INTERNATIONAL STANDARD

ISO/IEC 24752-2

FIrst edition
2008-02-15

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 24752-2:2008(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 shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 24752-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 35, *User interfaces*.

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

— *Part 1: Framework*

— *Part 2: User interface socket description*

— *Part 3: Presentation template*

— *Part 4: Target description*

— *Part 5: Resource description*

Introduction

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, constants, commands and notifications. 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. Constants represent fixed or constant information known before runtime which may be presented to the user, such as a model number. 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.

An example user interface socket description is included as Annex B.

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 URI, see 6.2) and the element ids (see sections 7.2, 10.2 and 11.2). 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 to facilitate operation of information and electronic products through remote and alternative interfaces and intelligent agents.

A user interface socket is an abstract concept that describes the functionality and state of a device or service (target) in a machine interpretable manner. 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 or customization 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 has the MIME type specified in 5.3, if applicable, and
- its root element is the <uiSocket> element as defined in Clause 6.

An XML file does not conform to this part of ISO/IEC 24752 if it uses any element, attribute or value that is not part of this specification.

NOTE 1 Target manufacturers who want to add information to a socket description beyond the elements, attributes and values specified in this document 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.

NOTE 2 Future versions of this part of ISO/IEC 24752 might add new elements, attributes and values. Also, future versions might drop the policy of strict language conformance in favor of allowing for language extensions. Therefore, URC manufacturers are encouraged to implement their URCs so that unrecognized markup is ignored without failing.

3 Normative references

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

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

ISO/IEC 10646:2003, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

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

ISO 15836:2003, *Information and documentation — The Dublin Core metadata element set*

DCMI Metadata Terms, http://dublincore.org/documents/dcmi-terms/

IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, November 1996, http://www.ietf.org/rfc/rfc2046.txt

IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, January 2005, http://www.ietf.org/rfc/rfc3986.txt

W3C Recommendation: Extensible Markup Language (XML) 1.0 (Third Edition), 04 February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

W3C Recommendation: Namespaces in XML, World Wide Web Consortium 14 January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

W3C Recommendation: XML Path Language (XPath) 2.0, W3C Recommendation 23 January 2007, http://www.w3.org/TR/2007/REC-xpath20-20070123/

W3C Recommendation: XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Recommendation 23 January 2007, http://www.w3.org/TR/2007/REC-xpath-functions-20070123/

W3C Recommendation: XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

4 Terms and definitions

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

4.1
context element
element to which a dependency pertains

4.2
dependency
expression that defines a relationship between a property of a socket variable, constant, command or notify element and the values of other socket elements

4.3
notify
socket element that represents a notification

4.4
uniform resource identifier
URI
name or address that refers to a resource, as defined in IETF RFC 3986

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 this International Standard and are part of the namespace with URI reference http://myurc.org/ns/uisocketdesc. 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:

- xsd: The XML Schema namespace (http://www.w3.org/2001/XMLSchema);
- dc: The Dublin Core Metadata Element Set namespace (http://purl.org/dc/elements/1.1/) (Element Set defined by ISO 15836);
- dcterms: The DCMI Metadata Terms namespace (http://purl.org/dc/terms);
- 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

This International Standard uses the syntax and semantics of XPath 2.0, with the following additions and exceptions:

- The XPath expressions shall be coded in UCS.
- The static expression context (see section 2.1.1 in XPath 2.0) shall be initialized with the following components:
    - “XPath 1.0 compatibility mode” shall be false.
    - The “statically known namespaces” are the namespace declarations that are in scope for the XML element that contains the XPath expression.

- The "default element/type namespace" shall be the null namespace (which refers to types that are defined in the socket description, see 12).

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

- 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 <schema> part (see 12).

  NOTE 1 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.

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

- 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 2 The following components of the XPath 2.0 static expression context are not used in this International Standard: "context item static type", "statically known collations", "default collation", "base URI", "statically known documents", "statically known collections", "statically known default collection type".

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

  - The "context item" shall be the socket set or element that the XPath expression is specified for as dependency.

  - The "variable values" shall be empty.

  - 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.

  - The "current dateTime" shall be the current time with local timezone of the URC, represented as a value of type xsd:dateTime.

  - The "implicit timezone" shall be the local timezone of the URC.

  NOTE 3 The following components of the XPath 2.0 dynamic expression context are not used in this International Standard: "context item", "context position", "context size", "Available documents", "Available collections", "Default collection".

- 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 section 3.1.4 in XPath 2.0) shall not be used. Path expressions (see section 3.2 in XPath 2.0) shall not be used. Node operations such as node comparison (see section 3.5.3 in XPath 2.0), and the union, intersect and except operators (see section 3.3.3 in XPath 2.0) shall not be used.

- 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).

- The XPath 2.0 implementation shall be based on XML 1.0 and Namespaces in XML.
- The XPath 2.0 implementation may support the namespace axis.

5.2.3 The undefined value

This International Standard adds the "undefined" value as a special value for all types from XPath 2.0 (see 5.2.2) and locally defined types (see 12).

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

This International Standard defines the following additional functions that may be used in expressing socket dependencies.

NOTE These functions are defined in the namespace http://myurc.org/ns/uisocketdesc. 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://myurc.org/ns/uisocketdesc 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, constant, 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 ;

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

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

timedCommandPath = *elementId* , { "[" , *elementIndex* , "]" } , ["[state]" | "[timeToComplete]"] ;

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

An absolute path is 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 starts with a slash character ("/") which stands for the root element.

A relative path has 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 omits set ids, and uses only an element's id and indices, if any. It does 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 have no *setIndex*.

elementId is a placeholder for the 'id' attribute of a <variable>, <constant>, <command> 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 and <constant> elements have no *elementIndex*.

For <command> elements of type uis:basicCommand or uis:timedCommand, a selector may be added in square brackets at the end of the path. Allowed is for uis:basicCommand: "[state]"; and for uis:timedCommand: "[state]" or "[timeToComplete]". If the selector is missing for those types, "[state]" is assumed.

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 is 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 is returned. For a command of type uis:basicCommand or uis:timedCommand and an *elementIndex* of "state" or empty, the state (as string) of the command or its component is returned (see 10.3). For commands of type uis:timedCommand and an *elementIndex* of "timeToComplete", the time to complete (as string in the xsd:duration format) of the command or its component is returned if it is currently defined, otherwise an empty string is returned. For a notification its state (as string) or the state of its component is returned (valid return values are "active", "inactive" and "stacked").

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 constant of type xsd:string with id "const" is nested inside two sets with ids "outerSet" and "innerSet". Neither the constant nor any of the nesting sets are dimensional. (In fact, a constant can never be dimensional). One can retrieve its value by either one of the following XPath expressions:

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

EXAMPLE 3 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[state]")
uis:value("/setId/cmd")
uis:value("cmd[state]")
uis:value("cmd")

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

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

EXAMPLE 4 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 5 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 6 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 7 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 timeToComplete field of the command with index 0 by either one of the following XPath expressions:

uis:value(“/setId/cmd[0][timeToComplete]”)
uis:value(“cmd[0][timeToComplete]”)

EXAMPLE 8 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]”)
uis:value(“var[3][none]”)

EXAMPLE 9 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 1-dimensional outer set element with id “outerSet” and index type xsd:boolean, and a non-dimensional inner set element with id “innerSet”. One can retrieve the value of the component with indices “true” (for the outerSet element), “3” and “none” (for the var element) by the following XPath expression:

uis:value(“/outerSet[true]/innerSet/var[3][none]”)

EXAMPLE 10 Same as in example 9, but now using the values of three other variables (with ids “index1”, “index2” and “index3) as index values:

uis:value(concat(“/outerSet[“, uis:value(“index1”), ”]/innerSet/var[“, uis:value(“index2”), ”][“, uis:value(“index3”), ”]”))

EXAMPLE 11 This example illustrates the use of relative paths. The following minimal socket defines a 1-dimensional set with an index type of xsd:integer. For every set instance, a “power” variable and a “dimmer” variable is defined. The “dimmer” value can only be changed if the power of the pertinent light is on. (The relative path “../power” in the write dependency of the dimmer variable refers to the “power” variable, with the index for the parent set being the same as for the “dimmer” variable that contains the write dependency.)

```
<uiSocket name="http://example.com/lights/socket” id=”socket”

    xmlns="http://myurc.org/ns/uisocketdesc" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <set id="lights” dim=”xsd:integer”>
        <variable id=”power” type=”xsd:boolean” />
        <variable id=”dimmer” type=”dimmerType”>
            <dependency>
                <write> value(“../power”) </write>
            </dependency>
        </variable>
    </set>
    <xsd:schema>
        <xsd:simpleType name=”dimmerType” id=”dimmerTypeId>
            <xsd:restriction base=”xsd:integer”>
                <xsd:minInclusive value=”0” />
                <xsd:maxInclusive value=”10” />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:schema>
</uiSocket>
```


EXAMPLE 12 This example uses a relative path inside the 'relevant' dependency of a set. For each set instance, one variable instance (with id="isRelevant") inside the set instance is controlling whether the set instance is to be presented to the user or not.

```
<uiSocket name="http://example.com/relevantset/socket" id="socket"
    xmlns="http://myurc.org/ns/uisocketdesc" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <set id="set1" dim="xsd:integer">
        <variable id="isRelevant" type="xsd:boolean" />
        <variable id="someData" type="xsd:string" />
        <dependency>
            <relevant> value("./isRelevant") </relevant>
        </dependency>
    </set>
</uiSocket>
```

5.2.5.3 boolean uis:hasDefinedValue(string path)

This is a function which takes as its argument the path of a socket variable, constant or command and returns "true" if the current value of that variable or constant, or the current state of that command is defined, otherwise "false". For command types that don't have state information (uis:voidCommand), "false" is returned.

The argument path denotes an XPath expression that shall evaluate to a string.

NOTE Since XPath defines the 'or' and 'and' operators so that the right operand is not evaluated if the left operand pre-determines the result, one can check for undefined values/states by calling uis:hasDefinedValue(id) before calling uis:value(id).

EXAMPLE The following expression will never evaluate to an undefined value. It yields true if the command with id 'reset' has an undefined state or the state 'succeeded'.


```
not(uis:hasDefinedValue('reset')) or uis:value('reset') eq 'succeeded'
```


5.2.5.4 boolean uis:isNotifyActive()

This is a Boolean function with no argument; returns true if any notify element in the socket is active, and false if no notify element is active.

NOTE The XPath expression "not(uis:isNotifyActive())" can be useful to check for normal mode (i.e. no notification is active).

5.2.5.5 boolean uis:sessionForward(string type, string uri)

This is a function that states session forwarding (see ISO/IEC 24752-1). The value of the type argument is either "destructive" or "spawn". The value of the uri argument is the name (URI) of the socket that the URC is forwarded to. The return value is "true" if the target has sent the specified session forward event to the URC, otherwise "false". This function can be used in postconditions of socket commands (see 10.10.6), to provide a hint to the URC that the command may trigger a session forwarding.

5.3 MIME type

A user interface socket description shall have a MIME type of "application/urc-uisocketdesc+xml", if applicable.