INTERNATIONAL STANDARD

**ISO/IEC 24752-1**

First edition
2008-02-15

# Information technology — User interfaces — Universal remote console —

## Part 1: **Framework**

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

*Partie 1: Cadre général*



Reference number
ISO/IEC 24752-1:2008(E)



ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

 

# Contents

Page

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

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

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

— *Part 1: Framework*

— *Part 2: User interface socket description*

— *Part 3: Presentation template*

— *Part 4: Target description*

— *Part 5: Resource description*

# Introduction

This part of ISO/IEC 24752 is one of a set of standards to facilitate operation of information and electronic products through remote and alternative interfaces and intelligent agents. The purpose of this part of ISO/IEC 24752 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-based 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 [laptops, personal digital assistants (PDAs), telecommunications/wireless application protocol (WAP) devices (e.g. cell phones), etc.]. 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 behavior of universal remote controls today, except

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 PDA 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. (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 ISO/IEC 24752.

A more detailed overview of the URC framework is provided as Annex A.



 

# Information technology — User interfaces — Universal remote console —

## Part 1: Framework



## 1 Scope

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

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 implement

- at least one target-URC network link as specified in 5.5,
- the URC requirements as specified in Clause 5.

Table 1 summarizes the requirements on URCs, as specified in detail in Clause 5.

**Table 1 – Summary of URC requirements**

| Requirement ("A URC shall …") | See subclause |
|---|---|
| Retrieve documents from a target, including recognition of MIME types | 5.2.3 |
| Interpret a target description so that it can identify a target and open a control session with one of its sockets | 5.2.4 |
| Support an open session request to a target | 5.3.2 |
| Support a URC close session event to a target | 5.3.5 |
| Support an abort session event from a target | 5.3.6 |
| Track connection status information from the underlying network (TUN) | 5.3.7 |
| Synchronize values of socket variables | 5.4.2 |
| Request invocation of a socket command, including support for local parameters and command state updates | 5.4.3 |

| Requirement ("A URC shall …") | See subclause |
|---|---|
| Receive and acknowledge notifications, including support for stacking notifications and their states | 5.4.4 |
| Synchronize actual indices of socket sets and elements | 5.4.5 |
| Support timeout variables and timeout constants | 5.4.7 |
| Provide at least one target-URC network link (see 8.2 for TUN requirements) | 5.5.1 |
| Support reception and updating of atomic resources at runtime for those socket elements that come with atomic resources at runtime | 5.5.2 |
| Provide a concrete user interface for a control session with a target's socket | 5.7 |
| Implement the security and privacy functions available from the implemented TUNs | 9.2 |

## 2.2 Target

A conforming target shall implement:

- at least one target-URC network link as specified in 6.7;
- a target description, as specified in 6.1.4;
- one or more sockets that, when considered together, cover the full functionality of the target, as specified in 6.2;
- the target resources required to conform in at least one natural language (see 6.4.5); and
- the target components and requirements as specified in Clause 6.

Alternatively, a target's manufacturer may provide the above documents 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).

Table 2 summarizes the requirements on targets, as specified in detail in Clause 6.

**Table 2 – Summary of target requirements**

| Requirement ("A target shall …") | See subclause |
|---|---|
| Have an instance identifier | 6.1.2 |
| Provide a fetch mechanism for its documents to be retrieved by URI, including support for MIME types | 6.1.3 |
| Provide exactly one target description with references to all socket descriptions and resource sheets | 6.1.4 |
| 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 | 6.2.2 |
| Inside a target's socket:<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 | 6.2.3 |

| Requirement ("A target shall …") | See subclause |
|---|---|
| Provide a user interface socket description for each of the target's sockets | 6.3 |
| Provide the required target resources in at least one natural language:<br>• one grouping resource for every socket of the target<br>• label resources (textual) | 6.4.5 |
| Support an open session request from a URC | 6.5.1 |
| Support a suspend session request from a URC | 6.5.2 |
| Support a resume session request from a URC | 6.5.3 |
| Support a close session event from a URC | 6.5.4 |
| Send an abort session event in case of user session abortion | 6.5.5 |
| Track connection status information from the underlying TUN network | 6.5.6 |
| Send a session forward event to the URC in case of session forwarding | 6.5.7 |
| Create and maintain a session between a socket and the URC after successful open session request | 6.6.1 |
| Indicate to the URC the availability of socket elements at runtime | 6.6.3 |
| Synchronize the socket variables between the socket and the URCs that participate in a joint session with the socket | 6.6.5 |
| Support command invocation requests from a URC (including handling of local parameters) and synchronization of command states | 6.6.6 |
| Support propagation of notification states to the connected URCs, and acceptance of pertinent acknowledgments | 6.6.7 |
| Synchronize actual indices of socket sets and elements | 6.6.8 |
| Not rely on the URC doing the interpretation of socket element dependencies | 6.6.9 |
| Provide general timeout variables and timeout variables for notify elements to represent user response timeouts implemented | 6.6.10 |
| Provide atomic resources at runtime for those socket elements that are marked to come with atomic resources at runtime | 6.6.11 |
| Provide at least one target-URC network link (see 8.2 for TUN requirements) | 6.7 |



## 3 Normative references

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

ISO/IEC 24752-2, *Information technology — User interfaces — Universal remote console — Part 2: User interface socket description*

ISO/IEC 24752-3, *Information technology — User interfaces — Universal remote console — Part 3: Presentation template*

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

ISO/IEC 24752-5, *Information technology — User interfaces — Universal remote console — Part 5: Resource description*

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

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

## 4 Terms and definitions

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

**4.1**
**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 and keyword resources.

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

**4.1.1**
**supplemental atomic resource**
supplemental resource that is used as an atomic entity in the construction of a concrete user interface

**4.1.2**
**target atomic resource**
target resource that is used as an atomic entity in the construction of a concrete user interface

**4.2**
**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 See ISO/IEC 24752-2.

**4.3**
**command parameter**
variable whose value is used for the execution of a command

NOTE See ISO/IEC 24752-2.

**4.3.1**
**input parameter**
variable whose value is read by the target before execution of a command, to affect the execution and its result(s)

NOTE See ISO/IEC 24752-2.

**4.3.2**
**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 See ISO/IEC 24752-2.

**4.3.3**
**input-output parameter**
variable used as input and output parameter for the same command

NOTE See ISO/IEC 24752-2.

**4.3.4**
**global parameter**
reference from a command to a variable that serves as an input or output parameter for the command

NOTE See ISO/IEC 24752-2.

**4.3.5**
**local parameter**
input or output parameter that is attached to a command

NOTE See ISO/IEC 24752-2.

**4.4**
**constant**
element of a socket description representing fixed or constant information that is known before runtime

EXAMPLE model number

NOTE See ISO/IEC 24752-2.

**4.5**
**connection**
association established between functional units for data transmission

[ANSDIT]

**4.6**
**context of use**
use context
users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used

[ISO 9241-11:1998]

**4.7**
**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

**4.8**
**device**
physical device with a built-in user interface that can also be controlled electronically

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

**4.9**
**discovery**
process by which a URC locates and connects to targets in its environment

**4.10**
**discovery phase**
time period during which a URC scans the environment for available targets and identifies their sockets

**4.11**
**element**
fundamental logical unit of an XML document

**4.12**
**extensible markup language**
XML
meta-markup language that provides a format for describing structured data

NOTE XML has no fixed tag set or application semantics.

**4.13**
**grouping resource**
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 See ISO/IEC 24752-5.

**4.13.1**
**supplemental grouping resource**
grouping resource that is provided externally to the target, by resource services

**4.14**
**interface generator**
software that generates a user interface for a target that is appropriate for a known context of use

NOTE In the context of ISO/IEC 24752, interface generation is typically based on the socket description, atomic resources and presentation templates provided by the target

**4.15**
**notification**
special state of a target in which normal operation is suspended

EXAMPLE an exception state

NOTE See ISO/IEC 24752-2.

**4.16**
**personal digital assistant**
PDA
pocket-sized computer with communication capability, typically used to store an individual's personal information

**4.17**
**presentation template**
PreT
XML document that contains modality-independent presentation information for a user interface socket

EXAMPLE grouping and ordering of elements

NOTE It is expressed using the presentation template markup language (see ISO/IEC 24752-3).

**4.18**
**presentation template markup language**
PreTML
XML language for specification of presentation templates

NOTE The language is specified in ISO/IEC 24752-3.

**4.19**
**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 See ISO/IEC 24752-5.

**4.19.1**
**target resource**
resource provided by a target

**4.19.2**
**supplemental resource**
resource made available externally to a target

**4.19.3**
**standard resource**
resource in the format specified in ISO/IEC 24752-5

**4.19.4**
**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 When an access key is bound to a command, entering the access key will activate the command.

**4.19.5**
**help resource**
atomic resource intended to be used to provide help to a user of a target

**4.19.6**
**keyword resource**
atomic resource that specifies a keyword pertaining to a referenced element

**4.19.7**
**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.

**4.19.7.1**
**supplemental label**
label resource made available externally to a target

**4.20**
**resource description**
description of a resource in terms of its properties

NOTE A resource description is described in the Resource Description Framework (RDF) format. See ISO/IEC 24752-5.

**4.21**
**Resource Description Framework**
RDF
specification being developed by the W3C to provide an infrastructure to support metadata on the Internet and WWW

NOTE See http://www.w3.org/RDF/.

**4.22**
**resource directory**
hierarchical directory of resources and resource services

NOTE Resource directories may contain or reference resource sheets and other resource directories (as subdirectories), and may reference user interface implementation descriptions and resource services. Resource directories facilitate the selection of appropriate resources and resource services for the URC. Each node in the resource directory tree provides hereby a "scent" of the properties of its direct and indirect content. A resource directory description is based on the Resource Description Framework (RDF). See ISO/IEC 24752-5.

**4.23**
**resource service**
service that provides resources from target manufacturers and any third parties such as URC manufacturers, beyond the target resources

NOTE See ISO/IEC 24752-5.

**4.24**
**resource service description**
description of and reference to a resource service

NOTE A resource service description is given in the Resource Description Framework (RDF) format. See ISO/IEC 24752-5.

**4.25**
**resource sheet**
file or part of a file that contains atomic resource descriptions

NOTE A resource sheet is described in the Resource Description Framework (RDF) format. See ISO/IEC 24752-5.

**4.26**
**resource-URC network**
RUN
network connecting the URC to sources of supplemental resources and user interface implementation descriptions (UIIDs)

NOTE It may employ any networking and connection technologies.

**4.27**
**service**
functionality made available to a user electronically

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

**4.28**
**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

**4.28.1**
**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

**4.29**
**socket**
user interface socket
machine-operable access and control point for a target

NOTE See ISO/IEC 24752-2.

**4.30**
**socket description**
user interface socket description
specification that describes the functions and properties of a socket

NOTE A socket description is expressed in XML with the markup language specified in ISO/IEC 24752-2.

**4.31**
**socket element**
variable, constant, command or notify

NOTE See ISO/IEC 24752-2.

**4.31.1**
**dimensional socket element**
homogenous set of values pertaining to a socket element

NOTE See ISO/IEC 24752-2.

**4.32**
**socket element component**
element component
one value out of a set of values for a dimensional socket element

NOTE See ISO/IEC 24752-2.

**4.33**
**socket set**
set
set composed of socket elements and other sets

NOTE See ISO/IEC 24752-2.

**4.33.1**
**dimensional socket set**
repeating socket set
homogenous collection of sets with different indices

NOTE See ISO/IEC 24752-2.

**4.34**
**target**
device or service that the user wishes to use

EXAMPLES video cassette recorder (VCR), online telephone directory

NOTE See ISO/IEC 24752-2.

**4.35**
**target description**
TD
document containing information on a target that is necessary for discovery of and access to the target and its sockets

NOTE There is one target description per target.

**4.36**
**target instance identifier**
identifier for a target instance that is unique among all targets with the same name

**4.37**
**target resource**
resource that is provided by a target in its local network environment

**4.38**
**target-URC network**
TUN
network connecting the target and the URC, which may employ any networking and connection technologies

NOTE Examples of connection technologies include Ethernet, Bluetooth, and 802.11. Examples of networking technologies include UPnP and Java/Jini.

**4.39**
**target-URC network link**
TUNL
link between a target or a URC and the target-URC network

NOTE Each TUNL is specific to the particular networking and connection technologies that are used.

EXAMPLES a “UPnP on Bluetooth TUNL”, a “Jini on 802.11b TUNL”, or a “Jini on Bluetooth TUNL”

**4.40**
**universal remote console**
URC
device or software through which the user accesses a target

NOTE 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.

**4.41**
**user interface**
UI
means by which a user interacts with a target

NOTE 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.

**4.42**
**user interface implementation description**
UIID
description for implementing a user interface to a target, based on the target’s socket

NOTE May take the form of a standard description of a user interface, or consist of executable code intended for a particular class of URC.

**4.43**
**variable**
socket element representing a value relevant to the target’s user interface that may be varied by the target or the user

NOTE See ISO/IEC 24752-2.

# 5 Universal remote console (URC) requirements

## 5.1 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.

## 5.2 Discovery management

### 5.2.1 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 5.5).

### 5.2.2 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 5.2.4). The International Standard does not define specific ways for the URC to present this information to the user.

### 5.2.3 Retrieving documents from the target

#### 5.2.3.1 General

The URC shall be able to retrieve documents from a target through a TUN specific fetch mechanism (cf. 6.1.2). 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), presentation template (see ISO/IEC 24752-3), resource directory and resource sheet (see ISO/IEC 24752-5).

#### 5.2.3.2 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. If two documents have the same identifier but different modification dates the URC may assume that elements with identical identifiers remained unchanged.

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

### 5.2.4 Interpretation of the target description

Cross-platform information on a target is conveyed in the form of a target description (TD), according to ISO/IEC 24752-4.

NOTE A TD is written in XML and is independent from any natural language.

The URC is not required to interpret all information contained in a TD. However, the URC shall be able to identify a target and open a control session with one of its sockets. This includes the discovery of a socket description which is referenced from the TD.

#### 5.2.5 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 5.5), involving the identifier of the locator element in the TD.

### 5.3 Session management

#### 5.3.1 General

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

#### 5.3.2 URC open session request

##### 5.3.2.1 General

A URC shall support an open session request to a target, identifying the socket by URI as specified in the TD. 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 6.5.7).

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 8.2).

##### 5.3.2.2 URC preferences

The URC may indicate a set of preferences (coded as string in UCS according to ISO/IEC 10646) with the open session request.

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

##### 5.3.2.3 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).

##### 5.3.2.4 Target rejects request

If the target rejects the request, the URC should inform the user in an appropriate way.

##### 5.3.2.5 Target forwards request

If the target forwards the URC to another socket, 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 TD of the socket's target. In addition, TUN platforms may use specific forwarding mechanisms to expedite the forwarding process.

### 5.3.3 URC suspend session request

#### 5.3.3.1 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 5.3.4), if it hasn't timed out.

#### 5.3.3.2 Suggested timeout

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

#### 5.3.3.3 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.

### 5.3.4 URC resume session request

#### 5.3.4.1 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.)

#### 5.3.4.2 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.

### 5.3.5 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 6.5.4.2).

### 5.3.6 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.

### 5.3.7 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.

### 5.3.8 Support for session forwarding

A 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 5.3.2) 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 5.3.3) if it has received an abort session event (see 6.5.5) already from the target.

## 5.4 Socket management

### 5.4.1 General

After a successful open session request by a URC, a session is created and maintained between a target's socket and the URC.

A URC may have multiple sessions open at the same time if connected to multiple sockets (on the same target or multiple 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 5.5), 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 (8.2) 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.

### 5.4.2 Synchronization of variables and calculation of variable values

#### 5.4.2.1 General

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

NOTE A target 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).

#### 5.4.2.2 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.

#### 5.4.2.3 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.

#### 5.4.2.4 Undefined values

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

A URC shall not request that a variable's value be set from undefined to a defined value.

As long as a socket element's value is undefined, the URC should not present to the user any user interface object that is mapped to the variable.

NOTE 1 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.

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

#### 5.4.2.5 Regarding 'calculate' attribute

The URC may or may not make use of a calculate attribute attached to a socket variable. If it makes use of it, it can improve response times on the client. However, the URC shall not propagate a change to the socket if the change is triggered by the calculate attribute.

In any case, the target is responsible for updating the value of the variable and propagate changes to the connected URCs. In case of a conflict the target-provided value takes precedence over the client-calculated value.

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.

### 5.4.3 Command and synchronization of command state

#### 5.4.3.1 General

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 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.

#### 5.4.3.2 Acceptance/rejection

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

#### 5.4.3.3 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 5.4.2).

#### 5.4.3.4 Local output parameters

##### 5.4.3.4.1 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 5.4.2).

##### 5.4.3.4.2 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).

#### 5.4.3.5 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.

#### 5.4.3.6 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.

#### 5.4.3.7 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).

As long as a command's state is undefined, the URC should not present to the user any user interface object that is mapped to the command.

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

### 5.4.4 Notification receipt and acknowledgment

#### 5.4.4.1 General

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.

NOTE A target may 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.

#### 5.4.4.2 Explicit user acknowledgment

The target can require that a notification be explicitly acknowledged by the user (see ISO/IEC 24752-2, <explicitAck> dependency). In this case, 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.

#### 5.4.4.3 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 section 5.3.5) before aborting the session.

### 5.4.5 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.

#### 5.4.6 Support for socket element dependencies

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

This includes the interpretation of the dependency values (as XPath expressions) upon opening a session and whenever their value changes within a session. 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.

#### 5.4.7 User response timeouts

##### 5.4.7.1 General

A URC shall support timeout variables and timeout constants.

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'.

##### 5.4.7.2 Timeout variables and constants

Values for user response timeouts are negotiated between a target and a URC by the means of timeout variables and constants, as defined by ISO/IEC 24752-2 ('timeout' attribute for variables and constants).

Timeout variables and constants hold the value of the timeout duration of one or multiple user response timeouts of the socket. They do not represent count-down timers indicating the remaining time to the next timeout event.

##### 5.4.7.3 Updating a timeout variable

The URC may present the value of a timeout variable/constant to the user. The URC may update a timeout variable according to an internal user model or provide a mechanism for the user to set the value. The synchronization of a timeout variable follows the rules for variable synchronization (see section 5.4.2). This means that a URC's update of a timeout variable may be rejected by the target.

NOTE Under this scheme, the URC can sometimes request longer timeouts in advance of the timeout actually occurring, but the target has ultimate control of when a timeout period is up, and whether to offer the user the option of requesting more time at the end of that period.

### 5.5 Target-URC network link on the URC

#### 5.5.1 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 8.2 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.

#### 5.5.2 Support for atomic resources provided by the target at runtime

The URC shall support reception and updating of atomic resources at runtime for those socket elements that come with atomic resources at runtime (includesRes="true", see ISO/IEC 24752-2).

Atomic resources of those socket elements may change at runtime.

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

#### 5.6.1 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 label; 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".

#### 5.6.2 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 <ResSvcDesc> element in a TD, as specified in ISO/IEC 24752-5.

b) The link to a resource service is hard-coded into the URC, or the URC retrieves it from a 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.

### 5.7 User interface generation

#### 5.7.1 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).

#### 5.7.2 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.

#### 5.7.3 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.

#### 5.7.4 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.

#### 5.7.5 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.

#### 5.7.6 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.

### 5.8 Security and privacy requirements

See 9.2.

## 6 Target components and requirements

### 6.1 Discovery management

#### 6.1.1 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.

### 6.1.2 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 8.2.2), 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.

### 6.1.3 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), presentation template (see ISO/IEC 24752-3), resource directory and resource sheet (see ISO/IEC 24752-5).

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.

### 6.1.4 Target description

A target shall provide exactly one target description (TD), independent of the number of sockets the target provides. The format of a target description is specified in ISO/IEC 24752-4.

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, through its resource directory, all resource sheets containing the required target resources (see 6.4.4).

NOTE 1 The target description is an XML 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 TD is written in a TUN platform independent manner (except for platform-specific mapping information which may be included). An example target description for a digital thermometer is provided in Annex C.1.

NOTE 3 The TD is independent of natural languages. target atomic resources (see 6.4.2) and/or supplemental atomic resources need to be employed so that relevant parts of its content can be rendered to a user.

### 6.1.5 Locator functions

Locator functions that are specified in the target description can be invoked during the discovery and control phases by a URC in a manner that is specific to the TUN. The identifier of the <locator> element is used to identify the specific locator function.

## 6.2 User interface socket

### 6.2.1 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 ore more TUN platforms through target-URC network links.

### 6.2.2 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 providing a specific format for the interface (e.g. a visual interface, a verbal interface, an interface optimized for a PDA, etc).

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

### 6.2.3 Socket elements

#### 6.2.3.1 General

A socket contains variables, constants, 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) *Constant* elements represent fixed or constant information known before runtime, such as a safety notice or model number.

c) 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.

d) The *notifications* are special states where normal operation is suspended, such as an exception state. *Notifications* are triggered by the target. Examples include an announcement made by a public address system in an airport, a clock alarm, or a response to invalid input for a field of a form. A 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.

#### 6.2.3.2 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.

#### 6.2.3.3 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).

#### 6.2.3.4 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.

### 6.3 User interface socket description

#### 6.3.1 General

The target manufacturer shall provide a user interface socket description for each of the target's sockets. 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.

The socket description shall conform to ISO/IEC 24752-2, which defines an XML based language for providing socket descriptions for particular targets. An example socket description for a digital thermometer is provided in C.2.

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.

### 6.4 Target resources

#### 6.4.1 Target grouping resources

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

Except for the grouping resource which shall be provided for conformance (see 6.4.4), 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).

#### 6.4.2 Target atomic resources

#### 6.4.2.1 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 6.1.2). 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.

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

#### 6.4.2.2 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 6.4.4), any number of labels may be provided by the target manufacturer.

#### 6.4.2.3 Keywords

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

#### 6.4.2.4 Help

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

#### 6.4.2.5 Access keys

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

#### 6.4.2.6 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>, <location>, <locator> and <socket> elements;
- In the socket description: for the root <uiSocket> element; for the <set> element; for <variable>, <constant>, <command> and <notification> elements; and for types and enumerated values;
- In the resource sheet: for <Group> elements;
- For any UIID element that has an identifier.

#### 6.4.2.7 Target atomic resources for <location> element

For targets that have a physical location, the content of the target atomic resources that reference the <location> element of the target description should be updated when the target is moved.

EXAMPLE When a device is purchased and installed in a home/office/hotel, the label and help resources that reference the <location> element should be updated.

### 6.4.3 Target resource sheets

Target grouping resources and target atomic resources are specified in *resource sheets,* as defined in ISO/IEC 24752-5. An example resource sheet for a digital thermometer is included in Annex C.4.

A target resource sheet should remain stable as much as possible. A target resource sheet may change between sessions only if its identifier (URI) or its modification date is changed. If a target resource sheet has multiple versions with the same identifier (but different modification dates), changed elements (grouping resources and atomic resource descriptions) should have a new identifier assigned.

### 6.4.4 Required target resources

#### 6.4.4.1 General

The following label resources (see 6.4.3) 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.

NOTE 1 A text label is defined as part of an atomic resource description, with 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 being an unformatted character sequence , as defined in ISO/IEC 24752-5.

c) A text label for every <socket> element in the target description.

NOTE 2 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.

e) A text label for every <variable>, <constant>, <command> and <notify> element in the socket description for every socket.

f) A text label for every type in the socket description for every socket if the type name is not human understandable.

g) A text label for every enumerated value in the socket description for every socket if the enumerated value is not human understandable.

h) A text label for every <Group> element in the required grouping resources (see a above).

NOTE 3 All resource sheets containing a required target resource have to be referenced from the target description and its resource directory (see 6.1.4).

NOTE 4 In case of a socket element being marked with includesRes="true" (see ISO/IEC 24752-2), its label can only be delivered by the target at runtime. These labels are considered part of the required target resources and need to be provided at runtime for conformance (cf. 6.4.5).

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.

#### 6.4.4.2 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.

### 6.4.5 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 6.4.4) 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 that come with atomic resources at runtime only (see ISO/IEC 24752-2, attribute 'includesRes' for socket elements), 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.

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

#### 6.4.6.1 General

UIIDs are specific descriptions for implementing an interface to a socket. They are described more fully in 7.9. 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 7).

#### 6.4.6.2 Target presentation templates

A presentation template, or PreT, is a form of user interface implementation description that is modality-independent, and whose format is defined ISO/IEC 24752-3. A PreT provides information on presentation of a socket description.

NOTE 1 The purpose of a *presentation template* is to provide a URC with hints to help it to build a user interface for the target that is easy for a user unfamiliar with that target to navigate in order to discover the available functionality and operate the target. The hints are of an abstract nature, and are intended to apply to any delivery context. These hints primarily provide information on structuring, grouping and linearization of the socket elements. Elements within a presentation template may be referenced by target atomic resources and supplemental atomic resources as described in ISO/IEC 24752-5. An example of a presentation template for a digital thermometer is given in Annex C.3.

NOTE 2 The PreT is not required from the target manufacturer.

## 6.5 Session management

### 6.5.1 Support for URC open session request

#### 6.5.1.1 General

A target shall support an open session request from a URC.

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 6.5.7). The open session request and the target's response are specific to a particular networking platform.

#### 6.5.1.2 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.

#### 6.5.1.3 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 TD 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.

### 6.5.2 Support for URC suspend session request

A target shall support a suspend session request from a URC. It responds to a URC suspend session request (see section 5.3.3) by either one of the following:

- Reject request.
- 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.

### 6.5.3 Support for URC resume session request

A target shall support a resume session request from a URC. It responds to a URC resume session request (see section 5.3.4) 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.)

### 6.5.4 Support for URC close session event

#### 6.5.4.1 General

A target shall support a close session event from a URC. It responds 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.

#### 6.5.4.2 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.

### 6.5.5 Abort session event

#### 6.5.5.1 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.

#### 6.5.5.2 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 example 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 5.4.7 and 6.6.10.

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.

### 6.5.6 Established connection

#### 6.5.6.1 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.

#### 6.5.6.2 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.

### 6.5.7 Session forwarding

#### 6.5.7.1 General

A 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.

#### 6.5.7.2 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).

#### 6.5.7.3 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.

#### 6.5.7.4 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 5.3.2) to the target it is being forwarded to. 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, the target may abort the old session by sending an abort session event to the URC (see 6.5.5).

#### 6.5.7.5 Forwarding within the same target

For session forwarding within the same 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.

### 6.6 Socket management

#### 6.6.1 General

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

#### 6.6.2 A socket’s sessions

A 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.

#### 6.6.3 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. 8.2.5.2).

NOTE “Optional socket elements” are socket elements that have an ‘optional’ attribute with value “true” (see ISO/IEC 24752-2).

### 6.6.4 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 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.

### 6.6.5 Variable and synchronization of variable values

#### 6.6.5.1 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.

#### 6.6.5.2 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.

A target shall ignore a URC's request to set a variable's value from undefined to a defined value.

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.

#### 6.6.5.3 URC initiated value change

A URC that has a session 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.

#### 6.6.5.4 Target initiated value update

The target shall update the values of a socket's variables and propagate them to the connected URCs. This includes variables that have a calculate dependency in the socket description.

NOTE 1 The socket will not receive value change requests from connected URCs that are based on the execution of the calculate attribute (see 5.4.2.4).

NOTE 2 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.

#### 6.6.5.5 Multiple control sessions

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).

### 6.6.6 Command and synchronization of command state

#### 6.6.6.1 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.

#### 6.6.6.2 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 6.6.5).

#### 6.6.6.3 Local output parameters

##### 6.6.6.3.1 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 6.6.5).

#### 6.6.6.4 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.

#### 6.6.6.5 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.

#### 6.6.6.6 Multiple control sessions

The socket shall maintain separate sets of socket command states for each control session.

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.

#### 6.6.6.7 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.

#### 6.6.6.8 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.

#### 6.6.6.9 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.

### 6.6.7 Notifications and acknowledgment reception

#### 6.6.7.1 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.

#### 6.6.7.2 States

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

#### 6.6.7.3 Multiple control sessions

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

#### 6.6.7.4 Explicit user acknowledgment

A notification requires explicit user acknowledgement if its <explicitAck> dependency evaluates to true at the time of the notification being active (or stacked). The socket shall not change its state in any way that would change the value of the <explicitAck> dependency while the notification is active (or stacked).

The socket shall accept an acknowledgement from the URC after the 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.

#### 6.6.7.5 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.

#### 6.6.7.6 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.

### 6.6.8 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 socket may accept or reject a URC's request to add or remove an actual index on a dimensional socket set or element (see 5.4.5). 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.

### 6.6.9 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.

### 6.6.10 User response timeouts

#### 6.6.10.1 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.

#### 6.6.10.2 General timeout variable

A socket variable or constant of type xsd:duration may have the attribute timeout="true", indicating that this is a timeout value. All user response timeouts implemented by the target shall be represented by such an element, although a single element may cover a number of actual timeouts. The variable should indicate the range of allowable values for the timeout, and should be a variable that allows adjustment up to 5 times the default value. The URC (with or without explicit user input) can adjust the variable values as desired.

NOTE Timeout variables (or constants) hold the value for the timeout duration of one or multiple user response timeouts of the socket. They do not represent count-down timers indicating the remaining time to the next timeout event.

#### 6.6.10.3 Timeout variable for notify element

A notify element in the socket may have a subelement that is a variable or constant with attribute timeout="true", as described above. All notify elements that have a user response timeout shall represent the timeout with such an element. This element indicates the time allowed for the user to respond to the notification before the target will dismiss it. The value shall be either at least 10 seconds or adjustable up to at least 10 seconds.

When a user response timeout is indicated by a notification, the notification should offer the user the opportunity to request more time (by allowing the user to activate a command or set a variable through a dialog while the notification is active). If the user requests more time, 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.

### 6.6.11 Providing atomic resources at runtime

The target shall provide atomic resources at runtime for those socket elements that are marked to come with atomic resources at runtime (includesRes="true", see ISO/IEC 24752-2).

Atomic resources of those socket elements may change at runtime which requires synchronization with the URC during a control session.

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

### 6.7.1 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 8.2 for further details on target-URC networks.

### 6.7.2 Requirement

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

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

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

## 6.8 Security and privacy requirements

See 9.3.

# 7 Supplemental resources

## 7.1 General

*Supplemental resources* are any kind of information that is provided externally to a target, by resource services. Supplemental resources may supplement or replace the target resources, and may be provided by the target manufacturer, a URC manufacturer, or 3rd parties. The URC can retrieve supplemental resources for a specific target from any resource service.

Supplemental resources may be used by URCs in constructing a user interface. They may also be used by people designing user interface implementation descriptions in constructing UIIDs for particular URC products.

## 7.2 Third party supplemental resources

Supplemental UIIDs can be prepared and distributed by third parties.

EXAMPLE Anyone could create an alternate resource for a given product and make it available on the Internet. Thus, someone who is interested could create a very user friendly interface (UIID) for a particularly popular home stereo system that would run on a PDA and make it available on the Internet. Anyone who wanted to could download the UIID and use it in connection with the product in somewhat the same fashion as people can purchase universal remote controls and use them to control appliances today.

## 7.3 Supplemental resources are optional

The availability of supplemental resources is not required for the use of a URC within the URC framework. Supplemental resources are optional, additional resources. It is also important to note that there is no particular location that the supplemental resources shall be stored.

EXAMPLE Supplemental resources could be stored on the manufacturer's website, on a common public site or on private proprietary sites. They could be stored at any other network-accessible location, and be retrieved by a resource service for use by the URC. They could be provided free of charge or they can be value added products that are designed and sold by others

## 7.4 Format of supplemental resources

Although supplemental resources may take any form and be proprietary or public, ISO/IEC 24752-3 and ISO/IEC 24752-5 provide standard formats for supplemental resources to facilitate their use by URCs and UIID builders.

Target and URC manufacturers are free to define additional forms of supplemental resources if these do not conflict with the supplemental resource categories that are defined by ISO/IEC 24752-3 and ISO/IEC 24752-5. In particular, target and URC manufacturers may use both standard and proprietary resources in specialized URC-specific user interface implementation descriptions (UIIDs).

## 7.5 Forms of resource services

In general, resource services providing supplemental resources can take a wide variety of forms. They include

- services providing any form of UIID, i.e. any user interface component that plugs into a target's socket;
- services providing alternate labels, including images, animations, and sounds;
- services providing semantic information for the functions of a target;
- services which will translate an arbitrary set of labels from one language to another;
- inference engines that can help to interpret ambiguous commands from a user into the matching control on a device;
- meta-databases;
- thesauruses;
- dictionaries;
- speech recognition engines;
- natural language processing engines;
- translation services;
- icon libraries;
- services providing common control set (e.g. the basic play controls for a VCR or CD), etc;
- services available locally or remotely via network connection (either real time or downloadable);
- etc.

## 7.6 Supplemental resource sheets

Supplemental grouping resources and supplemental atomic resources are specified in *resource sheets,* as defined in ISO/IEC 24752-5. An example resource sheet for a digital thermometer is included in Annex C.4.

A supplemental resource Sheet should remain stable as much as possible. A supplemental resource sheet may change between sessions only if its identifier (URI) or its modification date is changed. If a supplemental resource Sheet has multiple versions with the same identifier (but different modification dates), changed elements (grouping resources and atomic resource descriptions) should have a new identifier assigned.

## 7.7 Supplemental groupings

*Supplemental grouping resources* are grouping resources that are provided externally to a target, by resource services. Grouping resources and their format are specified in ISO/IEC 24752-5.

A URC may use supplemental grouping resources when constructing a user interface for a target's socket.

## 7.8 Supplemental atomic resources

*Supplemental atomic resources* are atomic resources that are provided externally to a target, by resource services.

The following supplemental atomic resources currently have standardized forms defined by ISO/IEC 24752-5: label, keywords, help and access key resources. According to ISO/IEC 24752-5, atomic resources, including supplemental atomic resources, are described in resource sheets, using RDF syntax. For more information on standardized atomic resources see ISO/IEC 24752-5.

## 7.9 Supplemental user interface implementation descriptions (UIIDs)

### 7.9.1 General

UIIDs are specific descriptions for implementing a user interface for a target, based on a target's socket. A supplemental UIID is a UIID that is provided externally to a target, by resource services. Supplemental UIIDs may be in the format specified by ISO/IEC 24752-3 (presentation template), or other public or proprietary formats.

EXAMPLE The manufacturer for a home audio-visual system might create UIIDs which were optimized to run on their own special Remote Consoles, on popular PDAs (such as Palm or Pocket PC), or to run on laptops and personal computers. These UIIDs could take the form of user interfaces described in a standard fashion or could contain executable code intended to only run on a particular device (URC). Companies could choose to create some standard generic UIIDs and some specific compiled UIIDs.

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

A supplemental UIID may use supplemental resources, or alternatively use equivalent information built into the supplemental UIID.

### 7.9.2 Supplemental presentation template

ISO/IEC 24752-3 defines the presentation template, an XML document that contains modality-independent presentation information for a socket. A supplemental presentation template is a subtype of a supplemental UIID.

# 8 Networks

## 8.1 General

In order for the URC to communicate with the target, or to access and download target resources, there shall be some network connection between them. The network between the URC and the target can be the same as the network with supplemental resources and resource services. But since the network to the supplemental resources is not required and can be non-real-time in nature (supplemental resources could be downloaded to the URC for later use), the network with the target (which shall be real-time) is treated separately.

## 8.2 Target-URC network (TUN)

### 8.2.1 General

This section describes requirements for the networking platform used for target-URC communication applicable to both discovery and session management. This International Standard assumes underlying networking layers (middleware) that provide discovery (and optionally localization) of targets, object distribution and synchronization, programmatic control, and security. Rather than requiring any particular device architecture the International Standard assumes the availability of these services in the network functionality, and states the requirements for them to be used as underlying TUN platform for implementing this International Standard.

The target and the URC device can be connected using any of a variety of methods. They can network using any method that supports the requirements for a TUN. Where a target and URC support different TUN technologies, a gateway may be used to create a seamless target-URC network between them.

EXAMPLE 1 Potential connectivity technologies include infrared, RF, wire line networks, Ethernet, Bluetooth, IrDA, and 802.11.

EXAMPLE 2 Potential networking technologies include Universal Plug and Play (UPnP), and Java/Jini.

#### 8.2.2 Unique instance identifier

The underlying network shall make available the targets' instance identifiers (see 6.1.2) . If a target supports multiple networks, its unique identifier shall be common across all networks.

#### 8.2.3 Support for discovery

##### 8.2.3.1 General

During the discovery phase, the underlying network shall provide information about available targets to a URC. Available targets (services or devices) advertise themselves to the underlying TUN.

##### 8.2.3.2 Scope of discovery

Discovery is usually restricted to a local environment, i.e. the immediate TUN. However, some mechanism for discovering long distance targets may be needed for certain networking platforms.

At a minimum, if a target is available and its address is known, and that address is accessible through the target-URC network, and any network authorization requirements are satisfied, it shall be possible to initiate a control session with the target.

#### 8.2.4 Support for document deployment

##### 8.2.4.1 Fetch mechanism

A TUN shall support a fetch mechanism that allows a URC to retrieve the target description, socket descriptions and target resources by URI. This fetch mechanism may include authentication mechanisms for URC and/or target.

##### 8.2.4.2 Scope of document deployment

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

##### 8.2.4.3 Target description

The underlying network shall have support for providing information about targets, in particular the transmission of the target description, as described in ISO/IEC 24752-4. Discovery might happen in several loops, i.e. not all target properties need to be available by a single request.

##### 8.2.4.4 Target locator functions

The underlying network shall support invoking a target's locator functions from a URC in the discovery or control phase.

#### 8.2.5 Support for session management

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

##### 8.2.5.1 Transmission of documents

In the discovery phase, or at the beginning of a control session, the target provides the necessary information for building a remote user interface via documents (see 6.2).

#### 8.2.5.2 Support for indication of available socket elements at runtime

The target-URC network shall convey a target's indication of available socket elements at runtime for a particular session (see 6.6.3).

#### 8.2.5.3 Support for socket synchronization

The target's state is described in its socket by a set of variables, command states and notification states. When opening a session with a URC, the target transfers these values to the URC. When the target changes state, this change shall be communicated to the URC. Similarly, when the state on the URC changes or a target command is called from the URC, this shall be communicated to the target.

NOTE Events, polling or any other synchronization mechanism may be employed to facilitate the propagation of state changes from the target to the URC and vice versa.

#### 8.2.5.4 Support for synchronization of atomic resources at runtime

The target-URC network shall provide a means of synchronizing atomic resources at runtime for those socket elements that are marked to come with atomic resources at runtime (includesRes="true", see ISO/IEC 24752-2).

Note that atomic resources of those socket elements may change at runtime which requires synchronization with the URC during a control session.

#### 8.2.5.5 Ordering requirements for messages

The target-URC network shall deliver all messages generated by a URC to a target in the order in which they were generated. Similarly, the order of messages from a target to a URC shall be preserved.

#### 8.2.5.6 Preservation of document content

The target-URC network shall transport the documents in such a way that the exact content of each document is preserved.

#### 8.2.5.7 Status of established connection

During the control phase, the underlying network shall provide connection status information, so that a target and a URC can determine whether they are still connected, and react appropriately.

#### 8.2.5.8 Connections to multiple URCs

The TUN should support simultaneous independent connections from a target to multiple URCs.

### 8.2.6 Security and privacy requirements

See 9.4.

## 8.3 Resource-URC network (RUN)

This International Standard does not specify any particular network link or technology be used as RUN. The URC and the resources can be connected using any of a variety of networking methods. It is anticipated that the link will usually be some type of TCP/IP networking technology that connects real-time or occasionally to the Internet.

For security and privacy considerations see 9.4.

# 9 Security and privacy considerations

## 9.1 General

Tasks performed through the URC-target dialog may have serious potential outcomes or involve sensitive data whose disclosure to the wrong parties could seriously harm the interests of either the user or a target service provider or target product vendor.

The processing of computing devices and the information sharing across data networks are subject to compromise with more or less effort depending on the precautions taken.

A target has a relatively uniform population of dialog partners: individuals wielding URC devices. The target has knowledge of the outcomes of its functions and hence can reasonably assess security and privacy requirements and implement protections appropriate to these requirements.

A URC, on the other hand, faces a more diverse population of dialog partners. The security requirements appropriate to different targets will vary considerably. A URC should therefore implement security functions to enable access to high-security, and not just low-security, targets.

EXAMPLE Example scenarios are provided in Annex B.

## 9.2 URC considerations

A URC shall implement the security and privacy functions available from the implemented TUNs in order to interoperate appropriately with targets which elect to implement these functions. Beyond what is available from the implemented TUNs, it is the responsibility of the target to make access available to users with the aid of a user dialog.

NOTE URCs may implement key wallet or certificate management functions to expedite authentication at the application level. These functions, however, are outside the scope of this International Standard.

## 9.3 Target considerations

Target developers should review the tasks that can be performed with their product or service, and implement a security and privacy strategy appropriate to the downside potential of inappropriate disclosure or control through the URC support.

Depending on the privacy and authorization requirements appropriate to these tasks, the target should implement privacy and authentication functions available from the implemented TUNs and supplement these, as necessary, with privacy, authentication and authorization functions such as login implemented as part of the target functionality. This target functionality, per 6.2 above, shall be operable through a socket and meet the other documentation and operation requirements of this International Standard.

## 9.4 Network considerations

Networks should provide security and privacy protections appropriate to the signal exposure of their physical medium and context of use.

# Annex A
(informative)

# Overview of the universal remote console framework

This International Standard specifies communications between a target that a user wishes to access and operate, and a universal remote console (URC) that presents the user with a remote user interface through which they can discover, select, access and operate the target. The URC is a device or software through which the user accesses the target. If the URC is software, it is typically hosted on the user's physical device, but a distributed approach is also possible.

Communications between the target and URC take place over a network, the *target-URC network*. This International Standard does not specify the network protocol to be used, and a target or URC may support any number of appropriate connection protocols. These protocols are used to provide discovery of targets, and to establish and maintain control sessions between URCs and targets. targets and URCs access the target-URC network through *target-URC network links*. Targets support discovery by providing essential information in a *target description.*

Each target provides a *user interface socket* (or short "socket"), or set of sockets, through which a URC can access part or all of the target's internal states and provide control inputs to the target. For each socket, a target provides a *user interface socket description (or short "socket description")* which describes the socket in a machine interpretable manner. Additionally the target provides resources that pertain to the user interface of the target, as viewed through that socket. The socket description and resources are used by the URC to find or generate an appropriate user interface, given the functionality of the target, the nature of the URC device, and the user's interaction preferences.

Interaction between a target and a URC consists of a *discovery phase* and an optional *control phase*. The discovery phase initializes the URC to locate and identify all available targets and their sockets. The *control phase* is the time period during which a target and a URC initiate, maintain and terminate a *control session*. A *control session* is a connection between the URC and a target's socket for the purpose of the URC controlling a functional unit of the target via the socket. When multiple URCs are connected to the same socket, each has an independent control session, although target state values may be shared.

One type of resource is a *user interface implementation description (UIID).* A UIID is a description of a user interface for a particular socket. This description may take an abstract or concrete form, and may be very general or specific to a given URC device class, user profile and task. The UIIDs provide a mechanism by which a manufacturer can provide tuned interfaces for their target sockets which are predefined and optimized to work on particular URCs, for example Pocket PCs or Palm-based PDAs. UIIDs can be provided by a target or by external sources.

This International Standard defines one, very general, form of UIID - the *presentation template.* A presentation template provides general hints as to how to build a usable user interface for a socket, and does not assume any particular class of URC.

Another type of resource is an *atomic resource*. An atomic resource is an object within a user interface that is used as an atomic entity in the construction of a concrete user interface. Atomic resources include labels, help text, access keys and keywords. An atomic resource may be of any form and modality, including text, images, sounds, animations and video clips.

Socket descriptions and UIIDs are documents that are independent of any natural language. In order to render their content to a user, atomic resources are provided (by the target and external to it) in the form of *resource sheets.* A resource sheet is a file that contains descriptions of atomic resources (or *"atomic resource descriptions"*).

A *grouping resource* is a resource that defines a hierarchical structure for the elements of a socket or UIID. Grouping resources are useful for constructing the layout and navigation mechanisms of a particular user interface. Grouping resources are provided in resource sheets.

A URC may take advantage of resources from the target or from a resource service. These are referred to as *target resources* and *supplemental resources* respectively. Supplemental resources can be used by a URC to replace, to supplement, or to help interpret or translate, any resources provided by a target.

In general, resources for building user interfaces may be obtained from the target, stored on the URC, or gathered from the Internet. A URC uses the target-URC network to retrieve target resources. It may use any form of networking or other mechanism to access supplemental resources via a *resource-URC network*. This International Standard does not specify the mechanism by which a URC accesses resources external to the target.

**Table A.1 – Overview of components, their functions and references to further information.**

| **Component** | **Purpose** | **References to clauses in this document and to other standards.** |
|---|---|---|
| target-URC network | network connecting the target and the URC | 8.2 |
| target-URC network link | link between a target or a URC and the target-URC network | 5.5, 6.6.11 |
| target description | information on a target that is necessary for access to the target and its sockets | 6.1.4, ISO/IEC 24752-4 |
| (user interface) socket | machine-operable access and control point for a target | 6.2 |
| (user interface) socket description | specification that describes the functions and properties of a socket | 6.3, ISO/IEC 24752-2 |
| resource | object that is used as an entity or to support decision making in the construction of a concrete user interface | Table A.2, ISO/IEC 24752-5 |
| target resource | resource provided by a target | 6.4.2 |
| supplemental resource | resource made available externally to a target | 7 |
| resource-URC network | network connecting the URC to sources of supplemental resources | 8.3 |

**Table A.2 – Overview of resource types defined by this International standard and ISO/IEC 24752-5**

| Resource Type | Purpose | References to clauses in this document and to other standards |
|---|---|---|
| grouping resource | hierarchical structure of user interface elements | 6.4.1, 7.7, ISO/IEC 24752-5 |
| user interface implementation description (UIID) | description for implementing a user interface to a target, based on the target's socket | 6.4.6, 7.9 |
| presentation template | modality-independent presentation information for a socket description; subtype of UIID | 6.4.6.2, 7.9.2, ISO/IEC 24752-3 |
| atomic resource | atomic entity in the construction of a concrete user interface (e.g. label, help text, access key, keyword, location) | 6.4.2, 7.8, ISO/IEC 24752-5 |
| resource sheet | file containing descriptions of atomic resources (atomic resource descriptions) | 6.4.2, 7.6, ISO/IEC 24752-5 |

## A.1 Major components of the URC framework

Figure A.1 illustrates the major components of the URC framework within the broader context of the URC application. It illustrates both elements for which there are formats specified by the URC framework, and elements that can use these or other standard or proprietary formats. This figure is provided as a conceptual map and is not intended to prescribe a particular architecture. Any alternative architecture may be adopted provided the required functionality is implemented and standard format documents are provided. Technical details for each component are provided in separate documents as referenced in the pertaining sections of this document. See Table A.1 and Table A.2 for a summary of the purpose and reference to the technical descriptions of the major components.

As shown in Figure A.1, the URC architecture can be thought of as comprising four major components and two networks.

a) The Universal Remote Console (URC)

b) The Target

c) supplemental User Interface Implementation descriptions (UIIDs)

d) supplemental Resources

The two networks are

- **Target-URC Network (TUN) -** between the URC and the Target (required)
- **Resource-URC Network (RUN)** – between the URC and sources of supplemental Resources

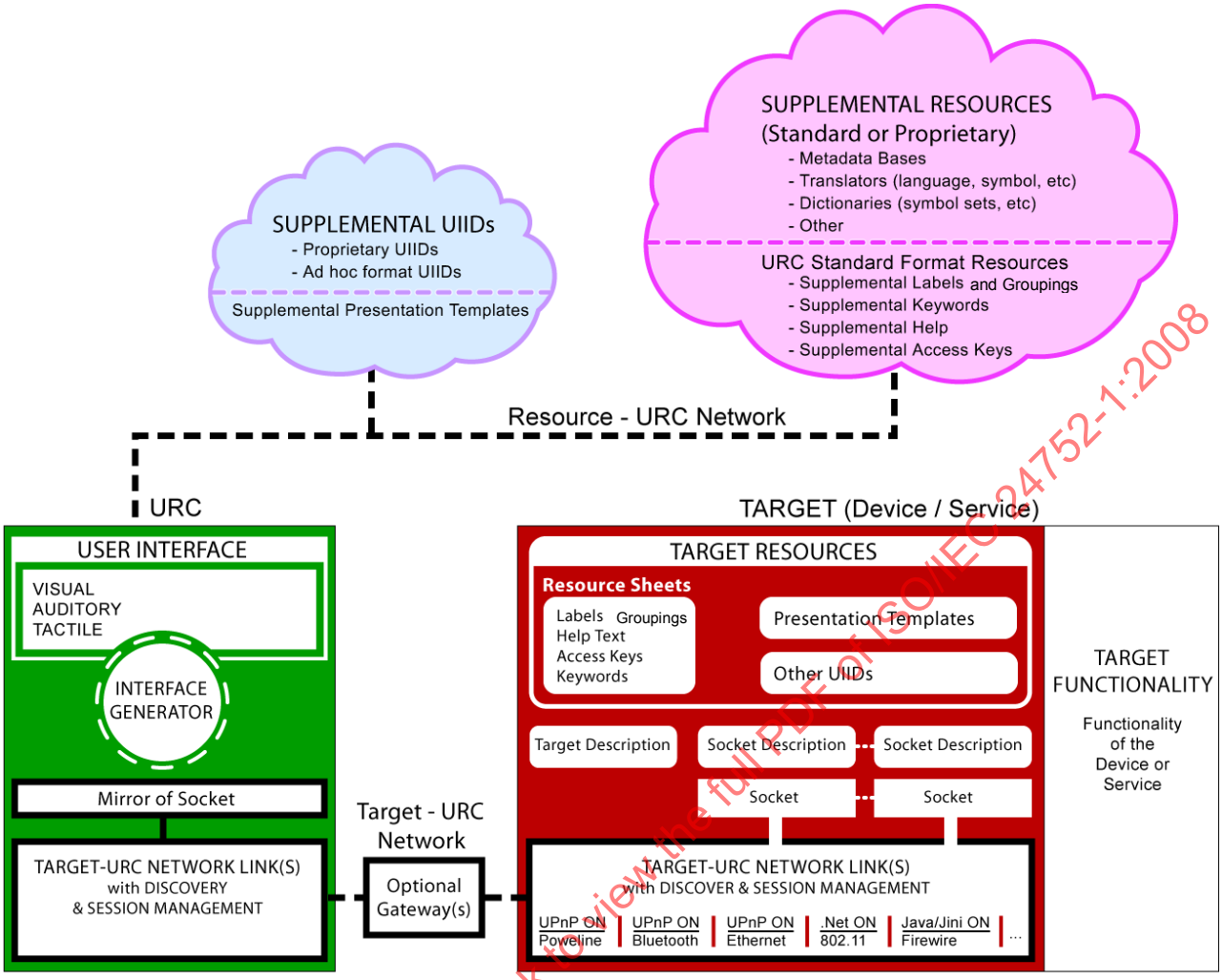


**Figure A.1 – Overall structure and components of the URC framework**

In Figure A.1, documents are represented as rectangles with rounded corners.

NOTE For screen reader users, a long description of Figure A.1 is provided as follows:

The diagram shows a "URC" (Universal Remote Console) which is connected to a "Target (Device/Service)" through a "Target-URC Network," and "Optional Gateway(s)". The URC is also connected through the "Resource-URC Network" to two resource clusters which are shown as clouds at the top of the diagram. The resource clouds are labeled "SUPPLEMENTAL UIIDs" and "SUPPLEMENTAL RESOURCES (Standard or Proprietary)".

The "TARGET" is in the center of the diagram. It is made up of a red box representing the part of the Target that is exposed to the URC, and a box "TARGET FUNCTIONALITY (Functionality of the Device or Service)".

Inside the red box there is (from bottom to top):

- "TARGET-URC NETWORK LINK(S) with DISCOVERY AND SESSION MANAGEMENT", for example "Universal Plug and Play (UPnP) ON Powerline", "UPnP ON Bluetooth", "UPnP ON Ethernet", ".NET ON 802.11", "Java/Jini ON Firewire", ...
- "TARGET DESCRIPTION" (document)
- 1 or more SOCKETS, each of which with a socket description (document)
- "TARGET RESOURCES". The "TARGET RESOURCES" include:
- "Resource Sheets" (documents) with one or more of the following: "Labels", “Groupings”, "Keywords", "Help", "AccessKeys".
- "Presentation Templates" (documents) and
- "Other UIIDs (User Interface Implementation Description)" (documents).

The URC is made up of (from bottom to top):

- "TARGET-URC NETWORK LINK(S) with DISCOVERY AND SESSION MANAGEMENT", connecting to the "TARGET-URC NETWORK LINK(S)" of the Target side through the optional gateway(s).
- A "mirror of socket" that is connected to the "TARGET-URC NETWORK LINK(S)"
- An "INTERFACE GENERATOR" that feeds into the "USER INTERFACE" that may include "VISUAL", "AUDITORY" and "TACTILE" presentation mechanisms.

The resource cloud "SUPPLEMENTAL UIIDs" provides non-V2 UIIDs to the URC:

- "Proprietary UIIDs"
- "Ad hoc format UIIDs"

"SUPPLEMENTAL UIIDs" also provides V2-standardized UIIDs to the URC:

- "Supplemental Presentation Templates"

The resource cloud "SUPPLEMENTAL RESOURCES (Standard or Proprietary)" provides non-V2 resources to the URC:

- "Metadata Bases"
- "Translators (language, symbol, etc.)"
- "Dictionaries (symbol sets, etc.)"
- "Other"

"SUPPLEMENTAL RESOURCES" also provides V2-standardized resources to the URC:

- "Supplemental Labels and Groupings"
- "Supplemental Keywords"
- "Supplemental Help"
- "Supplemental Access Keys"

## A.2 Essential components

Only 3 of the components shown in Figure A.1 are actually needed for a URC to function. The rest are optional. The three essential components are:

- The URC,
- The target, and
- The target-URC network

## A.3 Target URC synchronization

The URC framework assumes a fine-grained two-way synchronization of the target's state between the target and any URC that has a control session open with it. On the target side the sockets are responsible for propagating relevant changes to the connected URCs and to receive requests to change the target's state.

The purpose of the user interface socket is to expose the relevant information about a target so that a user can perceive its state and operate it. There are three types of socket elements whose state is synchronized between a target and a connected URC. The *Variables* are state variables of the target and can typically be changed by the URC. The *Commands* represent function calls on the target that can be invoked by the URC. The *Notifications* propagate special target states in which normal operation is suspended on the target.

## A.4 URC and component interaction

URCs can take many different forms. The following brief examples show how the components of this International Standard interact. The different components of the URC framework and their interactions are then described in more detail in the following sections.