INTERNATIONAL STANDARD

ISO/IEC 24752-2

Second edition
2014-12-15

Corrected version
2016-04-01

Information technology — User interfaces — Universal remote console —

Part 2: User interface socket description

Technologies de l'information — Interfaces utilisateur — Console à distance universelle —

Partie 2: Description de "socket" d'interface utilisateur

Reference number
ISO/IEC 24752-2:2014(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*.

This second edition cancels and replaces the first edition (ISO 24752-2:2008), which has been technically revised.

This corrected version of ISO/IEC 24752-2:2014 incorporates the following modifications.

Clause 8: in the last line of paragraph 2 of 8.9.3, <relevant> has been changed to <write> and the reference has been amended to 7.4.3 instead of 8.9.3.

Clause 11: in the EXAMPLE in 11.3, the spaces contained in the values of the value attribute have been removed.

ISO/IEC 24752 consists of the following parts, under the general title *Information technology — User interfaces — Universal remote console*:

— *Part 1: General framework*

— *Part 2: User interface socket description*

— *Part 4: Target description*

— *Part 5: Resource description*

— *Part 6: Web service integration*

Introduction

This is the second edition of this part of ISO/IEC 24752. The main purpose of the revision is an alignment with recent developments in the Web service area, in particular with the new ISO/IEC 24752-6 on Web service integration, along with an overall simplification of the specified technologies.

A user interface socket is an abstract concept that, when implemented, exposes the functionality and state of a target in a machine-interpretable manner. A user interface socket is independent of any specific implementation platform.

A user interface socket contains variables, commands, and notifications, optionally structured in sets that may be nested in a hierarchical fashion. The variables include all of the dynamic data a user can perceive and/or manipulate, and may also include additional dynamic supporting data that is not presented to the user. Example variables include the volume of a television, the current floor of an elevator, or an internal variable representing the current state of a transaction that is used to control dynamic features of the interface. A command is a core function that a user can request a target to perform and that cannot be represented by a variable. The commands include all target functions that can be called by users. Examples include the 'search' command of an airline reservation system or the 'seek' command of a CD player. A user interface socket does not include commands for accessing the values of the variables. There are typically no commands that simply change the values of variables. An exception would be a 'reset' operation which puts the target into a specific state. The notifications are special states where normal operation is suspended, such as an exception state. Notifications are special states triggered by the target. Examples include an announcement made by a public address system in an airport, a clock alarm, or a response to invalid input for a field of a form.

A user interface socket specification is an XML document that uses the constructs defined in this part of ISO/IEC 24752 to describe a user interface socket.

See Annex A for an example user interface socket description.

NOTE Additional information is needed before the socket can be presented to a user, including natural language labels and help text associated with the elements of the user interface. This information is provided externally to the socket description. Resources reference socket elements using the socket's name (as given in the socket descriptions 'about' attribute value, see 6.2) and the element 'id' attribute (see 7.2, 9.2 and 10.2). Refer to ISO/IEC 24752-5 for further details.

Information technology — User interfaces — Universal remote console —

Part 2: User interface socket description

1 Scope

ISO/IEC 24752 is a multi-part International Standard that aims to facilitate operation of information and electronic products through remote and alternative interfaces and intelligent agents.

A user interface socket is an abstract user interface that describes the functionality and state of a device or service (target) in a machine-interpretable manner that is independent of presentation and input capabilities of a user interaction device. This part of ISO/IEC 24752 defines an Extensible Markup Language (XML)-based language for describing a user interface socket. The purpose of the user interface socket is to expose the relevant information about a target so that a user can perceive its state and operate it. This includes data presented to the user, variables that can be manipulated by the user, commands that the user can activate, and exceptions that the user is notified about. The user interface socket specification is applicable to the construction and adaptation of user interfaces.

2 Conformance

An XML file conforms to this part of ISO/IEC 24752 (i.e. is a user interface socket description) if it fulfils all of the following requirements:

a) it has an MIME type as specified in 6.1, if applicable;

b) it is coded in UCS (see 6.1);

c) its root element is the <uis:uiSocket> element (with uis representing the namespace "http://openurc.org/ns/uisocketdesc-2"), as specified in Clause 6;

d) it contains all required elements and attributes with their proper values, as specified in Clause 6;

e) if it contains recommended or optional elements or attributes with their values, these are presented as specified in Clause 6.

NOTE 1 Strict language conformance (i.e. no additional elements or attributes allowed) is not required because future versions of this part of ISO/IEC 24752 might add new elements, attributes, and values. Therefore, URC manufacturers are encouraged to implement their URCs so that unrecognized markup is ignored without failing.

NOTE 2 Target manufacturers who want to add manufacturer-specific information to a socket description beyond the elements, attributes, and values specified in this part of ISO/IEC 24752 can do so by externally providing (proprietary) resource descriptions that point into the structure of a socket description. Refer to ISO/IEC 24752-5 for details.

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 15836:2009, *Information and documentation — The Dublin Core metadata element set*

ISO/IEC 10646:2011, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 14977:1996, *Information technology — Syntactic metalanguage — Extended BNF*

ISO/IEC 24752-1, *Information technology — User interfaces — Universal remote console — Part 1: Framework*

ISO/IEC 24752-4, *Information technology — User interfaces — Universal remote console — Part 4: Target description*

W3C Recommendation: XML Path Language (XPath) 2.0 (Second Edition), W3C Recommendation 14 December 2010 (Link errors corrected 3 January 2011)1)

W3C Recommendation: XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition), W3C Recommendation 14 December 20102)

W3C Recommendation: XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 20043)

W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 20044)

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24752-1 and ISO/IEC 24752-4, and the following apply.

4.1
context element
element to which a dependency pertains

5 Relation to other standards

5.1 Relation to XML

This specification defines an extensible Markup Language (XML) based language. Markup in XML is case sensitive.

Tag names, and attribute names and values are not localizable, i.e. they are identical for all international languages. However, the text content between tags can be language specific. As with all XML based languages, white space characters immediately surrounding tags are non-significant.

This specification makes use of the XML namespaces concept to enable the import of element and attribute names defined elsewhere.

All element and attribute names used in this document with no namespace prefix are defined by ISO/IEC 24752 series and are part of the namespace with URI reference http://openurc.org/ns/uisocketdesc-2. If not defined as the default namespace, the namespace identifier 'uis' should be used.

Throughout this document, the following namespace prefixes and corresponding namespace identifiers are used for referencing foreign namespaces:

— dc: The Dublin Core Metadata Element Set namespace (http://purl.org/dc/elements/1.1/) (Element Set defined by ISO 15836);

1) File can be accessed in http://www.w3.org/TR/2010/REC-xpath20-20101214/

2) File can be accessed in http://www.w3.org/TR/2010/REC-xpath-functions-20101214/

3) File can be accessed in http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

4) File can be accessed in http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

— dcterms: The DCMI Metadata Terms namespace (http://purl.org/dc/terms);

— xsd: The XML Schema namespace (http://www.w3.org/2001/XMLSchema);

— xsi: The XML Schema Instance namespace (http://www.w3.org/2001/XMLSchema-instance).

For an XML Schema definition for the user interface socket description see Annex A.

5.2 XPath expressions

5.2.1 General

This specification uses XML Path Language (XPath) Version 2.0 for addressing elements within the socket. Specifically, XPath is used in describing dependencies between the elements of the socket.

XPath 2.0 syntax is used without XPath 1.0 compatibility.

5.2.2 Use of XPath 2.0 syntax and semantics

The ISO/IEC 24752 series uses the syntax and semantics of XPath 2.0, with the following additions and exceptions:

a) The XPath expressions shall be coded in UCS.

b) The static expression context (see 2.1.1 in XPath 2.0) shall be initialized with the following components:

1) "XPath 1.0 compatibility mode" shall be false.

2) The "statically known namespaces" are the namespace declarations that are in scope for he XML element that contains the XPath expression.

3) The "default element/type namespace" shall be the null namespace (which refers to types that are defined in the socket description, see Clause 11).

4) The "default function namespace" shall be the standard function namespace of XPath 2.0: http://www.w3.org/2005/xpath-functions.

5) The "in-scope schema definitions" shall only contain "in-scope schema types" with the following content: All types of namespace http://www.w3.org/2001/XMLSchema, as specified in section 2.5.1 of XPath 2.0; and the local types defined in a socket description's <xsd:schema> part (see Clause 11).

NOTE XPath 2.0 adds the following pre-defined types to the pre-defined types of XML Schema Definition Part 2: xsd:untyped, xsd:untypedAtomic, xsd:anyAtomicType, xsd:dayTimeDuration, xsd:yearMonthDuration.

6) The "in-scope variables" shall be empty.

7) The "function signatures" shall be the functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators, with exceptions as specified in 5.2.4 ; the constructor functions for all the atomic types in the "in-scope schema definitions"; and the additional functions defined in 5.2.5.

NOTE The following components of the XPath 2.0 static expression context are not used in this part of ISO/IEC 24752: "context item static type", "statically known collations", "default collation", "base URI", "statically known documents", "statically known collections", "statically known default collection type".

c) The dynamic expression context (see 2.1.2 in XPath 2.0) shall be initialized with the following components:

   1) The “context item” shall be the socket set or element that the XPath expression is specified for as dependency.

   2) The “variable values” shall be empty.

   3) The “function implementations” shall include implementations of the functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators; the constructor functions for all the atomic types in the “in-scope schema definitions”; and the additional functions as defined in 5.2.5.

   4) The “current dateTime” shall be the current time with local timezone of the URC, represented as a value of type xsd:dateTime.

   5) The “implicit timezone” shall be the local timezone of the URC.

   NOTE The following components of the XPath 2.0 dynamic expression context are not used in this part of ISO/IEC 24752: “context item”, “context position”, “context size”, “Available documents”, “Available collections”, “Default collection”.

d) There is no Data Model (XDM instance). Expressions and functions that refer to a data model instance shall not be used in socket descriptions. The context item expression (see 3.1.4 in XPath 2.0) shall not be used. Path expressions (see 3.2 in XPath 2.0) shall not be used. Node operations such as node comparison (see 3.5.3 in XPath 2.0), and the union, intersect and except operators (see 3.3.3 in XPath 2.0) shall not be used.

e) The evaluation of logical expressions (AND/OR) shall be strictly from left to right, and shall not evaluate the right operand if the result is already determined by the left operand. I.e. with an expression of the form “A and B”, B shall not be evaluated if A is false; and in the case of “A or B”, B shall not be evaluated if A is true. In addition, Boolean operations shall respect the “undefined” value (see 5.2.3).

f) The XPath 2.0 implementation shall be based on XML 1.0 and Namespaces in XML.

g) The XPath 2.0 implementation may support the namespace axis.

5.2.3 The undefined value

The ISO/IEC 24752 series adds the “undefined” value as a special value for all types from XPath 2.0 (see 5.2.2) and locally defined types (see Clause 11).

If any part of an XPath expression is undefined, the whole expression shall be undefined. This rule shall not apply, if, based on evaluation logic, the result of an expression is determined without evaluating any undefined part of it.

EXAMPLE The following expression will never evaluate to an undefined result. It yields true if the element with id ‘myvar’ is available and has the value 4, otherwise it yields false.

```
    uis:hasDefinedValue(‘myvar’) and uis:value(‘myvar’) eq 4
```

NOTE Implementations may vary as long as the described effect is warranted. For example, an error exception could be internally raised to signal that an XPath expression yields “undefined”.

5.2.4 XPath functions

The following XPath functions may be used:

— Functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators

— The constructor functions for all atomic types in the "in-scope schema definitions"

with the following exceptions:

— The function string() shall only be used with one argument.

— The function resolve-uri() shall not be used.

— The functions related to QName (section 11 of XQuery 1.0 and XPath 2.0 Functions and Operators), operators on NOTATION (section 13) and Functions and Operators on Nodes (section 14) shall not be used.

— The following context functions (section 13 of XQuery 1.0 and XPath 2.0 Functions and Operators) shall not be used: position, last, default-collation, static-base-uri.

The XPath 2.0 specific rules for implicit conversion between types apply.

5.2.5 Additional functions

5.2.5.1 General

The ISO/IEC 24752 series defines the following additional functions that may be used in expressing socket dependencies.

NOTE These functions are defined in the namespace "http://openurc.org/ns/uisocketdesc-2". A namespace prefix for this namespace (e.g. "uis") needs to be declared on any one of the XML elements containing the XPath expression. Note that the namespace prefix for "http://openurc.org/ns/uisocketdesc-2" must always be used for these functions since the default function namespace is "http://www.w3.org/2005/xpath-functions" (XPath 2.0 function namespace). Using the 'xmlns' attribute to declare a default XML namespace does not change the default function namespace.

5.2.5.2 xsd:anyType uis:value(string path)

This is a function which takes as its argument the path of a socket variable, command, notification, or an indexed component of it, and returns the current value of that variable, command, or notification (component).

NOTE 1 The type of the return value of uis:value() function is the same as the type of the socket element referred to by the argument 'path'. Note that, although the static return type is xsd:anyType, the dynamic type of the return value is always a more specific one (derived from the static type); it can be queried by the boolean operator 'instance of'.

The following syntax is defined for path (in Extended BNF notation, see ISO/IEC 14977):

— path = absolutePath | relativePath | shortcutPath;

— absolutePath = "/", { setPath, "/" }, elementPath;

— relativePath = ("./" | ("../", { "../" })), { setPath, "/" }, elementPath;

— shortcutPath = elementPath;

— setPath = *setId*, { "[", *setIndex* "]" };

— elementPath = normalElementPath | basicCommandPath | timedCommandPath | notifyPath;

— normalElementPath = *elementId*, { "[", *elementIndex*, "]" };

— basicCommandPath = *elementId*, { "[", *elementIndex*, "]" };

— timedCommandPath = *elementId*, { "[", *elementIndex*, "]" }, ["[ttc]"];

— notifyPath = *elementId*, { "[", *elementIndex*, "]" } ["[ttc]"];

A path shall be an absolute path, a relative path, or a shortcut path.

An absolute path shall be a slash-separated list of set ids and an element id, walking the path from the root element <uiSocket> (see 6.1) down to a socket element, with indices in square brackets wherever a set or the element has dimensions. An absolute path shall start with a slash character ("/") which stands for the root element.

A relative path shall have as context item (node) the socket element or set that the dependency is defined on. Relative paths that start with "./" have their context item as starting point for the subsequent path. Relative paths that start with "../" refer to the parent <set> of their context item as first segment in the path. Every subsequent "../" in the path refers to the next parent toward the root. No indices shall be specified for any dimensional set or element that is referred to by "./" or "../". At runtime, the missing indices (starting from the root) will be taken from the set/element owning the dependency, if not otherwise specified in the relative path. That means that relative paths occurring on dimensional sets or elements are inherently referencing elements with the same set of indices or a subset thereof (i.e. they are referring to the same "slice" of components).

A shortcut path shall omit set ids, and shall use only an element's id and indices, if any. It shall not have a leading slash character ("/"). A shortcut path shall not be used if the path includes a dimensional set.

setId is a placeholder for the 'id' attribute of a <set> element (which is an ancestor of the requested socket element). For dimensional <set> elements (i.e. <set> elements with a non-empty 'dim' attribute) *setIndex* is a placeholder for an index value of a <set> element, with the index value being compatible to the pertaining index type. <set> elements with no dimension shall have no *setIndex*.

elementId is a placeholder for the 'id' attribute of a <variable> (see 8.1), <command> (see 9.1), or <notify> element. For dimensional elements, an *elementIndex* shall be used as a placeholder for an index value of the element, with the index value being compatible to the pertaining index type. Elements with no dimension shall have no *elementIndex*.

— For <command> elements of type uis:timedCommand, the selector "[ttc]" may be added at the end of the path.

— For <notify> elements with a 'timeout' attribute, the selector "[ttc]" may be added at the end of the path.

The path argument shall be a string. The following characters shall be escaped if used as part of *setIndex* or *elementIndex,* as follows. '^' shall be used as the escape character.

— "[" shall be coded as "^["

— "]" shall be coded as "^]"

— "^" shall be coded as "^^"

NOTE 2 For hard-coded paths (i.e. the path is known at authoring time), the author can use a string literal that is enclosed in single (') or double (") quotes. For paths that are computed at runtime (e.g. when referencing variables as index values), the author can use valid XPath string operations (see 5.2) to concatenate a viable path string. See examples below.

The return value shall be the current value of the specified socket element (or its component if it is a dimensional element or has a dimensional set as ancestor). For command types that don't have state information (uis:voidCommand), an empty string shall be returned. For a command of type uis:basicCommand or uis:timedCommand and no *elementIndex*, the state (as string) of the command or its component shall be returned (see 9.3). For commands of type uis:timedCommand and an *elementIndex* of "ttc", the time to complete (as string in the xsd:duration format) of the command or its component shall be returned if it is currently defined, otherwise an empty string shall be returned. For a notification and no *elementIndex* specified, its state (as string) or the state of its component shall be returned (valid return values are "active", "inactive" and "stacked"). For a notification and an *elementIndex* of "ttc", a value indicating the remaining time to timeout (in seconds) or that of its component shall be returned.

uis:value(string path) shall evaluate to an undefined result for socket elements (or their components) that have an undefined value/state. In this case the whole expression (of which uis:value(path) is part of) shall have an undefined result.

EXAMPLE 1 A variable of type xsd:string with id "var" is nested inside two sets with ids "outerSet" and "innerSet". Neither the variable nor any of the nesting sets are dimensional. One can retrieve its value by either one of the following XPath expressions:

uis:value("/outerSet/innerSet/var")

uis:value("var")

EXAMPLE 2 A command of type uis:timedCommand with id "cmd" is nested in a set with id "setId". One can retrieve its state by either one of the following XPath expressions:

uis:value("/setId/cmd")

uis:value("cmd")

And one can retrieve the value of its "ttc" component by either one of the following XPath expressions:

uis:value("/setId/cmd[ttc]")

uis:value("cmd[ttc]")

EXAMPLE 3 A notification with id "notifyId" is nested in a set with id "setId". One can retrieve its state by either one the following XPath expressions:

uis:value("/setId/notifyId")

uis:value("notifyId")

EXAMPLE 4 A variable of type xsd:string with id "var" has one dimension with index type xsd:string. It is nested within a non-dimensional set element with id "setId".

One can retrieve the value of the component with index "alpha" by either one of the following XPath expressions:

uis:value("/setId/var[alpha]")

uis:value("var[alpha]")

And the value of the component with index "3*[2^3]" by either one of the following XPath expressions (note that "[", "^" and "]" need to be escaped):

uis:value("/setId/var[3*^[2^^3^]]")

uis:value("var[3*^[2^^3^]]")

EXAMPLE 5 Same as in example 5, but now retrieving the value of the component with an index value taken from another variable with id="index":

uis:value(concat("/setId/var[", uis:value("index"), "]"))

uis:value(concat("var[", uis:value("index"), "]"))

EXAMPLE 6 A command of type uis:timedCommand with id "cmd" has one dimension with index type xsd:integer. It is nested within a non-dimensional set element with id "setId". One can retrieve the "ttc" field of the command with index 0 by either one of the following XPath expressions:

uis:value("/setId/cmd[0][ttc]")

uis:value("cmd[0][ttc]")

EXAMPLE 7 A variable of type xsd:string with id "var" has two dimensions with index types xsd:integer and xsd:string. It is nested within a non-dimensional set element with id "setId". One can retrieve the value of the component with indices "3" and "none" by either one of the following XPath expressions:

uis:value("/setId/var[3][none]")