an introduction to the Java API to assist:

— IT system level developers who create custom controls and/or interface to them;

— AT developers involved in programming "hardware to software" and "software to software" interactions

2 Terms and Definitions

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

2.1
accessible object
a part of the user interface that is accessible by and exposes the Java accessibility API

Note 1 to entry An accessible object is represented by an object of the “AccessibleContext” Java class

2.2
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

[SOURCE: ISO/IEC 18012-1 Information technology – Home electronic system – Guidelines for product interoperability – Introduction, definition 3.1.1]

2.3
application software
software that is specific to the solution of an application problem

[SOURCE: ISO/IEC 2381-1, definition 10.04.01]

EXAMPLE A spreadsheet program is application software.

2.4
Assistive Technology
(AT)
hardware or software that is added to or incorporated within a system that increases accessibility for an individual

EXAMPLE Braille displays, screen readers, screen magnification software and eye tracking devices are assistive technologies.

[SOURCE: ISO 9241-171, definition 3.5]

Note 1 to entry 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).

2.5
class
a term from object oriented programming, also used in the Java programming language, denoting the definition/description of an object containing code (methods) and data (fields)

EXAMPLE All objects in object oriented programming belong to a class (e.g. a specific window object is an instance of the window class).

Note 1 to entry Much of the Java accessibility API consists of these class definitions, and implementations of the Java accessibility API are instances of these classes.

Note 2 to entry In objected oriented programming – and specifically in the Java programming language – classes can be “subclassed” (e.g. a dialog box class is a subclass of the more generic window class), and portions of the Java accessibility API are implemented as subclasses (e.g. AccessibleRole, AccessibleState, and AccessibleRelation are all subclasses of the more generic AccessibleBundle class).

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

[SOURCE: ISO/IEC 2381-1, definition 10.06.11]

2.7
information/communication technology
(ICT)
technology for gathering, storing, retrieving, processing, analysing and transmitting information

[SOURCE: ISO 9241-20, definition 3.4]

EXAMPLE A computer system is a type of ICT.2.13

2.8
interface
<general software> a shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical interconnections, signal exchanges, and other characteristics, as appropriate

[SOURCE: ISO/IEC 2381-1, definition 10.01.38]

2.9
interface
<Java programming language> in object oriented programming generally – and the Java language in particular – an interface is a set of public methods (and potentially public fields) that all objects implementing the interface must include

Note 1 to entry As with Java programming language classes that can be "subclassed", interfaces can be "subclassed" as well – and the result is called a "subinterface".

Note 2 to entry Much of the Java accessibility API is implemented as Java interfaces, and some of these as subinterfaces (e.g. the `AccessibleEditableText` interface is a subinterface of the more generic `AccessibleText` interface).

2.10
interoperability

the 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

[SOURCE: ISO/IEC 2381-1, definition 10.01.47]

2.11
inter-process communication
(IPC)
a mechanism by which different software processes communicate with each other – across process boundaries, runtime environments, and sometimes also computers and operating systems

2.12
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 1 to entry Although operating systems are predominantly software, partial hardware implementations are possible.

[SOURCE: ISO/IEC 2381-1, definition 10.04.08]

2.13
package
<Java programming language> a collection of class and interface definitions that are related to one another, and which are bundled together (into a package)

EXAMPLE The Java accessibility API is collected together into the core Java platform package `javax.accessibility.*` and the fully qualified name of each class or interface in this package begins with the package name, e.g. `javax.accessibility.AccessibleEditableText`.

2.14
runtime environment
a software environment that provides all of the resources necessary for software applications to run, yet is not itself an operating system

EXAMPLE 1 The Java runtime environment.

EXAMPLE 2 Adobe Flash player.

EXAMPLE 3 Microsoft Silverlight's runtime.

Note 1 to entry Virtual machines are type of runtime environment, which are explicitly emulating one or more specific sets of hardware.

2.15
software
all or part of the programs, procedures, rules, and associated documentation of an information processing system

Note 1 to entry Software is an intellectual creation that is independent of the medium on which it is recorded.

[SOURCE: ISO/IEC 2381-1, definition 10.01.08]

2.16
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. It also provides output mechanisms, allowing the system to produce the effects of the users' manipulation.

2.17
user interface element
user interface object
user interface component
entity of the user interface that is presented to the user by the software

[SOURCE: ISO 9241-171 definition 3.38]

Note 1 to entry User interface elements may or may not be interactive.

Note 2 to entry 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.

3 General Description

3.1 General Description

The Java accessibility API was developed by Sun Microsystems, Inc. as part of the Java Foundation Classes (along with the "Swing" user interface library), and was then folded into the Java Platform release 1.2. Work began in early 1997, based on requirements gathered from industry and assistive technology stakeholders, and with review and design feedback from many of these stakeholders as it matured. The 1.0 release shipped with the Java Foundation Classes release 1.0, in 1997, and was folded into the Java SE platform in January 1998. An initial implementation of the API in the "Swing" library was also part of that release.

The Java Accessibility API was born out of necessity – the first screen access techniques for graphical systems of the Macintosh and Windows reverse-engineered and hooked into the graphics rendering pipeline to build off-screen models for screen readers (also known as "screen scraping") – techniques that would not work in the Java environment. The Java accessibility API was the first comprehensive accessibility API (a "3rd generation accessibility API"). It provided support for everything that a screen reader needed, and is the progenitor of the UNIX accessibility API (described in ISO/IEC 13066-4) and the UNO accessibility API of Oracle Open Office (which is the basis for IAccessible2, described in ISO/IEC 13066-3). It was also a model for the WAI-ARIA specification (described in ISO/IEC 13066-5).

Because the Java platform is commonly running on top of some other, underlying operating system (e.g. Microsoft Windows or Solaris or GNU/Linux or Macintosh), and users with significant disabilities are using assistive technologies designed to work with the underlying operating system, a key facet of AT-IT interoperability on the Java platform is the use of a "bridge", which exposes the Java accessibility API outside of the Java platform, and to assistive technologies running on the underlying operating system. While it is certainly possible to use AT directly within the Java platform – and such technologies have been created – it is rarely used in this fashion.

3.2 Architecture

The Java accessibility API is based on the Java object model. The API itself is contained in a Java package (`javax.accessibility.*`), that is a core part of the Java platform. User interface components that are accessible must directly implement the `javax.accessibility.Accessible` interface (and for the rest of this document we will drop the package name and simply use the class or interface names, e.g. `Accessible`). When requested by an AT (or by a bridge on behalf of the AT), the accessible user interface component must then return an object that implements the Java accessibility API (the `Accessible.getAccessibleContext()` method). This object then handles all accessibility API calls on behalf of the user interface component. This architecture means that implementation of the Java accessibility API can either be implemented directly by that object, or be "delegated" to some other object or library.

In addition to this "delegation" model, the Java accessibility API is implemented as a core `AccessibleContext` object – containing all of the information common to every user interface component – and then a set of accessibility "sub-interfaces" or "specializations" which are implemented only as appropriate for the user interface component in question. For example, components containing text would implement the `AccessibleText` optional interface (and more specifically, the `AccessibleEditableText` optional interface if that text were editable). Components which take on one of a range of values would implement the `AccessibleValue` optional interface. etc.

The diagram below shows the Java Foundation Classes user interface component `javax.swing.JSlider`, and those aspects of the Java Accessibility API implemented for it:

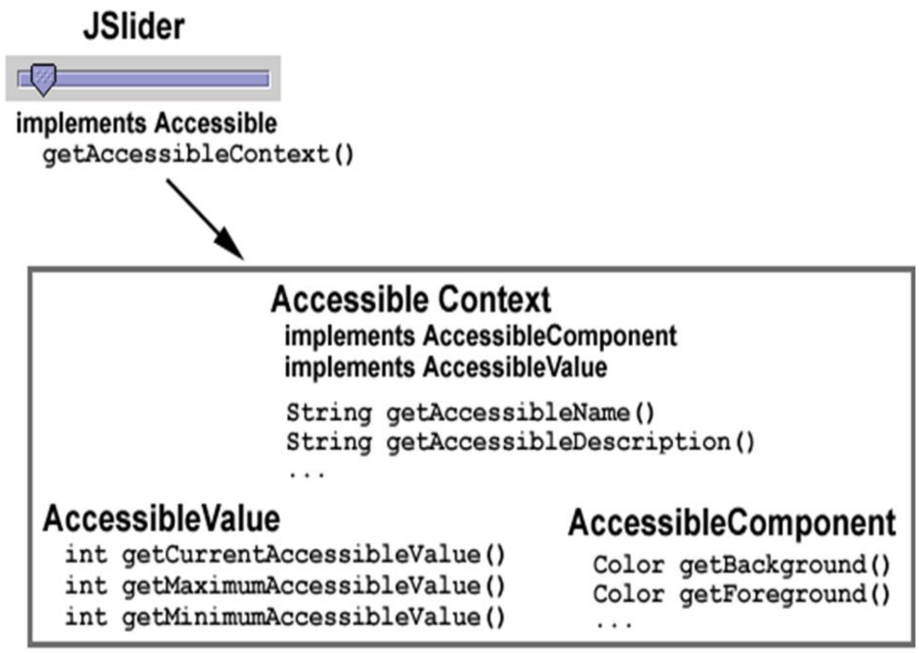
JSlider
implements Accessible
getAccessibleContext()
Accessible Context
implements AccessibleComponent
implements AccessibleValue
String getAccessibleName()
String getAccessibleDescription()
...
AccessibleValue
int getCurrentAccessibleValue()
int getMaximumAccessibleValue()
int getMinimumAccessibleValue()
AccessibleComponent
Color getBackground()
Color getForeground()
...

Figure 1 – Illustration of the accessibility interfaces implemented by the Swing JSlider component

4 Using the API

4.1 Overview

The Java accessibility API is contained in the `javax.accessibility.*` package. It consists of a core/tagging interface that all accessible user interface components must implement, a core accessibility class containing that portion of the Java accessibility API that all accessible user interface components must provide, an optional set of accessibility sub-interfaces, and then a set of helper classes. These are described below in the subclause

Applications can implement the Java Accessibility API in one of three ways:

a) They can use user interface components that have already implemented the Java accessibility API (such as those in the Java Foundation Classes or “Swing” library);

b) They can create one or more custom user interface components that derive from (or “subclasses”) an existing user interface component that already implements the Java accessibility API (such as a custom button that subclasses `javax.swing.JButton`, or a completely custom component that nonetheless subclasses `javax.swing.JComponent`);

c) They can create one or more custom user interface components “from scratch” and implement the necessary interfaces and methods to respond to the AT requests.

These distinct ways of implementing the API are described below.

4.2 Package javax.accessibility*

This package contains the Java Accessibility API.

The external entry point into the API is the interface Accessible, which contains the single method `getAccessibleContext()`. Every accessible user interface component must implement this interface, and must return a valid `AccessibleContext` when asked, which will implement the accessibility API on behalf of that component.

4.2.1 The AccessibleContext class

The AccessibleContext class contains the core implementation of the accessibility API. It is implemented as a class with a minimal existing implementation, with the expectation that it will be subclassed for each type of user interface component that uses it to implement the Java accessibility API. This class contains the core methods for traversing the UI hierarchy (several of which support requirements in ISO/IEC 13066-1, subclause 7.1.7(d)(1)):

— `getAccessibleParent()`

— `getAccessibleIndexInParent()`

— `getChild()`

— `getChildrenCount()`

basic information common to all user interface components (several of which support requirements in ISO/IEC 13066-1, subclause 7.1.7(a)):

— `getAccessibleName()`

— `getAccessibleDescription()`

— `getAccessibleRole()`

— `getAccessibleStateSet()`

— `getLocale()`

relationship information (which supports requirements in ISO/IEC 13066-1, subclause 7.1.7(d)(1)):

— `getAccessibleRelationSet()`

event tracking support (which support requirements in ISO/IEC 13066-1, subclause 7.1.10):

— `addPropertyChangeListener()`

— `removePropertyChangeListener()`

— `firePropertyChange()`

and several utility functions designed only to be called by implementations of the Java accessibility API:

— `setAccessibleName()`

— `setAccessibleDescription()`

— `setAccessibleParent()`