INTERNATIONAL STANDARD

**ISO/IEC 24752-1**

Second edition
2014-12-15

# Information technology — User interfaces — Universal remote console —

## Part 1: General framework

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

*Partie 1: Cadre général*



Reference number
ISO/IEC 24752-1:2014(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.

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*, SC 35, *User interfaces*.

This second edition cancels and replaces the first edition (ISO/IEC 24752-1:2008), which has been technically revised.

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 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, along with an overall simplification of the specified technologies.

This part of ISO/IEC 24752 is one of a set of International Standards to facilitate operation of information and electronic products through remote and alternative interfaces and intelligent agents. The purpose of ISO/IEC 24752-1 is to facilitate the development and deployment of a wide variety of devices (from different manufacturers) that can act as universal remote consoles (URCs) for an equally varied range of target devices and services (targets), also from different manufacturers. It allows users to control any number of information and electronic products in their environment.

The targets include both devices and services. They can range from things as simple as light switches and thermostats to more complex items such as audio-visual equipment, home appliances, in-car electronics, web services, and any other devices or services that can be controlled electronically (or via information technology).

Targets can be in the same location as the individual who desires to control the target through the URC, or at any distance from the URC/user as long as there is some type of network connection between the URC and the target. This is possible since a URC provides the user with all of the necessary controls as well as the prompts and other information displayed by the target.

The URCs could be software running on common mainstream devices such as personal computing and information technology devices (e.g. computers, laptops, tablet computers, smartphones, cell phones, or other telecommunications devices). They could also be functions implemented in assistive technology devices, or they could be devices which were specially built to function as URCs. They could be devices which were built to function primarily as a remote console for a particular family of products (e.g. a remote console designed to be part of a home audio-visual system), but could also serve to control any other devices compatible with this part of ISO/IEC 24752. They are similar to the behaviour of universal remote controls today, except for the following:

a) they have much greater function and scope,

b) they synchronize with the target in both directions (i.e. they can display the current status of the target),

c) they do not need to be programmed by the user (since they will automatically discover devices that are controllable in a user's vicinity, discover the abstracted user interface of the targets, and present it in the way preferred by the user and their URC), and

d) they can be used out of sight of the product they are controlling.

The URCs could be all visual, all tactile, or all verbal in nature, or any combination thereof, because this International Standard specifies the content of a target user interface independently from the form in which it is presented. Thus, URCs could be designed that an individual could talk to and, through the URC, the user could have speech access to any compatible target listed above without any of these targets having any voice recognition or voice control functionality themselves. A person might, therefore, be able to say to their URC, "Record channel 12 and show me 'Law and Order'". Or they could lie in bed and say, "Set the alarm to 6:30 AM, start brewing the coffee at 6:00 AM, and now set the home security system to 'active'". Or, if one's spouse is already asleep, a person could pick up their smartphone or any other compatible URC device and accomplish these same tasks silently either by calling up control panels or by issuing the instructions in writing.

NOTE The URC framework does not provide the natural language control, but would provide all of the information and control necessary for control by a natural language processing URC.

Note that, although a URC implementation can involve hardware, requirements on this hardware such as safety and design requirements are not within the scope of this International Standard.

A more detailed overview of the URC framework is provided on the OpenURC Alliance's website, at http://openurc.org/urc-overview.[6]

# Information technology — User interfaces — Universal remote console —

# Part 1: General framework

## 1 Scope

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

This part of ISO/IEC 24752 defines a framework of components that combine to enable remote user interfaces and remote control of network-accessible electronic devices and services through a universal remote console (URC). It provides an overview of the URC framework and its components.



## 2 Conformance

### 2.1 URC

A conforming URC shall conform to the URC requirements as specified in Clauses 4 and 8.

Table 1 summarizes the requirements on URCs.

**Table 1 — Summary of URC requirements**

| Requirement ("A URC shall ...") | See subclause |
|---|---|
| Retrieve documents from a target, including recognition of MIME types | 4.2.3 |
| Interpret a target description so that it can identify a target and open a control session with one of its sockets | 4.2.4 |
| Support the invocation of a target's locator function | 4.2.5 |
| Support an open session request to a target | 4.3.2 |
| Support a URC close session event to a target | 4.3.5 |
| Support an abort session event from a target | 4.3.6 |
| Track connection status information from the underlying network (TUN) | 4.3.7 |
| Synchronize values of socket variables | 4.4.2 |
| Request invocation of a socket command, including support for local parameters and command state updates | 4.4.3 |
| Receive and acknowledge notifications, including support for stacking notifications and their states | 4.4.4 |
| Synchronize actual indices of socket sets and elements | 4.4.5 |
| Support the 'timeout' attribute on notifications | 4.4.7 |
| Provide at least one target-URC network link (see 7.2 for TUN requirements) | 4.5.1 |
| Support reception and updating of dynamic atomic resources at runtime | 4.5.2 |
| Provide a concrete user interface for a control session with a target's socket | 4.7 |
| Implement the security and privacy functions available from the implemented TUNs | 8.2 |

## 2.2 Target

A conforming target shall meet all requirements specified in Clauses 5 and 8.

Table 2 summarizes the requirements on targets.

**Table 2 — Summary of target requirements**

| Requirement ("A target shall ...") | See subclause |
|---|---|
| Have an instance identifier | 5.1.2 |
| Provide a fetch mechanism for its documents to be retrieved by URI, including support for MIME types (see NOTE below) | 5.1.3 |
| Provide exactly one target description with references to all socket descriptions, required resource sheets, and grouping sheet (see NOTE below) | 5.1.4 |
| Support locator functions | 5.1.5 |
| Provide one or more user interface sockets that collectively provide access to all of the functionality provided by the built-in user interface of the target | 5.2.2 |
| Inside a target's socket are the following:<br>— the *variables* shall include all of the dynamic data on the target socket's state a user can perceive and/or manipulate,<br>— the *commands* shall include all of the target functions that can be called explicitly or implicitly by users, and<br>— the *notifications* shall cover all exceptions that the target needs to inform the user about | 5.2.3 |
| Provide a user interface socket description for each of the target's sockets (see NOTE below) | 5.3 |
| Provide the required target resources in at least one natural language for the following:<br>— one grouping resource for every socket of the target,<br>— label resources (textual) | 5.4.6 |
| Support an open session request from a URC | 5.5.1 |
| Support a suspend session request from a URC | 5.5.2 |
| Support a resume session request from a URC | 5.5.3 |
| Support a close session event from a URC | 5.5.4 |
| Send an abort session event in case of user session abortion | 5.5.5 |
| Track connection status information from the underlying TUN network | 5.5.6 |
| Send a session forward event to the URC in case of session forwarding | 5.5.7 |
| Create and maintain a session between a socket and the URC after a successful open session request | 5.6.1 |
| Indicate to the URC the availability of socket elements at runtime | 5.6.3 |
| Synchronize the socket variables between the socket and the URCs that participate in a joint session with the socket | 5.6.5 |
| Support command invocation requests from a URC (including handling of local parameters) and synchronization of command states | 5.6.6 |
| Support propagation of notification states and, for custom-type notifications, embedded variables and commands, to the connected URCs, and acceptance of pertinent acknowledgments | 5.6.7 |
| Synchronize actual indices of socket sets and elements | 5.6.8 |
| Not rely on the URC doing the interpretation of socket element dependencies | 5.6.9 |

**Table 2** *(continued)*

| Requirement ("A target shall ...") | See subclause |
|---|---|
| Provide the following mechanisms with regard to user response timeouts:<br>— after a timeout extension return to the state of the task the user had reached prior to the timeout,<br>— use the 'timeout' attribute on notifications to represent timeout durations,<br>— note time out notifications in less than 10 seconds | 5.6.10 |
| Provide at least one target-URC network link (see 7.2 for TUN requirements) | 5.7 |

NOTE As an alternative to having the target provide these documents (target description, socket descriptions, required atomic resources and grouping), they may be provided separately as supplemental resources, if the target is a legacy product that already provides the necessary communication and control functionality through a networking platform (target-URC network).

## 3 Terms and definitions

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

**3.1**
**access key resource**
atomic resource that specifies a single character that can be used in combination with a URC-specific mechanism to move the focus to an element of an interface

Note 1 to entry: Note1 to entry: When an access key is bound to a command, entering the access key will activate the command.

**3.2**
**atomic resource**
resource that is used as an atomic entity in the construction of a concrete user interface

EXAMPLE Atomic resources include label resources, help resources, access key resources, keyword resources, and location descriptions.

Note 1 to entry: An atomic resource may be of any form, including text, images, sounds, animations, and video clips. See ISO/IEC 24752-5.

**3.3**
**atomic resource description**
resource description of atomic resources

Note 1 to entry: ISO/IEC 24752-5 specifies a format for atomic resource descriptions.

**3.4**
**command**
socket command
socket element representing a core function that a user can request a target to perform that cannot be achieved through the manipulation of the value of a single variable

EXAMPLE A 'reset' or 'submit' operation.

Note 1 to entry: See ISO/IEC 24752-2.

**3.5**
**command parameter**
socket command parameter
variable whose value is used for the execution of a command

Note 1 to entry: See ISO/IEC 24752-2.

**3.6**
**connection**
association established between functional units for data transmission

[SOURCE: American National Standard Dictionary of Information Technology (ANSDIT)]

**3.7**
**context of use**
use context
users, tasks, equipment (hardware, software, and materials), and the physical and social environments in which a product is used

[SOURCE: ISO 9241-11:1998, 3.5]

**3.8**
**control phase**
time period during which a URC and a target initiate maintain and terminate a control session between the URC and a specific target socket

**3.9**
**dependency**
expression that defines a relationship between a property of a socket variable, command, or notify element and the values of other socket elements

**3.10**
**device**
physical device with a built-in user interface that can also be controlled electronically

EXAMPLE light switches, thermostats, home appliances, audio-visual equipment, vending machines, and point-of-sale devices.

**3.11**
**dimensional socket element**
dimensional element
homogenous set of values pertaining to a socket element

Note 1 to entry: See ISO/IEC 24752-2.

**3.12**
**dimensional socket set**
repeating socket set
homogenous collection of sets with different indices

Note 1 to entry: See ISO/IEC 24752-2.

**3.13**
**discovery**
process by which a URC locates and connects to targets in its environment

**3.14**
**discovery phase**
time period during which a URC scans the environment for available targets and identifies their sockets

**3.15**
**dynamic atomic resource**
atomic resource that is provided by the target at runtime

**3.16**
**element**
fundamental logical unit of an XML document



 

**3.17**
**global parameter**
global command parameter
reference from a command to a variable that serves as an input or output parameter for the command

Note 1 to entry: See ISO/IEC 24752-2.

**3.18**
**grouping resource**
grouping
hierarchical structure of user interface socket elements or user interface implementation description elements in a top-down fashion that is provided externally to a socket description

Note 1 to entry: Syntax and semantics of grouping resources are defined in ISO/IEC 24752-5.

**3.19**
**grouping sheet**
file that contains grouping resources

Note 1 to entry: Syntax and semantics of grouping sheets are defined in ISO/IEC 24752-5.

**3.20**
**help resource**
atomic resource intended to be used to provide help to a user of a target

**3.21**
**input parameter**
variable whose value is read by the target before execution of a command, to affect the execution and its result(s)

Note 1 to entry: See ISO/IEC 24752-2.

**3.22**
**input-output parameter**
variable used as input and output parameter for the same command

Note 1 to entry: See ISO/IEC 24752-2.

**3.23**
**interface generator**
software that generates a user interface for a target that is appropriate for a known context of use

Note 1 to entry: In the context of ISO/IEC 24752, interface generation is typically based on a socket description, resource sheets, and grouping sheets.

**3.24**
**keyword resource**
atomic resource that specifies a keyword pertaining to a referenced element

**3.25**
**label resource**
atomic resource used to label, identify, or present an element in a user interface

EXAMPLE The label “John F Kennedy International Airport” could be used to present the value “JFK”, or the label “Destination” could be used to identify an input field where the user must enter a travel destination.

**3.26**
**local parameter**
input or output parameter that is attached to a command

Note 1 to entry: See ISO/IEC 24752-2.

**3.27**
**locator function**
locator
function of a target that can be invoked by the user and that helps the user to locate the target

EXAMPLE Audio functions such as a beep or bell, visual functions such as a flash, and direction based functions such as an "infrared ping" function.

**3.28**
**notification**
special state of a target in which normal operation is suspended

EXAMPLE An exception state.

Note 1 to entry: See ISO/IEC 24752-2.

**3.29**
**notify element**
socket notify element
socket element that represents a notification

**3.30**
**output parameter**
command result
variable whose value is updated by the target after execution of a command, to reflect a result of the execution

Note 1 to entry: See ISO/IEC 24752-2.

**3.31**
**resource**
object that is used as an entity or to support decision making in the construction of a concrete user interface

EXAMPLE Resources include user interface implementation description, resource sheet, and any kind of atomic resource such as label resources, help resources, access key resources, and keyword resources.

Note 1 to entry: See ISO/IEC 24752-5.

**3.32**
**resource description**
description of a resource in terms of its properties

Note 1 to entry: The format of a resource description is specified in ISO/IEC 24752-5.

**3.33**
**resource service**
service that provides resources from target manufacturers and any third parties such as URC manufacturers, beyond the target resources

Note 1 to entry: See ISO/IEC 24752-5.

**3.34**
**resource service description**
description of any reference to a resource service

Note 1 to entry: The format of a resource service description is specified in ISO/IEC 24752-5.

**3.35**
**resource sheet**
file that contains atomic resource descriptions

Note 1 to entry: The format of a resource sheet is specified in ISO/IEC 24752-5.



 

**3.36**
**resource-URC network**
**RUN**
network connecting the URC to sources of supplemental resources and user interface implementation descriptions (UIIDs)

Note 1 to entry: It may employ any networking and connection technologies.

**3.37**
**resource-URC network link**
**RUNL**
link from the URC to the resource-URC network

**3.38**
**service**
functionality made available to a user electronically

EXAMPLE An airline reservation service, currency translation services, weather forecasting, restaurant recommendations, etc.

**3.39**
**session**
Control Session
period of connection between a target's socket and a URC for the purpose of user operation of the socket through a URC

**3.40**
**session-full target**
target that requires opening a session with a URC for a part or all of its functionality

**3.41**
**session-less target**
target that provides all of its functionality to any URC without requiring a session

**3.42**
**shared sessions**
sessions of one socket in which the socket element values are common or shared (across sessions) for socket elements with the same identifier

**3.43**
**socket**
user interface socket
machine-operable access and control point for a target

Note 1 to entry: See ISO/IEC 24752-2.

**3.44**
**socket description**
user interface socket description
specification that describes the functions and properties of a socket

Note 1 to entry: A socket description is expressed in XML with the markup language specified in ISO/IEC 24752-2.

**3.45**
**socket element**
variable, command, or notify element

Note 1 to entry: See ISO/IEC 24752-2.

**3.46**
**socket element component**
one value out of a set of values for a dimensional socket element

Note 1 to entry: See ISO/IEC 24752-2.

**3.47**
**socket set**
set
set composed of socket elements and other sets

Note 1 to entry: See ISO/IEC 24752-2.

**3.48**
**static atomic resource**
atomic resource that is provided before runtime, e.g. through a resource sheet

**3.49**
**supplemental atomic resource**
supplemental resource that is used as an atomic entity in the construction of a concrete user interface

**3.50**
**supplemental grouping resource**
supplemental grouping
grouping resource that is provided externally to the target, by resource services

**3.51**
**supplemental label**
label resource made available externally to a target

**3.52**
**supplemental resource**
resource made available externally to a target

**3.53**
**target**
device or service that the user wishes to use

EXAMPLE Video cassette recorder (VCR), online telephone directory.

Note 1 to entry: See ISO/IEC 24752-2.

**3.54**
**target atomic resource**
target resource that is used as an atomic entity in the construction of a concrete user interface

**3.55**
**target description**
**TD**
document containing information on a target that is necessary for discovery of and access to the target and its sockets

Note 1 to entry: There is one target description per target.

**3.56**
**target instance identifier**
identifier for a target instance that is unique among all targets with the same name

**3.57**
**target resource**
resource that is provided by a target in its local network environment



 

**3.58**
**target-URC network**
**TUN**
network connecting the target and the URC, which may employ any networking and connection technologies

Note 1 to entry: Examples of connection technologies include Ethernet, Bluetooth, and 802.11. Examples of networking technologies include UPnP and Java/Jini.

**3.59**
**target-URC network link**
**TUNL**
link between a target or a URC and the target-URC network

Note 1 to entry: Each TUNL is specific to the particular networking and connection technologies that are used.

EXAMPLE A "UPnP on Bluetooth TUNL", a "Jini on 802.11b TUNL", or a "Jini on Bluetooth TUNL".

**3.60**
**universal remote console**
**URC**
device or software through which the user accesses a target

Note 1 to entry: The URC is capable of rendering a user interface for any target. It is "universal" in the sense that it can be used to control any target. It is assumed that users will choose a URC capable of meeting their personal interaction requirements.

**3.61**
**user interface**
**UI**
means by which a user interacts with a target

Note 1 to entry: The interface includes information displayed to the user, values the user can enter or manipulate, and all other actions the user can instruct the target to take.

**3.62**
**user interface implementation description**
**UIID**
description for implementing a user interface to a target, based on the target's socket

Note 1 to entry: May take the form of a standard description of a user interface, or consist of executable code intended for a particular class of URC.

**3.63**
**variable**
socket variable
socket element representing a value relevant to the target's user interface that may be varied by the target or the user

Note 1 to entry: See ISO/IEC 24752-2.



# 4 Universal remote console (URC) requirements

## 4.1 General

A universal remote console (URC) is a device or software through which the user accesses a target. The URC is "universal" in the sense that it can be used to control any compliant target. It presents interfaces for compliant targets to the user and relays user actions to these targets. The URC may present the information to the user in whatever form (visual, auditory, etc) works for the user, the URC, and the environment (e.g. driving, in a quiet environment, etc). This International Standard does not impose any requirements on which form a user uses.

This section states the general requirements that shall be met by a compliant universal remote console, in order for targets to be discovered and chosen by the user of the URC, and for a control session between a URC and a target to be established, maintained and terminated. This document does not provide instructions for implementing this International Standard on specific networking platforms.

## 4.2 Discovery management

### 4.2.1 General

Discovery management comprises functions that handle the discovery of available targets and their sockets, and presentation of this information to the user. The URC builds on the existing discovery mechanisms provided by the implemented target-URC network links (see 4.5).

### 4.2.2 Support for target discovery

In the discovery phase the URC discovers targets and retrieves information about them through TUN specific mechanisms and interpretation of the target description (see 4.2.4). The International Standard does not define specific ways for the URC to present this information to the user.

### 4.2.3 Retrieving documents from the target

#### 4.2.3.1 General

The URC shall be able to retrieve documents from a target through a TUN-specific fetch mechanism (cf. 5.1.2). The URC needs to interpret URIs in order to retrieve the documents that the URIs reference. This includes the resolution of local URIs based on their context.

The URC shall recognize the specified MIME types of target description (see ISO/IEC 24752-4), socket description (see ISO/IEC 24752-2), grouping sheet and resource sheet (see ISO/IEC 24752-5), if applicable.

#### 4.2.3.2 Caching

For the purpose of caching, the URC may assume that a document provided by the target is stable over time if it doesn't change its identifier (URI) and modification date.

Document retrieval may happen in both the discovery and control phase.

### 4.2.4 Interpretation of the target description

Cross-platform information on a target is conveyed in the form of a target description (as defined by ISO/IEC 24752-4), or an implicit target description (as defined by ISO/IEC 24752-6).

NOTE A target description is independent from any natural language.

The URC is not required to interpret all information contained in a target description. However, the URC shall be able to identify a target and open a control session with one of its sockets.

### 4.2.5 Invocation of a target's locator function

A target's locator functions can be made available to the user if available on a target. A locator function is described in the target description and invoked by a URC in a manner that is specific to the TUNL (see 4.5), involving the identifier of the locator element in the target description.



 

## 4.3 Session management

### 4.3.1 General

Session management comprises functions for opening, maintaining and closing control sessions between a URC and a target's socket.

NOTE Session management is necessary for session-full targets (as opposed to session-less targets). URCs must support session management functions, in order to be able to deal with session-full targets.

### 4.3.2 URC open session request

#### 4.3.2.1 General

A URC shall support an open session request to a session-full target, identifying socket by URI, as specified in the target description. The open session request may also contain an authorization code that the URC has received from the target, for example through a session forward request (see 5.5.7).

The target may accept or reject the open session request, or forward the URC to another socket (on the same or a different target).

The open session request and the target's response are specific to a particular TUN platform (see 7.2).

#### 4.3.2.2 URC preferences

The URC may indicate a set of preferences with the open session request. The format of the preferences is out of scope for this International Standard.

NOTE It is intended to address the area of preferences (and related vocabularies) in future versions of this International Standard.

#### 4.3.2.3 Target accepts request

If the target accepts the request, the URC establishes a control session with the socket. A session identifier is provided by the target which is to be used for further events on the session (see following sections).

#### 4.3.2.4 Target rejects request

If the target rejects the request, the URC should inform the user in an appropriate way. This may include the interpretation of the reason for rejection, if applicable.

#### 4.3.2.5 Target forwards request

If a session-full target forwards the URC to another socket, and this socket belongs to a session-full target, the URC should send an open session request to the specified target and socket.

In general, a socket that the URC is forwarded to is specified by its name (URI), as given in the target description of the socket's target. In addition, TUN platforms may use specific forwarding mechanisms to expedite the forwarding process.

### 4.3.3 URC suspend session request

#### 4.3.3.1 General

A URC may request that an active session with a target's socket be suspended. A suspended session may be resumed later via the resume session request (see 4.3.4), if it hasn't timed out.

#### 4.3.3.2 Suggested timeout

The URC may provide a suggested timeout together with the suspend session request.

#### 4.3.3.3 Target response

The target responds to a suspend session request by either rejecting or granting the request. Upon granting, the target will provide a tentative timeout value for the suspended session to the URC. However, the target may at any time decide to drop the session, even before the tentative timeout occurs.

### 4.3.4 URC resume session request

#### 4.3.4.1 General

A URC may request that a suspended session be resumed. The URC shall transmit a session identifier together with the request. The session to be resumed may have been suspended on the same URC or on a different one. (Thus a session can be transferred from one URC to another.)

#### 4.3.4.2 Target response

The target responds to a resume session request by either rejecting or granting the request. Upon granting, the URC re-establishes a control session with the session's socket.

### 4.3.5 URC close session event

The URC shall support a Close Session event that is sent to a target's socket for the purpose of finishing the control phase.

NOTE The URC close session event is typically not the regular way of exiting an application. For complex tasks the built-in UI of a target will likely contain "exit" or "reset" cues (see 5.5.4.2).

### 4.3.6 Support for abort session event

The URC shall support an abort session event from the target. Upon receiving this event the URC shall alert the user to the aborted session in an appropriate way.

### 4.3.7 Established connection

During the control phase, the URC shall track connection status information from the underlying network (TUN), so that it can determine whether it is still connected with a target, and react appropriately.

### 4.3.8 Support for session forwarding

A session-full target may request the URC to open a session with another socket at any time of a control session. There are two types of session forwarding: one that closes the old session ("destructive forward") and one that keeps the old session ("spawn forward").

For session forwarding, the target sends a session forward event to the URC, identifying the type of the forwarding, and containing the URI of the new target and identifier of the specific target socket the URC is forwarded to. The session forward event may also contain an authorization code for the URC to open the new session.

On receipt of a session forward event the URC may send an open session request (see 4.3.2) to the socket that was specified in the session forward event, including the identifier of the new socket, and an authorization code, if given in the event. However, the URC may choose not to do so, in particular for the following reasons: The specified socket may be unavailable on the target-URC network used to communicate with the original socket, in which case the user shall be notified. Or the URC may conclude, with the aid of the user and security negotiation software on the URC, that the new socket is not trusted.



 

For destructive session forwards, the URC may omit the close session event (see 4.3.3) if it has received an abort session event (see 5.5.5) already from the target.

## 4.4 Socket management

### 4.4.1 General

After a successful open session request by a URC, a session is created and maintained between a session-full target's socket and the URC. For session-less target, a URC may communicate with the sockets without session management, i.e. without a prior open session request.

NOTE A URC may have multiple sessions open at the same time if connected to multiple sockets (on the same session-full target or multiple session-full targets).

For both session-full and session-less targets, the socket management on the URC comprises functions for synchronizing socket elements over the TUN. This includes updating the socket in response to a user-initiated value change. This synchronization mechanism is implemented via the TUNL (see 4.5), which relies on networking platform-specific mechanisms for achieving the synchronization.

This International Standard builds upon a distributed object implementation provided by an underlying networking layer (TUN). Implementers of this International Standard may either use an existing (middleware) solution for a distributed object model, or implement their own version.

The implementation of a synchronization mechanism for socket elements is specific to the TUN platform (7.2) and beyond the scope of this document. Typically, an event-based update mechanism is assumed, though polling may also be employed. In the following subsections, the term "propagation" is used to include all possible synchronization mechanisms.

### 4.4.2 Synchronization of variables

#### 4.4.2.1 General

NOTE 1 Socket variables are specified in ISO/IEC 24752-2.

The values of the socket variables (except for stream types), and their components, if any, shall be synchronized with the corresponding session on the URC.

NOTE 2 For variables with stream types, the stream may or may not be managed by the socket.

NOTE 3 Variables of sockets of session-less targets are always shared across all connected URCs. Session-full targets may share a variable across multiple sessions (with multiple URCs). In general, a URC cannot determine whether a variable is shared with other sessions or not. However, the 'sharedSessions' attribute of the <socket> element in the target description may provide hints (see ISO/IEC 24752-4).

#### 4.4.2.2 Acceptance/rejection

If a value is changed on the URC (e.g. in response to a user's action) this change shall be propagated to the socket on the target. The socket may accept or reject the change. If the change is accepted it is propagated back to the URC (and all other connected URCs). If the change is rejected the socket notifies the URC about the rejection. The URC shall update the variable value according to the socket's instructions.

#### 4.4.2.3 Only new values

A URC shall not propagate changes to the target where the old value is equal to the new value of a variable.

NOTE It is necessary to make it possible for the socket to reject a variable change from the URC. This is because of network uncertainties (delays) and security reasons. Pertinent use cases include that the variable is not writeable at the time of the change but the URC has not yet received a state change from the socket that would make the variable's write dependency false. Or multiple URCs are changing one variable at the same time in which case it is up to the socket to decide which value to accept.

#### 4.4.2.4 Undefined values

A URC shall support information on undefined variable values which are specified as part of the synchronization information from the target.

NOTE 1 Variable's values may be undefined for various reasons. For example, they may have failed to initialize properly. Or they could reflect features that are not available on a particular target instance (see 4.4.2.5).

NOTE 2 In dependency expressions, an author can check whether a variable's value is defined with the function uis:hasDefinedValue(path), see ISO/IEC 24752-2.

#### 4.4.2.5 Availability of variables

A URC shall support information on the availability of variables which is specified as part of the synchronization information from the target.

A URC shall not request to set a variable's value if the variable is unavailable at runtime.

The URC should not present to the user any user interface object that is mapped to an unavailable variable.

NOTE In dependency expressions, an author can check on the availability of a variable by using the function uis:isAvailable(path), see ISO/IEC 24752-2.

### 4.4.3 Command and synchronization of command state

#### 4.4.3.1 General

NOTE 1 Socket commands are specified in ISO/IEC 24752-2.

A URC shall support an invocation request on a socket command through TUN platform specific mechanisms, by providing the identifier of the command element in the socket description.

NOTE 2 Nothing in the above should be construed as implying that 'synchronous' or blocking remote method calls are used to invoke commands on the target or to synchronize the states of socket commands between URC and target. The synchronization of multiple socket commands may proceed concurrently and asynchronously, subject only to the target respecting constraints in the socket description and the URC respecting state values returned from the target.

#### 4.4.3.2 Acceptance/rejection

The request for execution may be rejected by the socket in which case the socket will notify the URC about the rejection.

#### 4.4.3.3 Local input parameters

On request for execution of a command with local input parameters the URC shall send to the target the values of all local input parameters that the command contains. This happens in a TUN platform specific mechanism.

No other synchronization of a command's local input parameters to the target or from the target shall take place.

NOTE Global input parameters (references from a command to variables that serve as input parameters for the command) are not conveyed to the target in this way. Instead, they are synchronized on value changes as specified for variables (see 4.4.2).



 

#### 4.4.3.4 Local output parameters

##### 4.4.3.4.1 General

On return of a command with local output parameters the URC shall receive from the target the values of all local output parameters that the command contains. This happens in a TUN platform specific mechanism.

No other synchronization of a command's local output parameters to the target or from the target shall take place.

NOTE Global output parameters (references from a command to variables that serve as output parameters for the command) are not received from the target in this way. Instead, they are synchronized on value changes as specified for variables (see 4.4.2).

##### 4.4.3.4.2 Concurrency restriction

A command of type uis:basicCommand or uis:timedCommand shall not be invoked multiple times unless the previous execution has returned and the previous results (values of output parameters) have been received.

NOTE This restriction does not apply to commands of type uis:voidCommand which cannot have output or input-output parameters (see ISO/IEC 24752-2).

#### 4.4.3.5 Command state

A command of type uis:basicCommand or uis:timedCommand has a state which is synchronized only one-way between a socket and the connected URCs. A URC shall support updates from a target with regard to a command's state - only the target can change the state of the command, and a URC shall not change the state of the command.

The URC should respect the meaning of command states, as defined in ISO/IEC 24752-2, and indicate a command's state and timeToComplete (when relevant) to the user.

#### 4.4.3.6 timeToComplete

A command of type uis:timedCommand has a timeToComplete field that is only valid if the command state is inProgress. timeToComplete is synchronized one-way between a socket and the connected URCs. A URC shall support this synchronization. The value of timeToComplete is a hint from the target to the URC as to how long the estimated time to complete is – no guarantee is provided by the target whatsoever. A URC shall not change timeToComplete - only the target can change it.

#### 4.4.3.7 Undefined command states

A URC shall support information on undefined command states which are specified as part of the synchronization information from the target. This applies only to command types that allow for command state information (uis:basicCommand, uis:timedCommand).

NOTE 1 A command's state may be undefined for various reasons. For example, the command may have failed to initialize properly. Or it could not be available at runtime on a particular target instance (see 4.4.3.8).

NOTE 2 In dependency expressions, an author can check whether a command's state is defined with the function uis:hasDefinedValue(path), see ISO/IEC 24752-2.

#### 4.4.3.8 Availability of commands

A URC shall support information on the availability of commands which is specified as part of the synchronization information from the target.

A URC shall not request to execute a command that is unavailable at runtime.

The URC should not present to the user any user interface object that is mapped to an unavailable command.

NOTE In dependency expressions, an author can check on the availability of a command by using the function uis:isAvailable(path), see ISO/IEC 24752-2.

### 4.4.4 Notification receipt and acknowledgment

#### 4.4.4.1 General

NOTE 1 Socket notifications are specified in ISO/IEC 24752-2.

A URC shall provide support for receiving and acknowledging notifications.

If a socket notification becomes active (triggered by the target's socket), a URC should present this to the user in an appropriate way, based on the type of the notification.

NOTE 2 A target can share a notification across multiple sessions (with multiple URCs). In general, a URC cannot determine whether a notification is shared with other sessions or not.

#### 4.4.4.2 Explicit user acknowledgment

For all notifications except of type "show", the URC should request the user to acknowledge the notification. Upon the user's acknowledgment, the URC shall send a notification acknowledge event to the target. However, the URC shall not change the notification's state on its own, but rather wait until the target dismisses the notification (by setting its state back to "inactive").

EXAMPLE A URC requests a user to explicitly acknowledge a notification by providing a dialog box with an "OK" button. Pressing this button is an explicit acknowledgement.

NOTE Notifications of type "show" do not require that they be explicitly acknowledged by the user.

#### 4.4.4.3 Stacked notifications

A target may activate a notification while another notification is already active. In this case, the second notification takes priority over the first. The URC shall keep a stack of notifications with the most recent on top. All notifications that are not on the top are deemed to be in state "stacked" (this is internal to the URC, and not shared with the target). The URC shall respect the order in which notifications become active, so that the most recently activated notification is always the one presented to the user. If the target inactivates the top notification the next notification on the URC's stack becomes active again and is presented to the user. It is up to the URC to ensure that the user doesn't accidentally dismiss a new notification if it is presented just as the user is responding to a previous notification.

NOTE To prevent misuse such as denial of service attacks, a URC can quit the control session with the target at any time, if the target is overloading the URC by notifications or other events. In this case the URC should send a URC close session event (see 4.3.5) before aborting the session.

#### 4.4.4.4 Undefined notification states

A URC shall support information on undefined notification states which are specified as part of the synchronization information from the target.

NOTE 1 A notification's state may be undefined for various reasons. For example, the notification may have failed to initialize properly. Or it could not be available at runtime on a particular target instance (see 4.4.4.5).

NOTE 2 In dependency expressions, an author can check whether a notification's state is defined with the function uis:hasDefinedValue(path), see ISO/IEC 24752-2.



 

#### 4.4.4.5 Availability of notifications

A URC shall support information on the availability of notifications which is specified as part of the synchronization information from the target.

A URC shall not acknowledge a notification that is unavailable at runtime.

The URC should not present to the user any user interface object that is mapped to an unavailable notification.

NOTE In dependency expressions, an author can check on the availability of a notification by using the function uis:isAvailable(path), see ISO/IEC 24752-2.

### 4.4.5 Actual indices for dimensional socket sets/elements

The sets of actual index value combinations for every dimensional socket set and socket element, if any, shall be synchronized between the socket and the URCs that participate in a joint session with the socket.

The URC may request to add or remove an actual index for a dimensional socket set or element. Since indices are dealt with in a hierarchical fashion ("index path"), this request shall include all indices of the index path which precede the set/element where a new index is requested to be added (or the actual index is requested to be removed, respectively). The request may include proposed initial values for resulting new socket element components, if a new index is requested to be added.

The socket will either accept or reject the modification. If it accepts the modification this is propagated back to the URC (and all other connected URCs). If it rejects the request the socket notifies the URC about the rejection.

A URC shall update its corresponding session upon receipt of a modification of the actual index values for a dimensional socket set or element. In case of a new index being added the socket will provide initial values for the resulting new element components, and all connected URCs shall initialize the new components' values according to the socket's instructions.

### 4.4.6 Support for socket element dependencies

A URC may provide support for assert dependencies of commands. It should provide support for all other dependencies of variables, commands and notifications. Socket element dependencies are specified in ISO/IEC 24752-2.

This includes the interpretation of the dependency values (as XPath expressions) while the URC is connected with a socket. The URC should indicate to the user whether a socket element is readable or writeable, and whether the set of actual indices for a dimensional socket element is modifiable, according to the current dependency values.

### 4.4.7 User response timeouts

#### 4.4.7.1 General

A URC shall support the 'timeout' attribute on notifications (see ISO/IEC 24752-2).

The following requirements apply to timeouts generated by the target as a result of user inactivity or slow performance. These are referred to as 'user response timeouts'.

#### 4.4.7.2 Presenting the timeout to the user

The URC may present the value of the 'timeout' attribute to the user.

#### 4.4.7.3 Requesting a timeout extension

The URC may request a timeout extension from the target, according to an internal user model or based on a value set by the user.

NOTE The mechanism for timeout extension requests between a URC and a target is out of scope for this International Standard.

## 4.5 Target-URC network link on the URC

### 4.5.1 General

The target-URC network link (TUNL) on the URC is the link from the URC to the target-URC network. A URC can support multiple TUNLs allowing it to function with different network technologies and platforms. Every URC shall provide at least one TUNL. Each TUNL is specific to the particular networking and connection technologies that are used. Examples would be a "UPnP on Bluetooth TUNL", a "Jini on 802.11b TUNL", etc. See 7.2 for further details on target-URC networks.

The specification of TUN platform specific techniques and mechanisms to implement this International Standard on a particular platform is outside the scope of this International Standard.

### 4.5.2 Support for dynamic atomic resources

The URC shall support reception and updating of dynamic atomic resources. These atomic resources may change at runtime.

## 4.6 Resource-URC network link (RUNL) on the URC

### 4.6.1 General

The resource-URC network link (RUNL) is the link from the URC to the resource-URC network. The URC may contact one or more resource services through the RUNL in order to retrieve supplemental resources that can be used wherever needed.

NOTE Supplemental resources can be used wherever needed, such as: for constructing tailored user interfaces for sessions with sockets; for translating information on a target or socket into another language; for displaying images for discovered targets and sockets instead of their labels; etc.

A RUNL is specific to the particular networking and connection technologies that are used on the resource-URC network.

This International Standard does not specify any particular RUNL, nor does it specify how the URC requests resources from a resource service.

EXAMPLE Web service based technologies such as "SOAP over HTTP over TCP/IP".

### 4.6.2 Locating resource services

Resource services provide supplemental resources (beyond the target resources) for target descriptions, socket descriptions and user interface implementation descriptions (UIIDs). A URC may employ any resource service that is available on any network link (RUNL) of the URC.

A URC may use, but is not limited to, any of the following ways to locate a resource service:

a) the target description includes a reference to a resource service, via the <resSvc> element in a target description, as specified in ISO/IEC 24752-4;

b) the link to a resource service is hard-coded into the URC, or the URC retrieves it from a configuration file or known server address. Typically the resource service would be maintained by the URC manufacturer;



 

c) there are known public directory services for resource services, for example based on UDDI.

## 4.7 User interface generation

### 4.7.1 General

Finally, a URC shall provide a concrete user interface for a control session with a target's socket, based on the socket description and target resources, and optionally involving target-external resources (supplemental resources).

### 4.7.2 Use of target resources and supplemental resources

The URC may use the target resources or use supplemental resources obtained from resource services (or any combination of them). It may use a grouping resource or a user interface implementation description (UIID) obtained from the target or through the resource-URC network or any other source (e.g. a UIID available on the URC). There are no limits regarding the possibilities of mixing and matching as long as the target's socket is used as the vehicle for controlling the target.

### 4.7.3 Use of atomic resources

Labels and other atomic resources can be defined for a socket element and for a UIID element that binds to the socket element. When basing a user interface on a UIID, the URC should give precedence to the atomic resources defined for the UIID element, if it is specified. If no atomic resources are given for a UIID element, the URC should use the appropriate atomic resources for the socket element that the UIID element binds to.

### 4.7.4 Preference for resources from target vendor

As a general principle, the URC should construct a user interface which is as close to one designed and published by the target vendor as is practical given the limitations of the URC device and the needs and preferences of the user.

As a consequence, when the URC has multiple resources available to fill a single role in the user interface construction, such as multiple labels that all are applicable to the same socket element, in the absence of user preferences indicating otherwise the resources provided by the target and target vendor should be used in preference to those offered by third parties. In particular, dynamic atomic resources should be preferred to static atomic resources at runtime.

### 4.7.5 About user preferences

Nevertheless the URC still may elect to employ any applicable resources. There is no standard established in this specification for what constitutes user preferences. This may be an interactive input from the user during the session, a setting made by the user during other sessions performed with this same URC, or even a reasonable inference from the product class of the URC itself. A consumer who elects to buy a speech output URC, for example, can reasonably be assumed to prefer audio display of labels.

### 4.7.6 Ordering of elements

A URC need not construct a user interface that preserves the ordering of elements within a socket description or UIID document, but it is recommended that target vendor's ordering be preserved as far as is practical given the limitations of the URC device and the needs and preferences of the user.

The URC should present to the user the instances of a dimensional socket set or element in the order that reflects the order of its index values, as defined by the fundamental facet 'ordered' of the index type definition (see XML Schema Part 2: Datatypes). For index types defined by restriction (enumeration types), the order should reflect the order in which the values are enumerated in the definition. If enumeration types are merged by the union operator, the order should follow the order in which the base types appear in the union statement.

### 4.8 Security and privacy requirements

See 8.2.

## 5 Target components and requirements

### 5.1 Discovery management

#### 5.1.1 General

Discovery management comprises functions that handle the advertisement of available sockets to the network and dissemination of their properties to interested URCs. The target builds on the existing discovery mechanisms provided by the implemented target-URC network links. In the discovery phase the targets advertise themselves to URCs through platform-specific mechanisms and the distribution of the target description.

#### 5.1.2 Target instance identifier

A target shall have an instance identifier that is made available to the URCs through all of the target's target-URC network links. The instance identifier shall be unique among all targets with the same name (see ISO/IEC 24752-4).

NOTE The target instance identifier is made available to the URC through TUN-specific ways (see 7.2.2), and can be referred to in atomic resource descriptions (see ISO/IEC 24752-5). However, target description and socket description are supposed to be common for all instances of a product, and therefore do not bear references to target instances.

#### 5.1.3 Support for document deployment

A target shall provide a TUN-specific fetch mechanism that allows a URC to retrieve the target description, socket descriptions and target resource sheets by URI. This fetch mechanism may optionally include authentication mechanisms for URC and/or target.

The target shall support the specified MIME types of target description (see ISO/IEC 24752-4), socket description (see ISO/IEC 24752-2), resource sheet and grouping sheet (see ISO/IEC 24752-5), if applicable.

To facilitate performance improvements through URC-side caching, the target may support specific mechanisms such as conditional document requests, based on the modification date of the document.

Document deployment shall be available in the discovery AND in the control phase.

#### 5.1.4 Target description

A target shall provide exactly one target description, independent of the number of sockets the target provides. This shall be either a target description as specified in ISO/IEC 24752-4, or an implicit target description as specified in ISO/IEC 24752-6.

The target description shall contain pointers to socket descriptions for all of its sockets (including those that the URC may be forwarded to). It shall also reference all resource sheets containing the required target atomic resources and the grouping sheet containing the required grouping (see 5.4.5).

NOTE 1 The target description is a document that provides all information and references required by a URC in the discovery phase, allowing the URC to discover, identify and understand a target and its socket(s). The location of the target description is provided to the URC by the target in a TUN platform specific way.

NOTE 2 The target description is written in a TUN platform independent manner (except for platform-specific mapping information which may be included). See Annex B for an example target description.



 

NOTE 3 The target description is independent of natural languages. Target atomic resources (see 5.4.2) and/or supplemental atomic resources need to be employed so that relevant parts of its content can be rendered to a user.

#### 5.1.5 Support for locator functions

The target shall provide support for the locator functions that are specified in the target description, so that the URC can invoke them during the discovery and control phases in a manner that is specific to the TUN. The identifier of the <locator> element is hereby used to identify the specific locator function.

### 5.2 User interface socket

#### 5.2.1 General

The *user interface socket* provides the actual access to and control of a functional unit of the target. The socket is used to operate the target from a *universal remote console (URC)* once a control session is initiated. A socket exposes the functionality and state of a target in a machine-interpretable manner so that a URC can access and operate it, thus providing access and control to the URC user.

NOTE A UI socket is independent of a TUN platform. At runtime, a UI socket is connected to one or more TUN platforms through target-URC network links.

#### 5.2.2 A target's sockets

A target shall provide one or more user interface sockets (or short "sockets"). A socket is located on a target.

NOTE In an object-oriented implementation of a URC, the socket might be represented on the client side as a proxy object. We recommend calling this object "socket mirror object", to make a differentiation to the target-side master socket.

If a target has multiple sockets, each socket shall represent a functional unit of the target's functionality. In order to accommodate a wide variety of URCs, the user interface information is provided in an abstract fashion rather than in a specific format for the interface (e.g. a visual interface, a verbal interface, an interface optimized for a smartphone, etc).

Collectively, the target's sockets shall provide access to all of the functionality provided by the built-in user interface of the target.

#### 5.2.3 Socket elements

##### 5.2.3.1 General

A socket contains variables, commands and notifications.

a) The *variables* represent dynamic data related to the state of the user-target interaction. They shall include all of the dynamic data on the target socket's state 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, or the current floor of an elevator. All variables have a value. A target may change the value of a variable at any time. A URC may attempt to change the value of a variable at any time, but the target may accept or reject the change. Changes of a variable on one end (target or URC) are propagated to the other end immediately (two-way synchronization).

b) A *command* is a core function that a user can request a target (through its socket) to perform and that cannot be represented by a variable. The *commands* shall include all of the target functions that can be called explicitly or implicitly by users. Examples include the 'search' command of an airline reservation system or the 'seek' command of a CD player.

c) The *notifications* are special states where normal operation is suspended, such as an exception state. *Notifications* are triggered by the target. They shall cover all exceptions that the target needs to inform the user about. 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 notification has a state, which shall be made available for inspection by the URC. Only the target (through its socket) can update the state of a notification; updates are propagated to the URC.

The socket is responsible for maintaining accurate value and state information for variables, commands and notifications.

#### 5.2.3.2 Multiple connected URCs

The socket may share variable values, and command and notification states among multiple connected URCs, or maintain separate sets (sessions) for each URC. For shared elements, the socket shall propagate the changes to all connected URCs.

#### 5.2.3.3 Multi-user and single-user targets

A multi-user target such as a television may maintain a socket that shares all variables for all connected URCs. A single user target such as a currency conversion service may maintain a socket that has multiple copies of the variable set, each for every connected URC. Other targets may adopt a mixed approach. For example, an elevator with multiple URCs connected may have some shared variables (e.g. the position of each elevator) and some URC-specific variables (e.g. the user's desired floor).

#### 5.2.3.4 Synchronization

This International Standard builds on top of existing networking platforms rather than replacing them. It relies on TUN platform specific ways of synchronizing the socket state between a target's socket and connected URCs.



## 5.3 User interface socket description

### 5.3.1 General

The target manufacturer shall provide a user interface socket description for each of the target's sockets. The socket description shall either conform to ISO/IEC 24752-2 which specifies an XML based language for dedicated socket description documents (see Table 2 for an example), or to ISO/IEC 24752-6 which specifies an "implicit socket description" embedded in a WSDL document.

The socket description shall describe all of the information and control features represented in the socket. The socket description shall also specify all static information about the socket that is available to users of the built-in user interface, such as model numbers.

NOTE 1 The *user interface socket description* is a specification that describes the elements of the socket in a TUN platform independent way. It is designed to support every possible URC class, including explorable user interfaces and other user interface concepts, such as intelligent software agents.

NOTE 2 For legacy products that have a TUNL by default (e.g. UPnP or Java/Jini), the manufacturer of a target could provide a set of missing socket descriptions externally.

## 5.4 Target resources

### 5.4.1 Target grouping resources

Grouping resources, as defined in ISO/IEC 24752-5, may be provided as part of grouping sheets.

Except for the grouping resource which shall be provided for conformance (see 5.4.5), any number of grouping resources may be provided by the target manufacturer.

NOTE A grouping resource specifies a presentational structure of user interface socket elements or UIID elements in a top-down fashion, and is provided externally to a socket description. In a grouping resource individual subgroups and user interface elements may occur multiple times (in different parent groups).

### 5.4.2 Target atomic resources

#### 5.4.2.1 General

A target may provide target atomic resources.

A *target atomic resource* is an atomic resource that is provided by a target through its support for document deployment (see 5.1.2). As with other resources, target atomic resources may or may not be used by a URC, and may be supplemented or replaced by appropriate supplemental resources from external sources.

Target atomic resources are categorized into the following classes: labels, keywords, help, access keys, and location. For more information on atomic resources, refer to ISO/IEC 24752-5.

#### 5.4.2.2 Labels

A *label resource* is any object that is used as a label entity in a concrete user interface. In general, label resources may be textual, graphical, aural, or employ any other modality, including multiple modalities.

Except for the labels which shall be provided for conformance (see 5.4.5), any number of labels may be provided by the target manufacturer.

#### 5.4.2.3 Keywords

Any number of keyword resources may be provided by the target manufacturer.

#### 5.4.2.4 Help

Any number of help resources may be provided by the target manufacturer.

#### 5.4.2.5 Access keys

Any number of access key resources may be provided by the target manufacturer.

#### 5.4.2.6 Location

Any number of location resources may be provided by the target manufacturer.

NOTE A location resource can be used to describe a location of a target, for example. In this case, the location resource must point to the <target> element of the target description.

For targets that have a physical location, the content of the target location resources should be updated when the target is moved.

EXAMPLE When a device is purchased and installed in a home/office/hotel, the location resource should be updated.

#### 5.4.2.7 Entities that target atomic resources apply to

Manufacturers may provide atomic resources (in one or more natural languages) for the following entities:

— in the target description: for the <target>, <locator> and <socket> elements;

— in the socket description: for the root <uiSocket> element; for the <set> element; for <variable>, <command> and <notification> elements; and for internal type definitions via <simpleType> and <complexType>;

— in the resource sheet: for <Group> elements;

— in an external XML definition (XSD) file: for <simpleType> and <complexType> elements;

— for any UIID element that has an identifier.

### 5.4.3 Target resource sheets

Target atomic resources are specified in *resource sheets,* as defined in ISO/IEC 24752-5. See Annex B for an example resource sheet.

A target resource sheet should remain stable as much as possible. A target resource sheet may change only if its identifier (URI) or its modification date is changed.

### 5.4.4 Target grouping sheets

Target grouping resources are specified in *grouping sheets,* as defined in ISO/IEC 24752-5. See Annex B for an example grouping sheet.

A target grouping sheet should remain stable as much as possible. A target grouping sheet may change only if its identifier (URI) or its modification date is changed.

### 5.4.5 Required target resources

#### 5.4.5.1 General

The following label resources (see 5.4.3) shall be provided by the manufacturer of the target:

a) a grouping resource for every socket of the target which references all UI socket elements but no elements from any other socket or UIID;

b) a text label for the <target> element in the target description (as static atomic resource);

NOTE 1 A text label has a role value of http://myurc.org/ns/res#label, with a dc:type value of "Text" and a dc:format value of "text/xml" or "application/xml", and its content is an unformatted character sequence (see ISO/IEC 24752-5).

NOTE 2 Text labels can be either static or dynamic. Static labels are part of resource sheets (see ISO/IEC 24752-5). Dynamic labels are considered part of the required target resources and need to be provided at runtime for conformance (cf. 5.4.6).

c) a text label for every <socket> element in the target description (as static atomic resource);

NOTE 3 This would also be the text label for the <uiSocket> element of the socket description.

d) a text label for every <locator> element in the target description (as static atomic resource);

e) a text label for every <variable>, <command> and <notify> element in the socket description for every socket (as static or dynamic atomic resource);

f) a text label for every type in the socket description for every socket if the type name is not human understandable (as static or dynamic atomic resource);

g) a text label for every enumerated value in the socket description for every socket if the enumerated value is not human understandable (as static or dynamic atomic resource);

h) a text label for every <Group> element in the required grouping resources, see (a) above (as static atomic resource).

NOTE 4 All resource sheets containing a required target resource have to be referenced from the target description (see 5.1.4).

NOTE 5 For legacy products that have a TUNL by default (e.g. UPnP or Java/Jini), the manufacturer of a target could provide a set of required target resources externally, i.e. as external (supplemental) resource sheets.



 

#### 5.4.5.2 Textual and modality independent

In order to allow a variety of URC interfaces to be constructed (to match the environment, user constraints, and device characteristics of the URC, as well as to support intelligent agents) the information in the required target resources shall be textual and modality independent. That is, it shall not assume that it will be presented in visual or auditory form and does not make assumptions about the size of the display, presence of a keyboard, etc.

### 5.4.6 Conformance in a natural language

A target shall conform in at least one natural language.

For ease of translation, a target should conform in at least one commonly used language.

A target conforms in a natural language if it provides at least one instance for each of the required target resources (a complete set, as defined in 5.4.5) that is applicable for that natural language, with the labels covering the target description and each of the target's sockets. If one of the target's sockets contain socket elements for which atomic resources are provided at runtime ("dynamic atomic resources"), these labels are considered part of the required target resources and need to be accommodated for conformance in a natural language.

There is an exception for culture-specific sockets: If there is a subset of sockets so that each socket in this subset provides access to the very same information and functions of the target, it is sufficient that the required target resources in that natural language are provided for one of the sockets in the subset.

### 5.4.7 Target user interface implementation descriptions (UIIDs)

UIIDs are specific descriptions for implementing an interface to a socket. They are described more fully in 6.10. A UIID may be provided in any form. This allows support for proprietary technologies on specific URC devices.

A target may provide multiple user interface implementation descriptions (UIIDs), called *target UIIDs*, possibly some of them pertaining to the same socket.

As with all UIIDs, a target UIID is always based on one or more sockets, i.e. any interaction between a target and a UIID passes through a socket.

NOTE Unlike the socket description, a UIID may or may not be used by a URC. A URC can decide to use any UIID provided externally to the target, or to generate a user interface directly from the socket description.

A target UIID may use target resources, or alternatively use equivalent information built into the target UIID or pulled from supplemental resources (see Clause 6).

## 5.5 Session management

### 5.5.1 Support for URC open session request

#### 5.5.1.1 General

A session-full target shall support an open session request from a URC. A session-less target shall not provide a method for opening a session.

An open session request from a URC contains the identifier (URI) of the requested socket. A target is required to respond to a URC open session request in either one of the following ways: accept, reject, or forward to another socket (see 5.5.7). The open session request and the target's response are specific to a particular networking platform.

#### 5.5.1.2 Acceptance

If the target accepts the request, it shall provide sufficient information for the URC to open a session with the socket. This shall include the provision of a session identifier. Each socket shall be connected to a TUNL that supports the same TUN platform as the target uses for advertising its socket (discovery). If the target advertises via multiple TUN platforms these shall be supported by every socket as well.

#### 5.5.1.3 Rejection

In case of rejection, the target shall respond to the open session request with either one of the following codes:

— "Bad Request": the target/socket name is invalid;

— "Unauthorized": the open session request did not contain an authorization code, though required;

— "Authorization Failed": the authorization code is invalid;

— "Too Many Sessions": the target has too many sessions open;

— "Inappropriate": the target/socket is inappropriate for the user and/or the use context;

— "Gone": the target/socket is currently not functioning;

— "Internal Server Error": an internal error occurred in the target;

— "Other": there is no code provided for the rejection reason;

— "Unknown": the target doesn't want to provide a reason for rejection.

#### 5.5.1.4 Forwarding

If the target forwards the URC to another socket (of the same or another target), it shall provide sufficient information for the URC to request a session with the other socket. This includes the names (URIs) of the target and socket which the URC is forwarded to, as given in the target description of the destination target. In addition, TUN platforms may use other forwarding mechanisms to expedite the forwarding process.

NOTE The way a target reacts to open session requests while there is already a session open (with another URC), determines whether it is able to handle multiple URCs at the same time. A parallel session on the socket may or may not share the socket state of the existing session. Typically, a TV would allow multiple sessions that share socket state values. Using the forward feature, an ATM may allow for one "operation" session, and multiple "waiting queue" sessions.

### 5.5.2 Support for URC suspend session request

A session-full target shall support a suspend session request from a URC. A session-less target shall not provide a method for suspending a session.

A session-full target shall respond to a URC suspend session request (see 4.3.3) by either one of the following:

— rejection;

— grant request, and provide a tentative timeout value for the suspended session to the URC. The target may take the value suggested by the URC, or any other timeout value. The tentative timeout value shall indicate how long the target intends to keep the session alive. It is no guaranteed value, though; the target can drop the session at any time after the request.

In case of rejection, the target shall respond to the suspend session request with either one of the following codes:

— "Bad Request": the session identifier is invalid;



 

— “Inappropriate”: the current session state is inappropriate for session suspension;

— “Not Implemented”: the target doesn’t implement the suspend/resume session function;

— “Internal Server Error”: an internal error occurred in the target;

— “Other”: there is no code provided for the rejection reason;

— “Unknown”: the target doesn’t want to provide a reason for rejection.

### 5.5.3 Support for URC resume session request

A session-full target shall support a resume session request from a URC. A session-less target shall not provide a method for resuming a session.

A session-full target shall respond to a URC resume session request (see 4.3.4) by either one of the following:

— reject request. This may be because the session has been dropped by the target, or for other reasons;

— grant request, and re-establish the suspended session with the URC. The session that is resumed may have been suspended on the requesting URC or on a different one. (Thus a session can be transferred from one URC to another.)

In case of rejection, the target shall respond to the resume session request with either one of the following codes:

— “Bad Request”: the session identifier is invalid;

— “Too Many Sessions”: the target has too many sessions open;

— “Inappropriate”: the current session state is inappropriate for session resumption;

— “Not Implemented”: the target doesn’t implement the suspend/resume session function;

— “Internal Server Error”: an internal error occurred in the target;

— “Other”: there is no code provided for the rejection reason;

— “Unknown”: the target doesn’t want to provide a reason for rejection.

### 5.5.4 Support for URC close session event

#### 5.5.4.1 General

A session-full target shall support a close session event from a URC. A session-less target shall not provide a method of closing a session.

A session-full target shall respond to a URC close session event by closing the session immediately.

— For single user sockets (e.g. ATM): Any transaction already completed and confirmed or saved shall not be undone. Incomplete transactions will be abandoned and not saved.

— For multi-user sockets (e.g. TV): It is up to the socket how to react to a URC disconnect.

#### 5.5.4.2 Exiting an application

The URC close session event should not be the regular way of exiting an application.

For complex tasks the built-in user interface of the target will likely contain “exit” or “reset” cues. In this case the user interface socket shall contain equivalent cues (using socket commands, variables or notifications) to let the user quit the application in a proper way.

### 5.5.5 Abort session event

#### 5.5.5.1 General

Sockets may at various times and for various reasons abort user sessions, that is to say terminate a session without a user request or confirmation. In this case the target shall send an abort session event to the URC.

In case of abortion, the target shall provide one of the following codes stating the reason for session abortion:

— "Session Expired": A timeout has occurred on the target;

— "Suspended Session Timeout": a timeout has occurred on the target after the session has been suspended;

— "Session Forwarded": the session has been closed by the target due to a forwarding request;

— "Too Many Sessions": the target has too many sessions open;

— "Gone": the target/socket is currently not functioning;

— "Internal Server Error": an internal error occurred in the target;

— "Other": there is no code provided for the rejection reason;

— "Unknown": the target doesn't want to provide a reason for rejection.

#### 5.5.5.2 Conditions

Conditions for aborting sessions may vary. However, socket designers should make sure that the conditions for aborting sessions do not impose a burden or barrier for people with disabilities or people in constrained environments, see (c) below.

Particular conditions for aborting sessions may include:

a) connection dropouts. In this case the URC may not even receive the abort session event because of lost connection;

b) timeout for an expected user response. See 4.4.7 and 5.6.10;

c) user error rate. Users with disabilities or in constrained environments (e.g. noisy, bumpy) may generate a higher error rate than the socket designer expects. Sessions should not be aborted because of a high user error rate, except for security reasons such as repeated password failures in the user login procedure.

### 5.5.6 Established connection

#### 5.5.6.1 General

During the control phase, the target shall track connection status information from the underlying TUN network, so that it can determine whether it is still connected with a URC.

#### 5.5.6.2 Loss of connection

In case of a loss of connection during the control phase, it is up to the socket how to react, e.g. performing a reset operation immediately, or only after a certain timeout. Where reasonable, the socket should prompt the user whether or not they want to resume their dialog, if the connection has been re-established within a certain amount of time. In either case, the state of the socket shall be made clear to the user after re-connecting.

### 5.5.7 Session forwarding

#### 5.5.7.1 General

A session-full target may request the URC to open a session with another socket (of the same or a different target) at any time of a control session. There are two types of session forwarding: one that closes the old session ("destructive forward") and one that keeps the old session ("spawn forward" or "sub-session request"). In the second case, the old and new sessions will end independently from each other.

#### 5.5.7.2 Indication of session forwarding in a socket description

If the execution of a socket command can trigger a session forwarding, the target should indicate so in the socket description by including the "subSession" function in the postcondition of the command element (see ISO/IEC 24752-2).

#### 5.5.7.3 Session forward event

For session forwarding, the socket shall send a session forward event to the URC, identifying the type of the forwarding, and containing the URI of the new target and identifier of the specific target socket the URC is forwarded to. The session forward event may also contain an authorization code for the URC to open the new session.

The new socket is not required to be available on the TUN used by the originating target.

#### 5.5.7.4 After the session forward event

After the session forward event it is up to the URC, possibly aided by the user, whether it follows the session forward request from the target. If the session forward event is followed, the URC will send an open session request (see 4.3.2) to the target it is being forwarded to, if the target is session-full. In this case, the target should accept such an open session request triggered by a session forward event, if occurring within a reasonable timeframe from each other.

The socket cannot transfer the URC to a new session on its own. However, in the case of a destructive forward, a session-full target may abort the old session by sending an abort session event to the URC (see 5.5.5).

#### 5.5.7.5 Forwarding within the same target

For session forwarding within the same session-full target, or between targets that implement the same target-URC network link, the forward event to the URC may contain networking platform-specific parameters to optimize the transition between the sockets.

EXAMPLE The provision of a networking specific target-socket identifier may obviate the time-consuming discovery process for a new socket.

## 5.6 Socket management

### 5.6.1 General

After a successful open session request by a URC, a session shall be created and maintained between a socket and the URC.

### 5.6.2 A socket's sessions

A session-full target's socket may have multiple sessions open at the same time if multiple URCs are participating in a joint control session with the socket. It is up to the target and its socket whether the values of the socket elements are shared between sessions or not. In fact, this may be different for different socket elements, and a shared element may only be shared among a subset of connected URCs.

For session-less targets, there are no sessions, only socket connections with URCs. In the case of multiple URCs being connected with the same socket, the socket elements are shared between the URCs.

### 5.6.3 Indication of availability of socket elements at runtime

The target shall indicate to the URC through TUN-specific means which of the optional socket elements are available for a particular session (cf. 7.2.5.2).

NOTE "Optional socket elements" are socket elements that have an 'optional' attribute with value "true" (see ISO/IEC 24752-2).

### 5.6.4 Synchronization mechanism

The socket management comprises functions for synchronizing socket elements with connected URCs. This includes updating the socket in response to a user-initiated value change, and propagating changes to the connected URCs. This synchronization mechanism is implemented via the TUNL, which relies on TUN platform specific mechanisms for achieving the synchronization.

NOTE 1 The socket specification builds upon a distributed object implementation provided by the underlying TUN layer. Implementers of sockets may either use an existing (middleware) solution for a distributed object model, or implement their own version.

NOTE 2 The implementation of a synchronization mechanism for socket elements is specific to the TUN platform and beyond the scope of this International Standard. Typically, an event-based update mechanism is assumed, though polling may also be employed. In the following subsections, the term "propagation" is used to include all possible synchronization mechanisms.

### 5.6.5 Variable and synchronization of variable values

#### 5.6.5.1 General

Socket variables are specified in ISO/IEC 24752-2.

The values (except for stream types) of the socket variables, and their components, if any, shall be synchronized between the socket and the URCs that participate in a joint session with the socket.

NOTE For variables with stream types, the stream may or may not be managed by the socket.

#### 5.6.5.2 Undefined values

A target may, at any time, set a variable's value to be undefined. Information on undefined values shall be included in the synchronization information from the target to the URC.

NOTE Variable's values may be undefined for various reasons. For example, they may have failed to initialize properly. Or they may reflect features that are not available on a particular target instance (see 7.6.5.3).

#### 5.6.5.3 Unavailable variables

The value of an unavailable variable shall be undefined. Information on the availability of variables shall be included in the synchronization information from the target to the URC.

A target shall not change the availability status of any of its variables during runtime.

NOTE This allows URC to check for the availability of a variable at the beginning of a session only.

A target shall ignore a URC's request to set the value of an unavailable variable.

#### 5.6.5.4 URC initiated value change

A URC that has a connection with a socket may request to change the value of a socket variable. The target may accept or reject changes given by connected URCs. If the URC's change is accepted the target shall respond by propagating the new value to all connected URCs that share the pertaining variable. If the change is rejected, the target shall notify the URC from whom the change was requested.

NOTE It is necessary to make it possible for the socket to reject a variable change from the URC. This is because of network uncertainties (delays) and security reasons. Pertinent use cases include that the variable is not writeable at the time of the change but the URC has not yet received a state change from the socket that would make the variable's write dependency false. Or multiple URCs are changing one variable at the same time in which case it is up to the socket to decide which value to accept.

#### 5.6.5.5 Target initiated value update

The target shall update the values of a socket's variables and propagate them to the connected URCs.

NOTE Nothing in the above should be construed as implying that 'synchronous' or blocking remote method calls are used to synchronize the values of socket variables between URC and target. The synchronization of multiple socket variables may proceed concurrently and asynchronously, subject only to the target respecting constraints in the socket description and the URC respecting values returned from the target.

#### 5.6.5.6 Multiple control sessions

For session-full targets, the socket shall maintain separate sets of socket variable values for each control session, except for values that are shared across sessions (e.g. the current temperature of an oven).

#### 5.6.5.7 Synchronization of dependency values

A target may provide a mechanism for providing updates on dependency values to connected URCs.

### 5.6.6 Command and synchronization of command state

#### 5.6.6.1 General

A target shall support command invocation requests from a URC and synchronization of command states.

Socket commands are specified in ISO/IEC 24752-2.

#### 5.6.6.2 Local input parameters

On receiving a URC's request for invocation of a command with local input parameters the target shall receive from the URC the values of all local input parameters that the command contains. This happens in a TUN platform specific mechanism.

No other synchronization of a command's local input parameters to the URC or from the URC shall take place.

NOTE Global input parameters (references from a command to variables that serve as input parameters for the command) are not conveyed to the target in this way. Instead, they are synchronized on value changes as specified for variables (see 5.6.5).

#### 5.6.6.3 Local output parameters

##### 5.6.6.3.1 General

On conclusion of a command with local output parameters the target shall send to the URC the values of all local output parameters that the command contains. This happens in a TUN platform specific mechanism.

No other synchronization of a command's local output parameters to the URC or from the URC shall take place.

NOTE Global output parameters (references from a command to variables that serve as output parameters for the command) are not received from the target in this way. Instead, they are synchronized on value changes as specified for variables (see 5.6.5).

#### 5.6.6.4 Command state

A command of type uis:basicCommand or uis:timedCommand has a state which a target shall synchronize only one-way from a socket to the connected URCs, and only at user interface initialization and upon completion of command executions.

Only the target can change the state of the command, and shall ignore any propagated state change from a URC.

The target shall respect the meanings of a command's state values (as specified in ISO/IEC 24752-2) to reflect the result of the previously invoked execution. The initial command state (before any execution occurred) shall be "undefined".

Upon completion of execution, the target shall ensure that all modifications of the command's local and global output parameters are propagated to the URC before or together with the propagation of the command's completion and its new state.

#### 5.6.6.5 Undefined command states

A target may, at any time, set a command's state to be undefined. Information on undefined command states shall be included in the synchronization information from the target to the URC. This applies only to command types that allow for command state information (uis:basicCommand, uis:timedCommand).

NOTE Command states may be undefined for various reasons. For example, they may have failed to initialize properly. Or they may reflect features that are not available on a particular target instance (see 5.6.6.6).

#### 5.6.6.6 Unavailable commands

The state of an unavailable command shall be undefined. Information on the availability of commands shall be included in the synchronization information from the target to the URC.

A target shall not change the availability status of any of its commands during runtime.

NOTE This allows URC to check for the availability of a command at the beginning of a session only.

A target shall ignore a URC's request to execute an unavailable command.

#### 5.6.6.7 Multiple control sessions

The socket shall maintain separate sets of socket command states for each control session (for session-full targets) or connected URC (for session-less targets).

NOTE The reason for not sharing a command's state is that it reflects the result of the last execution of the command as requested from a particular URC.



 

#### 5.6.6.8 Concurrency restriction

While a command of type uis:basicCommand or uis:timedCommand is being processed on the target, the target shall not allow a second invocation of the same command. The target shall indicate this by holding the command's state to inProgress while executing it.

NOTE It should not be assumed that 'synchronous' or blocking remote method calls are used to invoke commands on the target or to synchronize the states of socket commands between URC and target. The synchronization of multiple socket commands may proceed concurrently and asynchronously, subject only to the target respecting constraints in the socket description and the URC respecting state values returned from the target.

#### 5.6.6.9 timeToComplete

A command of type uis:timedCommand has a timeToComplete field that is only valid when the command is in state inProgress. If in state inProgress, the target shall periodically "count down" the value of timeToComplete until the command switches to another state. The value of timeToComplete is a hint from the target to the URC as to how long the estimated time to complete is – no guarantee is provided by the target whatsoever.

#### 5.6.6.10 Request for command execution by URC

The target may or may not execute a command (of any type) when requested so by a URC. A URC requests command execution through TUN platform specific mechanisms, referencing the command by its identifier.

If the target rejects a command execution request, the target shall update the command's state (if any) to "failed" in the pertaining session. Upon completion of a requested command execution, the target shall update the command's state to reflect the result of the invocation in the pertaining session.

### 5.6.7 Notifications and acknowledgment reception

#### 5.6.7.1 General

A target shall support propagation of notification states to the connected URCs, and acceptance of pertinent acknowledgments.

Socket notifications are specified in ISO/IEC 24752-2.

#### 5.6.7.2 States

A notification may have one of the following states: inactive, active, undefined.

#### 5.6.7.3 Undefined notification states

A target may, at any time, set a notification's state to be undefined. Information on undefined notification states shall be included in the synchronization information from the target to the URC.

NOTE Notification states may be undefined for various reasons. For example, they may have failed to initialize properly. Or they may reflect features that are not available on a particular target instance (see 5.6.7.4).

#### 5.6.7.4 Unavailable notifications

The state of an unavailable notification shall be undefined. Information on the availability of notifications shall be included in the synchronization information from the target to the URC.

A target shall not change the availability status of any of its notifications during runtime.

NOTE This allows URC to check for the availability of a notification at the beginning of a session only.

A target shall ignore a URC's acknowledging an unavailable notification.

#### 5.6.7.5 Multiple control sessions

The socket shall maintain separate notification states for each control session (for session-full targets) or connected URC (for session-less targets), except for notifications signifying events that are relevant to all sessions/URCs (e.g. a fire alarm) and need not be explicitly acknowledged by the users.

#### 5.6.7.6 Explicit user acknowledgment

The socket shall accept an acknowledgement from the URC after a notification becomes active. The socket may act on this acknowledgement in any way (e.g. set the notification state to "inactive"), or may ignore it. The socket may resume operation (i.e. set the notification state to "inactive") without receipt of an acknowledgement, even where one is expected.

NOTE All notifications require explicit user acknowledgment except those of type "show". See ISO/IEC 24752-2.

#### 5.6.7.7 Propagation of notification state

A socket shall propagate state information for socket notifications to the connected URC(s), and shall ignore any updates on notification states received from URCs.

#### 5.6.7.8 Stacking of notifications

A socket may activate a notification in a session while another notification is already active in the same session. In this case, the second notification takes priority over the first. The socket shall release notifications in reverse order of their activation.

### 5.6.8 Actual indices for dimensional socket sets/elements

The sets of actual index combinations for every dimensional socket set and socket element, if any, shall be synchronized between the socket and the URCs that participate in a joint session with the socket.

The target should convey to the URC the instances of a dimensional socket set or element in the order that reflects the order of its index values, as defined by the fundamental facet 'ordered' of the index type definition (see XML Schema Part 2: Datatypes). For index types defined by restriction (enumeration types), the order should reflect the order in which the values are enumerated in the definition. If enumeration types are merged by the union operator, the order should follow the order in which the base types appear in the union statement.

NOTE While there is also a recommendation for the URC to present the indexed elements in the described order, the above recommendation relieves the URC from having to sort a long list of initial index combinations. However, the URC is solely responsible for determining the recommended position of indexed values that it receives as updates from the target.

The socket may accept or reject a URC's request to add or remove an actual index on a dimensional socket set or element (see 4.4.5). If it accepts the request it shall propagate the modification to the URC (and all other connected URCs). If it rejects the request it shall notify the URC about the rejection.

In case of the socket accepting a new index, it shall provide initial values for the new element components to the connected URCs. Hereby the target may or may not adopt the URC's proposed values, if included in the request.

### 5.6.9 URC side interpretation of socket element dependencies

The socket shall not rely on the URC doing the interpretation of socket element dependencies. Socket element dependencies are specified in ISO/IEC 24752-2.

A URC may not correctly interpret the dependencies between socket elements, or the current socket state may not be reflected correctly on the URC because of network delays or failures.



 

#### 5.6.10 User response timeouts

##### 5.6.10.1 General

The following requirements apply to timeouts generated by the socket as a result of user inactivity or slow performance. These are referred to as 'user response timeouts'. Timeouts prompted by real-time events, such as the end of an auction, are not subject to these requirements.

##### 5.6.10.2 Using a notification as timeout warning

If the target has a timeout condition other than a time-bound notification, the target should use a notification to warn the user when the timeout condition occurs, and to allow the user to request more time. In such a case, if the user requests more time through the notification, the target shall return to the state of the task the user had reached prior to the timeout.

EXAMPLE If a user runs out of time while filling in a form, receives a timeout notification, and requests more time, then the fields of the form that they had already filled in should retain their values when the notification is dismissed and the user returns to the task.

NOTE In some jurisdictions and for some devices or services, user control of timeout or the ability to extend timeout may be required by law. This International Standard does not itself require this, but does enable targets to meet this if they choose.

##### 5.6.10.3 Timeout for notification

All notify elements that have a user response timeout shall represent their default timeout values by the 'timeout' attribute, indicating the default time allowed for the user to respond to the notification before the target will dismiss it.

The value shall be at least 10 seconds.

##### 5.6.10.4 Support for timeout extension request on notification

When a user response timeout is indicated by a notification, the target should allow the URC to extend the timeout up to 5 times the default timeout, even while the notification is active.

NOTE The mechanism for timeout extension requests between a URC and a target is out of scope for this International Standard.

#### 5.6.11 Providing dynamic atomic resources

The target may provide dynamic atomic resources for socket elements, or socket-internal and external types. Dynamic atomic resources may change at runtime which requires synchronization with the URC during a control session.

### 5.7 Target-URC network link (TUNL) on the target

#### 5.7.1 General

The target-URC network link on the target is the link from the target to the target-URC network. A target can support multiple TUNLs allowing it to function with different network technologies and platforms. See 7.2 for further details on target-URC networks.

#### 5.7.2 Requirement

Every target shall provide at least one TUNL. Each TUNL is specific to the particular networking and connection technologies that are used.

EXAMPLE "UPnP on Bluetooth TUNL", or "Jini on 802.11b TUNL", etc.