INTERNATIONAL STANDARD

ISO/IEC 24752-6

First edition
2014-12-15

Information technology — User interfaces — Universal remote console —

Part 6: Web service integration

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

Partie 6: Intégration du service web

Reference number
ISO/IEC 24752-6:2014(E)

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

Published in Switzerland

Contents

Page

Foreword ... **iv**

Introduction ... **v**

1 Scope ... **1**

2 Conformance ... **1**

3 Normative references ... **3**

4 Terms and definitions ... **3**

5 Relation to other standards ... **4**
5.1 Relation to XML ... 4

6 Mapping Descriptions ... **4**
6.1 General ... 4
6.2 Mapping a target to a Web service ... 5
6.3 Mapping a socket to a Web service partition ... 5
6.4 Mapping a socket variable ... 6
6.5 Mapping a socket command ... 7
6.6 Mapping a socket notification ... 8
6.7 Mapping a socket-internal type definition to a Web service's type definition ... 8

7 Embedding target description and socket descriptions in a WSDL document ... **9**
7.1 General ... 9
7.2 Restriction on target namespaces of internal schema part ... 9
7.3 Restriction on identifiers within target and socket descriptions ... 9
7.4 'name' attribute values ... 9
7.5 Implicit target description ... 10
7.6 Implicit socket description ... 20
7.7 Resources ... 63

Annex A (informative) **Example documents for a digital thermostat** ... **64**

Bibliography ... **65**

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 35, *User interfaces*.

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

— *Part 1: Framework*

— *Part 2: User interface socket description*

— *Part 4: Target description*

— *Part 5: Resource description*

— *Part 6: Web service integration*

Introduction

Web services are becoming increasingly ubiquitous in the form of public Internet-wide services and private services in protected environments. Even devices and appliances in the digital home are being made network-accessible by exposing them as Web services.

The universal remote console technology allows for pluggable user interfaces for any kind of devices and services, including web services. For a web service to adopt the URC concepts, it needs to expose a composition of user interface socket elements. This can be achieved in two ways: either the Web service provides one target description (see ISO/IEC 24752-4) or one or multiple separate user interface socket descriptions (see ISO/IEC 24752-2). Alternatively, the Web service can provide the target description and the socket description(s) in an "embedded" approach integrated with its Web service interface description. The web service description language (WSDL) defines suitable extension mechanisms for such integration. With this integrated approach, Web services do not need to provide a separate target description and separate socket descriptions. It is sufficient to integrate this information into their WSDL document. It is expected that this approach will help in the adoption of the URC technology for Web services and thus, make personalized and pluggable user interfaces widely available for Web services.

This part of ISO/IEC 24752 defines the syntax and semantics for embedding target description and socket descriptions in interface specifications of Web services so that there is a clear mapping between special elements in the WSDL document and elements of (implicit) target description and (implicit) socket descriptions.

This part of ISO/IEC 24752 lays the groundwork for an integration of the universal remote console framework within Web service environments. It gives rise to various URC-based architectures and implementations, including a middleware approach in which a user interface server provides access to web services, and Web service centric approach in which a Web service exposes a user interface socket via its WSDL-based interface.

Information technology — User interfaces — Universal remote console —

Part 6: Web service integration

1 Scope

This part of ISO/IEC 24752 defines the syntax and semantics for embedding target description and socket descriptions in interface specifications of web services so that there is a clear mapping between special elements in the WSDL document and elements of (implicit) target description and (implicit) socket descriptions.

2 Conformance

A WSDL1 document conforms to this International Standard if it complies to the web services description language (WSDL) 1.1 specification and with the requirements and recommendations in Clause 6 and Clause 7.

A WSDL2 document conforms to this International Standard if it complies to the web services description language (WSDL) 2.0 specification and with the requirements and recommendations in Clause 6 and Clause 7.

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

A Web service conforms to this International Standard if it fulfils the requirements of a conforming target in ISO/IEC 24752-1, in all of the following ways.

— The Web service shall provide at least one service binding (as specified in the Web service's WSDL document) as Target-URC network link.

— The Web service shall have a target name, given as the target namespace of the Web service, as specified in 7.5.2.

— The Web service shall have exactly one target description which shall be embedded in its WSDL document and shall include references to external files containing the target resources (grouping sheets and resource sheets) conforming to at least one natural language, as specified in 7.5.

— The Web service shall provide a fetch mechanism for its target resources (grouping sheets, resource sheets, UIIDs) to be retrieved by URI, including support for MIME types.

— The Web service shall provide a target instance identifier through the 'getTargetInstanceId' operation in the "_target" partition, as specified in 7.5.11.2.

— The Web service shall support locator functions through a "_target" partition, as specified in 7.5.7.

— The Web service shall expose one or more sockets that, when considered together, cover the full functionality of the Web service as a target. For each of these sockets, a socket description shall be embedded in the Web service's WSDL document (as specified in 7.6).

— For each of the Web service's sockets, the socket shall have variables that include all of the dynamic data on the socket's state a user can perceive and/or manipulate and commands that include all of the socket's functions that can be called explicitly or implicitly by users and notifications that cover all exceptions that the Web service needs to inform the user about.

— The Web service shall provide one grouping resource for every socket through external grouping sheets.

— The Web service shall provide textual label resources through external resource sheets, in at least one natural language.

— The Web service shall provide dynamic atomic resources at runtime for those socket elements where no (static) atomic resources are available in the target resources, as specified in 7.6.21.5, 7.6.22.5, and 7.6.23.6.

— The Web service, if representing a session-full target, shall support an open session request from a URC, as specified in 7.6.15.

— The Web service, if representing a session-full target, shall support a close session event from a URC, as specified in 7.6.16.

— The Web service, if representing a session-full target, shall support a suspend session event from a URC, as specified in 7.6.17.

— The Web service, if representing a session-full target, shall support a resume session event from a URC, as specified in 7.6.18.

— The Web service, if representing a session-full target, shall send an abort session event in case of user session abortion, as specified in 7.6.14.

— The Web service shall track connection status information from the underlying network its operations are bound to.

— The Web service, if representing a session-full target, shall send a session forward event to the URC in case of session forwarding, as specified in 7.6.14.

— The Web service, if representing a session-full target, shall create and maintain a session between a socket and the URC after a successful open session request.

— The Web service shall indicate to the URC the availability of socket elements at runtime (unavailable socket elements have an undefined value).

— The Web service shall synchronize the socket variables between the socket and the URCs that participate in a joint session with the socket (by means of the get-updates operation and the get operations of the variables).

— The Web service shall support command invocation requests from a URC (including handling of local parameters) and synchronization of command states (by means of the command operations).

— The Web service shall support propagation of notification states and, for custom-type notifications, embedded variables and commands, to the connected URCs, and acceptance of pertinent acknowledgments (by means of the get-updates operation and the check operations).

— The Web service shall synchronize actual indices of socket sets and elements (by means of the get-index operations).

— The Web service shall not rely on the URC doing the interpretation of socket element dependencies.

— Provide the following mechanisms with regard to user response timeouts:

  a) after a timeout extension, return to the state of the task the user had reached prior to the timeout;

  b) support the extend-timeout operation (see 7.6.23.4) for notifications that time out and let the client extend the timeout at least to five times the default timeout;

  c) note time out notifications in less than 10 s.

3 Normative references

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

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

ISO/IEC 24752-2:2013, *Information technology — User interfaces — Universal remote console — Part 2: User interface socket description*

ISO/IEC 24752-4:2013, *Information technology — User interfaces — Universal remote console — Part 4: Target description*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24752-1, ISO/IEC 24752-2 and ISO/IEC 24752-4 and the following apply:

4.1
fault item
named fault entity of a Web service operation, i.e. in WSDL1 a <message> referenced from a <fault> element of an <operation>, and in WSDL2 an <outfault> element of an <operation> referencing an interface fault

4.2
input item
named entity of input for a Web service operation, i.e. in WSDL1 a <part> of a <message> referenced from an <input> element of an <operation>; and in WSDL2 an <input> element of an <operation>

4.3
item element name
XML element name for an input or output item, i.e. in WSDL1 the value of the 'element' attribute on the pertaining <part> message; and in WSDL2 the value of the 'element' attribute on the <input> message

4.4
partition
Web service partition
named set of a Web service operations ("port type" in WSDL1, "interface" in WSDL2)

4.5
output item
named entity of output from a Web service operation, i.e. in WSDL1 a <part> of a <message> referenced from an <output> element of an <operation>, and in WSDL2 an <output> element of an <operation>

4.6
session-full socket
socket of a session-full target

4.7
session-less socket
socket of a session-less target

4.8
WSDL1 document
document that conforms to the Web Service Description Language (WSDL) 1.1 specification

4.9
WSDL2 document
document that conforms to the Web Service Description Language (WSDL) 2.0 specification

5 Relation to other standards

5.1 Relation to XML

This specification builds upon the extensible Markup Language (XML). Markup in XML is case sensitive.

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

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

Throughout this document, the following namespace prefixes and corresponding namespace identifiers are used for referencing namespaces. Authors are not bound to these prefixes, though their usage is recommended for better readability of public documents conforming to this International Standard.

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

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

— td: The target description namespace ("http://openurc.org/ns/targetdesc-2");

— uis: The user interface socket description namespace ("http://openurc.org/ns/uisocketdesc-2");

— wsdl-urc: The namespace for extending a WSDL1 or WSDL2 document by embedding an implicit target description and implicit socket descriptions ("http://openurc.org/ns/wsdl-urc");

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

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

6 Mapping Descriptions

6.1 General

A user interface socket (short "socket") consists of variables, commands, notifications, sets and type definitions. A Web service interface, as described by WSDL1, consists of port types, operations, messages, message parts, and type definitions. A Web service interface, as described by WSDL2, consists of interfaces, operations, messages, and type definitions.

A mapping description consists of the following parts: a mapping of a target and its properties to a Web service and its properties, a mapping of each of the target's sockets, its sets and its elements to one of the Web service's partition and its elements.

NOTE 1 A Web service partition is a functional unit of a Web service. In WSDL1, this is named a port type (element <portType>), in WSDL2 an interface (element <interface>).

NOTE 2 This section specifies general (semantic) requirements for mapping descriptions. The following section specifies a concrete syntax for mapping descriptions.

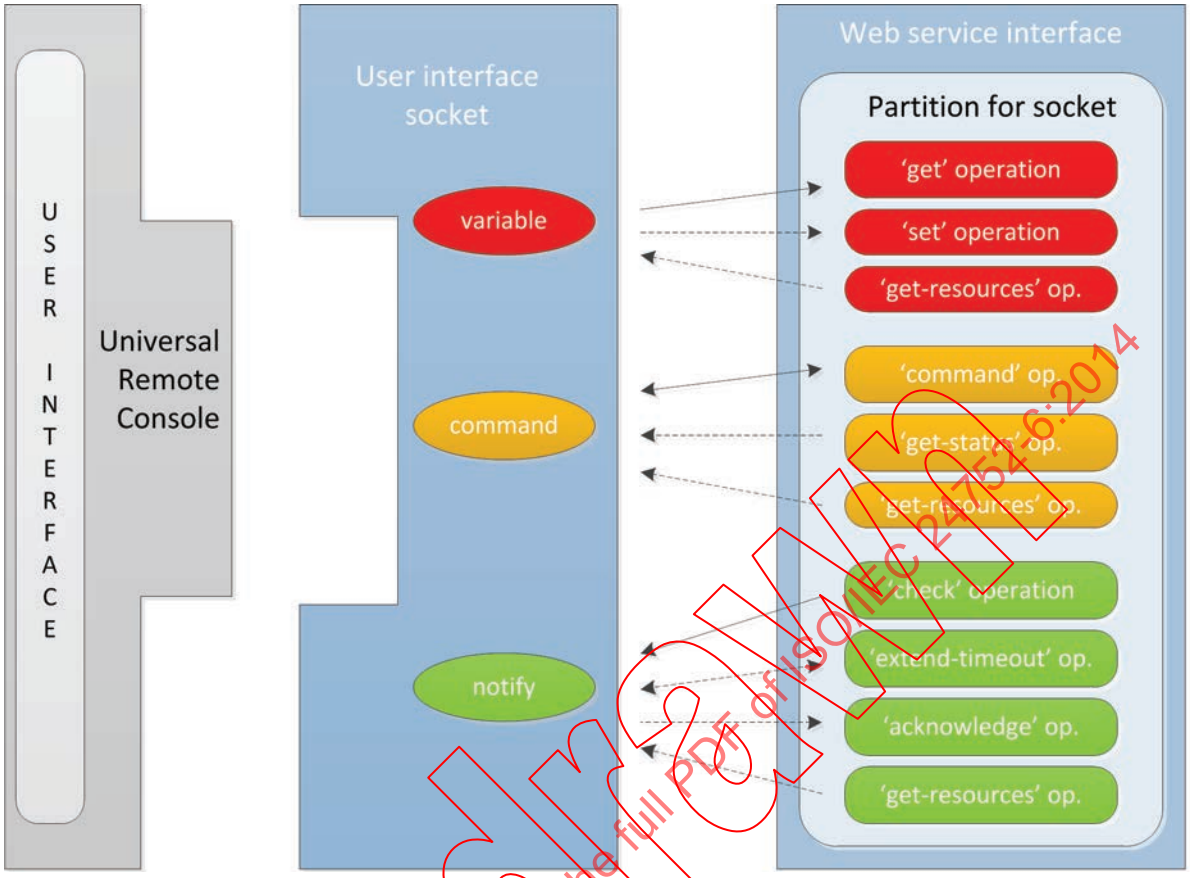

Figure 1 — Schematic mapping of a socket variable, a socket command, and a socket notify element to get, set, get-resources, command, get-status, check, extend-timeout and acknowledge operations provided by a partition of a Web service interface. The arrows indicate flow of socket content. Dashed arrows denote optional operations. Note that the Universal Remote Console (depicted on the left in gray) is included to provide contextual information for the socket, but is out of scope for this International Standard. (See ISO/IEC 24752-1 for more information on the Universal Remote Console.)

NOTE 3 In the following subsections, mappings are introduced for targets, sockets, variables, commands, notifications and type definitions. However, mappings for sets are implicitly contained in mappings for variables, commands and notifications (reflecting the structure of the socket).

6.2 Mapping a target to a Web service

The mapping description shall map exactly one target to exactly one Web service. The target shall be specified by its name (URI), and the Web service by its target namespace.

6.3 Mapping a socket to a Web service partition

A mapping description shall map the sockets of a mapped target to the partitions of a Web service that is mapped to the target (as specified in 6.2). Each socket included in the mapping shall be mapped to a single partition. The sockets shall be specified by their names (URIs), and the Web service partitions by their names.

6.4 Mapping a socket variable

6.4.1 General

A socket variable shall be mapped to the following Web service operations:

— a get operation,

— a set operation (only for writeable variables), and

— a get-resources operation (optional).

6.4.2 The get operation

A mapping description for a socket variable shall specify a get operation of a Web service with no input item and a single output item. At runtime, the get operation shall provide the current value of the socket variable through the output item. The get operation shall not change the state of the Web service. The types of the socket variable and the output item of the get operation shall be compatible. Time interval and expiration time for polling should be defined for each mapping mechanism.

6.4.3 The set operation

Unless the write dependency of the socket variable is always false (i.e. the variable is read-only), the mapping description shall also specify a set operation of a Web service with a single input item. The types of the socket variable and the input item of the set operation shall be compatible. At runtime, the set operation provides a way to modify the Web service's state. The input item shall reflect the requested new value for the variable. The output item shall carry the actual value of the variable after the operation has finished (i.e. the newly assigned value if the operation succeeded, or the old value if the operation failed).

6.4.4 The get-resources operation

The mapping description for a socket variable may specify a get-resources operation, if the Web service wants to provide dynamic atomic resources for the socket variable. If present, the get-resources operation shall have no input item and one output item with the element <wsdl-urc:resItems>, carrying a (possibly empty) set of dynamic atomic resource descriptions, each represented by an element <wsdl-urc:aResDesc> with the following subelements:

— either <wsdl-urc:content> (of type xs:string) or <wsdl-urc:contentAt> (of type xs:anyURI)

— <dc:type> (optional) - with any of the following string values: "Collection", "Dataset", "Event", "Image", "Interactive Resource", "Moving Image", "Physical Object", "Service", "Software", "Sound", "Still Image", "Text"

— <dc:format> (optional) - a valid MIME type

— <wsdl-urc:valRef> (optional) - of type xs:string

— <wsdl-urc:opRef> (optional) - with any of the following URIs: "http://openurc.org/ns/res#up", "http://openurc.org/ns/res#down"

— <wsdl-urc:role> - with any of the following URIs: "http://openurc.org/ns/res#label", "http://openurc.org/ns/res#help", "http://openurc.org/ns/res#accesskey", "http://openurc.org/ns/res#keyword", "http://openurc.org/ns/res#location"

— <wsdl-urc:forLang> (optional) - of type xs:language

These subelements correspond to the equally named properties of an atomic resource description in a resource sheet (see ISO/IEC 24752-5). However, as a restriction, the <wsdl-urc:content> element shall only contain text content, and shall not contain any subelements (such as <title>, or <value>).

EXAMPLE The socket variable "deviceStatus" is connected to the following dynamic atomic resources: (1) A text label for the current status value, and (2) an icon for the current status value. The current internal value of the "status" variable is "2" (which means standby mode, but this is not known before runtime). The output item of the get-resources operation would then assume the following XML fragment, provided that the default namespace is WSDL-URC ("http://openurc.org/ns/wsdl-urc"), and the namespace prefix "dc" is mapped to "http://purl.org/dc/elements/1.1/":

```
<resItems>
  <aResDesc>
    <content>standby</content>
    <dc:type>Text</dc:type>
    <dc:format>text</dc:format>
    <valRef>2</valRef>
    <role>http://openurc.org/ns/res#label</role>
    <forLang>en</forLang>
  </aResDesc>
  <aResDesc>
    <contentAt>http://example.com/deviceXY/standby.png</contentAt>
    <dc:type>Image</dc:type>
    <dc:format>image/png</dc:format>
    <valRef>2</valRef>
    <role>http://openurc.org/ns/res#label</role>
  </aResDesc>
</resItems>
```

NOTE By calling the get-resources operation, a client can retrieve a new dynamic resource (label, help text, access key, or location) for the socket variable or for any of its values. Note that this simple mechanism is restricted in that it does not allow for retrieving dynamic resources for types and type values rather than for socket elements and their values. Also, it does not allow for structuring textual labels, or integrating current values of socket elements.

6.5 Mapping a socket command

6.5.1 General

A socket command shall be mapped to the following Web service operations:

— a command operation,

— a get-status operation (only for commands of type other than 'uis:voidCommand'), and

— a get-resources operation (optional).

6.5.2 The command operation

A mapping description for a command shall specify a single command operation of a Web service. At runtime, the command operation is executed when the command is activated.

6.5.3 The get-status operation

A mapping description for a command may specify a single get-status operation of a Web service, if the command is executed synchronously, i.e. if it has a command type other than uis:voidCommand. At runtime, the URC can call the get-status operation to receive information on the command's status. Possible status values are: "initial", "rejected", "inProgress", "done", "succeeded", and "failed" (see ISO/IEC 24752-2).

6.5.4 The get-resources operation

The mapping description for a command may specify a get-resources operation, if the Web service wants to provide dynamic atomic resources for the socket command. If present, the get-resources operation shall have the same input and output items as for the get-resources operation for a socket variable (see 6.4.4).

6.6 Mapping a socket notification

6.6.1 General

A socket notification shall be mapped to the following Web service operations:

— a check operation,

— an extend-timeout operation (optional),

— an acknowledge operation (only for notifications of type other than “show”), and

— a get-resources operation (optional).

6.6.2 The check operation

A mapping description for a socket notification shall specify a check operation. At runtime, the check operation is frequently called (“polled”) for capturing the status of the notification.

6.6.3 The extend-timeout operation

A mapping description for a socket notification shall specify an extend-timeout operation if the notification has a ‘timeout’ attribute (see ISO/IEC 24752-2). Otherwise, it may specify an extend-timeout operation. At runtime, the extend-timeout operation can be called by the client to request an extension of the notification timeout.

6.6.4 The acknowledge operation

A mapping description for a socket notification shall specify an acknowledge operation of a Web service, if the notification has a type other than “show”. Upon user acknowledgment, the acknowledge operation is called.

6.6.5 The get-resources operation

The mapping description for a socket notification may specify a get-resources operation, if the Web service wants to provide dynamic atomic resources for the socket notification. If present, the get-resources operation shall have the same input and output items as for the get-resources operation for a socket variable (see 6.4.4).

6.7 Mapping a socket-internal type definition to a Web service’s type definition

A socket-internal type definition shall be mapped to a type definition in a Web service’s interface description.

NOTE 1 A socket-internal type definition is expressed by a <simpleType> or <complexType> element in the <xs:schema> part of a socket description (see ISO/IEC 24752-2).

NOTE 2 In a WSDL (both WSDL1 and WSDL2) document, a type definition is expressed by a <simpleType> or <complexType> element, contained in an <xs:schema> element in the <types> part.

7 Embedding target description and socket descriptions in a WSDL document

7.1 General

This section specifies syntax and conventions for providing mapping descriptions (see 6) inside a WSDL document, hereby providing an implicit target description and implicit socket descriptions. This approach is applicable to WSDL1 and WSDL2 documents, and is based on the generic mapping rules (as outlined in 6), naming conventions and some additional markup inside the <documentation> element in a WSDL file.

NOTE 1 In some cases, the syntax differs between WSDL1 and WSDL2. In these cases both versions are specified.

NOTE 2 When using the embedded approach, the structure of a target and its sockets is publicly available with the WSDL document. This means that the availability of the socket description cannot be made dependent on a user's opening a session (with proper credentials).

7.2 Restriction on target namespaces of internal schema part

A WSDL document shall define types in the same namespace as the target namespace of the WSDL document, unless types are imported from an external namespace via the <xs:import> directive.

EXAMPLE The target namespace of the type definition part is the same as the target namespace of the overall WSDL document (WSDL1). WSDL2 format accordingly. Ellipses ("...") indicate omissions.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/" ...>
  …
  <types>
     <xs:schema targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/">
       ...
     </xs:schema>
  </types>
  ...
</definitions>
```

7.3 Restriction on identifiers within target and socket descriptions

The values of 'id' attributes on XML elements in a target description or socket description that is implicitly mapped to a WSDL1 or WSDL2 document shall be restricted as follows:

— They shall not begin with an underscore character ('_').

— They shall not contain a dot character ('.').

— They shall not contain a hyphen character ('-').

NOTE Element names starting with underscore are reserved for internal use (e.g. the timeToComplete field for a timed command). Dots are not allowed since they are used in 'name' attributes inside WSDL documents as delimiters between path components for socket elements. Hyphens are not allowed since they are used as delimiters between notifications and their contained variables and commands.

7.4 'name' attribute values

A WSDL document contains 'name' attributes on various elements to identify an implicit target description and an implicit user interface socket description, and their components (see the following sections).

The values of these 'name' attributes shall be unique within the WSDL document, across all types of elements.

NOTE This is stricter than WSDL that requires uniqueness of 'name' attribute values within the same type of elements only.

7.5 Implicit target description

7.5.1 General

A target shall expose an implicit target description, embedded into its WSDL document, as described in the following subsections. Refer to ISO/IEC 24752-4 for the semantics of the information items of a target description.

7.5.2 Target name

The target's globally unique identifier ('name' attribute, see ISO/IEC 24752-4) shall be specified as value of the 'targetNamespace' attribute of the WSDL root element (<definitions> in WSDL1, <description> in WSDL2). Its value shall be a URI.

EXAMPLE 1 In WSDL1, the target's name is specified through the 'targetNamespace' attribute of the <definitions> element. Ellipses ("...") indicate omissions in the code.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" ...>
```

EXAMPLE 2 In WSDL2, the target's name is specified through the 'targetNamespace' attribute of the <description> element. Ellipses ("...") indicate omissions in the code.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl" ...>
```

7.5.3 The <wsdl-urc:hidden> element

A <wsdl-urc:hidden> element may be present as subelement of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2). If present, it shall have a Boolean value (i.e. either "true" or "false"). Its default value shall be "false".

This corresponds to the 'hidden' attribute on the <td:target> element in target descriptions (see ISO/IEC 24752-4).

EXAMPLE In this example, the target is specified as hidden. Note that this markup works for WSDL1 and WSDL2.

```
  <documentation>
    <wsdl-urc:hidden>true</wsdl-urc:hidden>
  </documentation>
```

7.5.4 The <dcterms:conformsTo> element

One or more <dcterms:conformsTo> elements shall be present as subelements of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2). Each of them shall have a URI as element content, specifying an established standard to which the Web service conforms.

The value "http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013" indicates that the Web service conforms to this International Standard.

EXAMPLE In this example, the Web service description is specified as conforming to this International Standard. Note that this markup works for WSDL1 and WSDL2.

```
  <documentation>
   <dcterms:conformsTo>http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013</
dcterms:conformsTo>
  </documentation>
```

NOTE 1 Clients of Web services can parse the Web service description, and look for occurrences of <dcterms:conformsTo> to find out whether the Web service has an implicit target description.

NOTE 2 <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation.

7.5.5 The <dcterms:modified> element

A <dcterms:modified> element may be present as subelement of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2). If present, its content shall be of type xs:date or xs:dateTime. The <dcterms:modified> element indicates that the implicit target description has been modified from its original version, while still referencing the same target.

EXAMPLE In this example, the implicit target description is marked as the version from Oct. 21, 2010. This markup works for WSDL1 and WSDL2.

```
<documentation>
 <dcterms:modified>2010-10-21</dcterms:modified>
</documentation>
```

NOTE <dcterms:modified> conforms to the Dublin Core metadata term modified, http://purl.org/dc/terms/modified.

7.5.6 Other target properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a target, if appropriate (as specified in ISO 15836). Each of them may occur multiple times as subelement of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2).

In particular, the following DCMI terms may be applied to a target (exposed through a Web service):

— <dc:identifier> specifying the product code (or instance code) of the target

— <dc:creator> specifying the manufacturer of the target

— <dc:publisher> specifying the provider of the target

— <dc:contributor> specifying co-manufacturers of the target

The 'xsi:type' attribute should be used to identify the coding schema, if appropriate.

EXAMPLE A company-specific identifier is provided for the target, based on the coding scheme 'myComp:companyCode'.

```
<documentation>
 <dc:identifier xsi:type="myComp:companyCode">0123456</dc:identifier>
</documentation>
```

7.5.7 The <wsdl-urc:resSheet> element

7.5.7.1 General

Any number of <wsdl-urc:resSheet> elements may occur as subelements of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2), each giving a reference to a resource sheet provided by the target manufacturer, and available within a local network.

NOTE 1 Resource sheets are collections of atomic resources, as defined in ISO/IEC 24752-5.

A resource sheet referenced in this way may contain atomic resources for one or more sockets of the target, as well as for the target itself (e.g. target label).

EXAMPLE In this example, a resource sheet with text and icon labels for English context is referenced. This resource sheet contains text and icon labels for the target itself, for its locators, for its socket with name="socket", and for the grouping sheet from the example in 7.5.9.1. Note that this markup works for WSDL1 and WSDL2.

```
<documentation>
```



```
    <wsdl-urc:resSheet about="http://openurc.org/TPL/basic-thermostat-1.0/basic-
thermostat.en.rsheet">
      <dcterms:conformsTo>http://openurc.org/ns/res/isoiec24752-5-2013</
dcterms:conformsTo>
      <dc:title xml:lang="en">English resource sheet for Thermostat connected via
Wi-Fi</dc:title>
      <dc:publisher>OpenURC Alliance</dc:publisher>
      <wsdl-urc:retrieveFrom>basic-thermostat1.0.en.rsheet</wsdl-urc:retrieveFrom>
      <wsdl-urc:scents>
        <wsdl-urc:forDomain>http://openurc.org/TPL/basic-thermostat-1.0/basic-
thermostat/</wsdl-urc:forDomain>
        <wsdl-urc:forDomain>http://openurc.org/TPL/basic-
thermostat-1.0/basic-thermostat/_locators
        </wsdl-urc:forDomain>
        <wsdl-urc:forDomain>http://openurc.org/TPL/basic-
thermostat-1.0/basic-thermostat/socket
        </wsdl-urc:forDomain>
       <wsdl-urc:forDomain>http://openurc.org/TPL/basic-thermostat-1.0/basic-thermostat.gsheet
        </wsdl-urc:forDomain>
        <wsdl-urc:forLang>en</wsdl-urc:forLang>
        <wsdl-urc:role>http://openurc.org/ns/res#label</wsdl-urc:role>
        <dc:type>Text</dc:type>
        <dc:type>Image</dc:type>
      </wsdl-urc:scents>
    </wsdl-urc:resSheet>
    ...
  </documentation>
```

NOTE 2 The <wsdl-urc:resSheet> element offers a possibility for a target manufacturer to provide "default" resource sheets in a local network. Other resource sheets - which are possibly more suitable for specific use contexts - can be retrieved from a resource server (see 7.5.11) provided by the target manufacturer or by third parties.

7.5.7.2 The 'about' attribute

A <wsdl-urc:resSheet> element (see 7.5.8.1) may have an 'about' attribute of the 'wsdl-urc' namespace, specifying an unambiguous identifier of the resource sheet. This shall be a globally unique identifier in the form of a Uniform Resource Identifier (URI, as specified in IETF RFC 3986), with no fragment identifier appended. This URI may or may not be resolvable.

NOTE 1 The 'about' attribute corresponds to the 'rdf:about' attribute on the <resSheet> element in a resource sheet (see ISO/IEC 24752-5), although of a different namespace.

NOTE 2 The URI provided by the 'about' attribute may or may not be resolvable. In any case, use the URI provided by <retrieveFrom> (see 7.5.8.6) for retrieving the resource sheet.

NOTE 3 The identifier as a value of the 'about' attribute conforms to the Dublin Core metadata element identifier, http://purl.org/dc/elements/1.1/identifier.

7.5.7.3 The <dcterms:conformsTo> element

A <wsdl-urc:resSheet> element (see 7.5.8.1) shall have one or more <dcterms:conformsTo> subelements, each specifying a reference to an established standard to which the resource sheet and its contents (atomic resource descriptions) conform. The value of each <dcterms.conformsTo> element shall be a URI (as specified in IETF RFC 3986), and shall be provided as element content.

EXAMPLE The following code specifies that the resource sheet complies with ISO/IEC 24752-5, as of 2013:

```
    <dcterms:conformsTo>http://openurc.org/ns/res/isoiec24752-5-2013<dcterms:conformsTo/>
```

NOTE 1 The <dcterms.conformsTo> element corresponds to the <dcterms:conformsTo> element as subelement of <ResSheet> in a resource sheet (see ISO/IEC 24752-5).

NOTE 2 The value of the <dcterms:conformsTo> element can be used when testing for conformance of a resource sheet.

NOTE 3 <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation.

7.5.7.4 Other resource sheet properties from DCMI

A <wsdl-urc:resSheet> element (see 7.5.8.1) may have any number of elements and element refinements from Dublin Core Metadata Terms (see ISO 15836) as subelements, if appropriate, to describe the resource sheet. Each of them may occur multiple times.

In particular, the following Dublin Core Metadata terms may occur:

— <dc:creator>

— <dc:publisher>

— <dc:contributor>

— <dc:rights>

— <dc:title> (with optional 'xml:lang' attribute)

7.5.7.5 The <wsdl-urc:scents> element

A <wsdl-urc:resSheet> element (see 7.5.8.1) may have a <wsdl-urc:scents> subelement.

If present, the <wsdl-urc:scents> element may have any number of subelements, providing hints as to what the resource sheet contains. The presence of each of these scent elements indicates that the scent value applies to at least one atomic resource in the resource sheet. The same scent elements may hereby occur multiple times, but with different values.

Applicable scent elements are the subelements of the <wsdl-urc:scents> element under <ResSheet> in a resource sheet, with their pertinent values given as element content (see ISO/IEC 24752-5). However, subelements of the namespace "http://openurc.org/ns/res#" are hereby imported into the namespace "http://openurc.org/ns/wsdl-urc". Subelements of other namespaces keep their original namespace.

NOTE The import into the namespace "http://openurc.org/ns/wsdl-urc" is necessary for automatic syntax checking with XML Schema Definition. The namespace "http://openurc.org/ns/res#" is defined in terms of RDF Schema and not suitable for import into an ordinary XSD file.

Applicable scent elements include:

— <dc:type>

— <dc:format>

— <wsdl-urc:forDomain>

— <wsdl-urc:forLang>

— <dcterms:audience>

— <wsdl-urc:role>

Other resource sheet scents from DCMI

7.5.7.6 The <wsdl-urc:retrieveFrom> element

A <wsdl-urc:resSheet> element (see 7.5.8.1) shall have a <wsdl-urc:retrieveFrom> subelement.

The <wsdl-urc:retrieveFrom> element specifies a URI (as specified in IETF RFC 3986), given as element content, that can be used to retrieve a copy of the referenced resource sheet that is thus made available to an entity in a local network environment.

NOTE The referenced resource sheet could be stored in a local network, or on the Internet. However, if stored on the Internet, it must be publicly available through the specified URI.

The URI may be relative, in which case it is based on the URI of the containing WSDL document.

7.5.8 The <wsdl-urc:grpSheet> element

7.5.8.1 General

Any number of <wsdl-urc:grpSheet> elements may occur as subelements of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2), each giving a reference to a grouping sheet provided by the target manufacturer, and available within a local network.

NOTE 1 A grouping sheet is a collection of grouping resources, as defined in ISO/IEC 24752-5.

A grouping sheet referenced in this way may contain groupings for one or more sockets of the target.

EXAMPLE A reference to a grouping sheet that provides a grouping for the target's socket with name="socket". This grouping sheet is not language-specific. Ellipses ("...") indicate omissions.

```
  <documentation>
    <wsdl-urc:grpSheet about="http://openurc.org/TPL/basic-thermostat-1.0/basic-
thermostat.gsheet">
      <dcterms:conformsTo>http://openurc.org/ns/res/isoiec24752-5-2013</
dcterms:conformsTo>
      <dc:title xml:lang="en">Grouping sheet for Thermostat connected via Wi-Fi</dc:title>
      <dc:publisher>URC Consortium</dc:publisher>
      <wsdl-urc:retrieveFrom>basic-thermostat1_0.gsheet</wsdl-urc:retrieveFrom>
      <wsdl-urc:scents>
        <wsdl-urc:forDomain>http://openurc.org/TPL/basic-
thermostat-1.0/basic-thermostat/socket
        </wsdl-urc:forDomain>
        <wsdl-urc:forLang></wsdl-urc:forLang>
      </wsdl-urc:scents>
    </wsdl-urc:grpSheet>
    …
  </documentation>
```

NOTE 2 The <wsdl-urc:grpSheet> element offers a possibility for a target manufacturer to provide "default" grouping sheets in a local network. Other grouping sheets - which are possibly more suitable for specific use contexts - can be retrieved from a resource server (see 7.5.11) provided by the target manufacturer or by third parties.

7.5.8.2 The 'about' attribute

A <wsdl-urc:grpSheet> element (see 7.5.9.1) shall have an 'about' attribute of the 'wsdl-urc' namespace, specifying an unambiguous identifier of a grouping sheet. This shall be a globally unique identifier in the form of a Uniform Resource Identifier (URI), as specified in IETF RFC 3986, with no fragment identifier appended.

NOTE 1 The 'about' attribute corresponds to the 'rdf:about' attribute on the <GrpSheet> element in a grouping sheet (see ISO/IEC 24752-5), although of a different namespace.

NOTE 2 The URI provided by the 'about' attribute may or may not be resolvable. In any case, use the URI provided by <wsdl-urc:retrieveFrom> (see 7.5.9.6) for retrieving the grouping sheet.

NOTE 3 The identifier as a value of the 'about' attribute conforms to the Dublin Core metadata element identifier, http://purl.org/dc/elements/1.1/identifier.

7.5.8.3 The <dcterms:conformsTo> element

A <wsdl-urc:grpSheet> element (see 7.5.9.1) shall have one or more <dcterms:conformsTo> subelements, each specifying a reference to an established standard to which the grouping sheet and its contents (groupings) conform. The value of each <dcterms.conformsTo> element shall be a URI (as specified in IETF RFC 3986), and shall be provided as element content.

EXAMPLE The following code specifies that the grouping resource complies with ISO/IEC 24752-5, as of 2013:

```
<dcterms:conformsTo>http://openurc.org/ns/res/isoiec24752-5-2013</dcterms:conformsTo>
```

NOTE 1 The <dcterms.conformsTo> element corresponds to the <dcterms:conformsTo> element as subelement of <GrpSheet> in a grouping sheet (see ISO/IEC 24752-5).

NOTE 2 The value of the <dcterms:conformsTo> element can be used when testing for conformance of a grouping sheet.

NOTE 3 <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation.

7.5.8.4 Other grouping sheet properties from DCMI

A <wsdl-urc:grpSheet> element (see 7.5.9.1) may have any number of elements and element refinements from Dublin Core Metadata Terms (see ISO 15836) as subelements, if appropriate, to describe the resource sheet. Each of them may occur multiple times.

In particular, the following Dublin Core Metadata terms may occur:

— <dc:creator>

— <dc:publisher>

— <dc:contributor>

— <dc:rights>

— <dc:title> (with optional 'xml:lang' attribute)

7.5.8.5 The <wsdl-urc:scents> element

A <wsdl-urc:grpSheet> element (see 7.5.9.1) may have a <wsdl-urc:scents> subelement.

If present, the <wsdl-urc:scents> element may have any number of subelements, providing hints as to what the grouping sheet contains. The presence of each of these scent elements indicates that the scent value applies to at least one grouping in the grouping sheet. The same scent elements may hereby occur multiple times, but with different values.

Applicable scent elements are the subelements of the <wsdl-urc:scents> element under <GrpSheet> in a grouping sheet, with their pertinent values given as element content (see ISO/IEC 24752-5). However, subelements of the namespace "http://openurc.org/ns/res#" are hereby imported into the namespace "http://openurc.org/ns/wsdl-urc". Subelements of other namespaces keep their original namespace.

NOTE The import into the namespace "http://openurc.org/ns/wsdl-urc" is necessary for automatic syntax checking with XML Schema Definition. The namespace "http://openurc.org/ns/res#" is defined in terms of RDF Schema and not suitable for import into an ordinary XSD file.

Applicable scent elements include:

— <wsdl-urc:forDomain>

— <wsdl-urc:forLang>

— Other grouping sheet scents from DCMI

7.5.8.6 The <wsdl-urc:retrieveFrom> element

A <wsdl-urc:grpSheet> element shall have a <wsdl-urc:retrieveFrom> subelement.

The <wsdl-urc:retrieveFrom> element specifies a URI (as specified in IETF RFC 3986), given as element content, that can be used to retrieve a copy of the referenced grouping sheet that is thus made available to an entity in a local network environment.

NOTE The referenced grouping sheet could be stored in a local network, or on the Internet. However, if stored on the Internet, it must be publicly available through the specified URI.

The URI may be relative, in which case it is based on the URI of the containing WSDL document.

7.5.9 The <wsdl-urc:uiid> element

7.5.9.1 General

Any number of <wsdl-urc:uiid> elements may occur as subelements of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2), each giving a reference to a user interface implementation description (UIID) provided by the target manufacturer, and available in a local network.

NOTE 1 UIIDs are instances of a broad range of file formats, some of which may be proprietary.

A UIID referenced in this way may apply to one or more sockets of the target.

EXAMPLE: This code fragment references a UIID of MIME type "text/html".

```
<wsdl-urc:uiid about="http://example.com/thermometer/uiid.html">
    <dc:format>text/html</dc:format>
    <wsdl-urc:retrieveFrom>uiid.html</wsdl-urc:retrieveFrom>
</wsdl-urc:uiid>
```

NOTE 2 The <wsdl-urc:uiid> element offers a possibility for a target manufacturer to provide "default" UIIDs. Other UIIDs - which are possibly more suitable for specific use contexts - could be made available on a resource server (see 7.5.11), provided by the target manufacturer or by third parties.

7.5.9.2 The 'about' attribute

A <wsdl-urc:uiid> element (see 7.5.10.1) shall have an 'about' attribute of the 'wsdl-urc' namespace, specifying an unambiguous identifier of the referenced UIID. This shall be a globally unique identifier in the form of a Uniform Resource Identifier (URI), as specified in IETF RFC 3986, with no fragment identifier appended.

NOTE 1 The 'about' attribute corresponds to a globally unique identifier, as used by the UIID.

NOTE 2 The URI provided by the 'about' attribute may or may not be resolvable. In any case, use the URI provided by <wsdl-urc:retrieveFrom> (see 7.5.10.6) for retrieving the UIID.

NOTE 3 The identifier as a value of the attribute 'about' conforms to the Dublin Core metadata element identifier, http://purl.org/dc/elements/1.1/identifier.

7.5.9.3 The <dcterms:conformsTo> element

A <wsdl-urc:uiid> element (see 7.5.10.1) may have one or more <dcterms:conformsTo> subelements, each specifying a reference to an established standard to which the UIID conforms. The value of each <dcterms.conformsTo> element shall be a URI (as specified in IETF RFC 3986), and shall be provided as element content.

NOTE 1 The value of the <dcterms:conformsTo> element can be used when testing for conformance of a UIID.

NOTE 2 <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation.

7.5.9.4 The <wsdl-urc:forLang> element

A <wsdl-urc:uiid> element (see 7.5.10.1) may have any number of <wsdl-urc:forLang> subelements.

The <wsdl-urc:forLang> element specifies (as element content) which language context the UIID can be applied to. Language contexts shall be 3-letter codes as for the <xml:lang> element of XML 1.0. An empty <wsdl-urc:forLang> element indicates that the UIID is not language-specific.

EXAMPLE <wsdl-urc:forLang>en</wsdl-urc:forLang>

NOTE Typically UIIDs will be defined in a language independent way. For these UIIDs, an empty <wsdl-urc:forLang> element is recommended.

7.5.9.5 Other UIID properties from DCMI

A <wsdl-urc:uiid> element (see 7.5.10.1) may have any number of elements and element refinements from Dublin Core Metadata Terms (see ISO 15836) as subelements, if appropriate. Each of them may occur multiple times.

In particular, the following Dublin Core Metadata terms may occur:

— <dc:creator>

— <dc:publisher>

— <dc:contributor>

— <dc:rights>

— <dc:title> (with optional 'xml:lang' attribute)

— <dc:format>

— <dcterms:audience>

7.5.9.6 The <wsdl-urc:retrieveFrom> element

A <wsdl-urc:uiid> element (see 7.5.10.1) shall have a <wsdl-urc:retrieveFrom> subelement.

The <wsdl-urc:retrieveFrom> element specifies a URI (as specified in IETF RFC 3986), given as element content, that can be used to retrieve a copy of the referenced UIID in a local network environment.

The URI may be relative, in which case it is based on the URI of the containing WSDL document.

7.5.10 The <wsdl-urc:resSvc> element

7.5.10.1 General

Any number of <wsdl-urc:resSvc> elements may occur as subelements of the <documentation> element underneath the root element (<definitions> in WSDL1, <description> in WSDL2), each giving a reference to a resource service that can be queried for any types of resources, including:

— Atomic resources such as labels, help texts, keywords, and access keys (as defined by ISO/IEC 24752-5),

— grouping resources (as defined by ISO/IEC 24752-5), and

— UIIDs (format not specified by this International Standard).

A resource service may provide resources from target manufacturers and any third parties, beyond the (default) resources that are provided by a target in its local network environment.

EXAMPLE The following is an example for a resource service description. The resource service conforms to the OpenURC's Resource Server HTTP Interface 1.0 specification of 2009-04-29 (as specified at http://openurc.org/TR/res-serv-http1.0-20090429/), and its interface description is available at http://res.openurc.org.

```
  <wsdl-urc:resSvc about="http://res.openurc.org">
    <dcterms:conformsTo>http://openurc.org/TR/res-serv-http1.0-20090429/</
dcterms:conformsTo>
    <dc:publisher>OpenURC Alliance</dc:publisher>
    <dc:title xml:lang="en">Resource Server of the OpenURC Alliance</dc:title>
  </wsdl-urc:resSvc>
```

7.5.10.2 The 'about' attribute

A <wsdl-urc:resSvc> element (see 7.5.11.1) shall have an 'about' attribute of the 'wsdl-urc' namespace, specifying an unambiguous identifier of a resource service. This shall be a globally unique identifier in the form of a Uniform Resource Identifier (URI), as specified in IETF RFC 3986.

This URI shall be globally resolvable and shall deliver a description file for the resource service.

The format of an external description file for a resource service is beyond the scope of this International Standard. If existing, interface description formats as defined by other standards, may be employed. If standardized, the contained <dcterms:conformsTo> element and/or the MIME type and/or the file extension of the resource service description file may induce its format.

NOTE The identifier as a value of the 'about' attribute conforms to the Dublin Core metadata element identifier, http://purl.org/dc/elements/1.1/identifier.

7.5.10.3 The <dcterms:conformsTo> element

A <wsdl-urc:resSvc> element (see 7.5.11.1) may have a <dcterms:conformsTo> subelement, specifying a reference (as URI, as specified in IETF RFC 3986) to an established standard to which the resource service conforms. The URI shall be specified as element content.

NOTE <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation.

7.5.10.4 Other Resource Server properties from DCMI

A <wsdl-urc:resSvc> element (see 7.5.11.1) may have any number of elements and element refinements from Dublin Core Metadata Terms (see ISO 15836) as subelements, if appropriate, to describe the resource server and its contents. Each of them may occur multiple times.

In particular, the following Dublin Core Metadata terms may occur:

— <dc:publisher>

— <dc:rights>

— <dc:title> (with optional 'xml:lang' attribute)

— <dcterms:audience>

7.5.11 The "_target" partition

7.5.11.1 General

A Web service shall provide a dedicated "_target" partition for the provision of the target instance identifier, and for its locators, if any.

All locators of a target shall occur as operations in the "_target" partition of the Web service.

In WSDL1, the "_target" partition shall be present as <portType> element as subelement of the <definitions> root element. It shall have a 'name' attribute with the value "_target".

EXAMPLE 1 A _target partition in WSDL1, with the mandatory operation "getTargetInstanceId" and a locator operation "beep". Ellipses ("...") indicate omissions.

```
<definitions targetNamespace=http://openurc.org/TPL/basic-thermostat-1.0/
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>
  ,,,
  <message name="empty" />
  <message name="targetInstanceId">
    <part name="targetInstanceId" element="wsdl-urc:targetInstanceId"/>
  </message>

  <portType name="_target">

    <operation name="getTargetInstanceId">
      <dc:description xml:lang="en">Get globally unique identifier for the target
instance.</dc:description>
      <input message="empty"/>
      <output message="targetInstanceId"/>
    </operation>

    <operation name="beep">
      <documentation>
        <wsdl-urc:locatorType>audio</wsdl-urc:locatorType>
      </documentation>
      <dc:description xml:lang="en">A locator operation causing the device to beep.</
dc:description>
      <input message="empty"/>
    </operation>

  </portType>
  ...
</definitions>
```

In WSDL2, the "_target" partition shall be present as <interface> element as subelement of the <description> root element. It shall have a 'name' attribute with the value "_target".

EXAMPLE 2 A _target partition in WSDL2, with the mandatory operation "getTargetInstanceId" and a locator operation "beep". Ellipses ("...") indicate omissions.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>
  ...
  <interface name="_target">
     <!-- The interface '_target' contains operations for the target instance identifier
and locator functions. -->

    <operation name="getTargetInstanceId"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
      wsdlx:safe="true">
      <dc:description xml:lang="en">Get globally unique identifier for the target
instance.</dc:description>
      <input messageLabel="In" element="#none" />
      <output messageLabel="Out" element="wsdl-urc:targetInstanceId" />
    </operation>

    <operation name="beep"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
      wsdlx:safe="true">
      <documentation>
        <wsdl-urc:locatorType>audio</wsdl-urc:locatorType>
```

```
        </documentation>
        <dc:description xml:lang="en">A locator operation causing the device to beep.</
dc:description>
        <input messageLabel="In" element="#none" />
        <output messageLabel="Out" element="#none" />
      </operation>

   </interface>
   ...
</description>
```

7.5.11.2 'getTargetInstanceId' operation

The 'getTargetInstanceId' operation shall have the following input and output items:

— No input item.

— Exactly one output item of element <wsdl-urc:targetInstanceId>, reflecting the target's instance identifier (see ISO/IEC 24752-1, section "Target instance identifier").

NOTE See 7.5.11.1 for WSDL1 and WSDL2 example code for the 'getTargetInstanceId' operation.

7.5.11.3 Locator functions

7.5.11.3.1 General

Locator functions shall occur as operations (with any names) within the "_target" partition.

A locator operation shall be invokable by the user without prior registration or login.

A locator operation shall have no input and no output items.

NOTE In WSDL1, this means that the locator operation shall have an empty <input> message, and no <output> element. In WSDL2, this means that the locator operation shall have <input> and <output> messages, each with element="#none".

7.5.11.3.2 Locator type

A locator operation shall have a <wsdl-urc:locatorType> element as subelement of the <documentation> element. Its content shall be any of the following values:

— "audio": Audible locator, i.e. the target emits an audible signal (such as a beep or bell) when invoked by the user.

— "visual": Visual locator, i.e. the target emits a visual signal (such as a flash) when invoked by the user.

— "other": Other means for localizing a target, e.g. IR pulse.

7.6 Implicit socket description

7.6.1 General

A target shall expose one or multiple implicit socket descriptions, embedded into its WSDL document, as described in the following subsections. Refer to ISO/IEC 24752-2 for the semantics of the information items of a socket description.

An implicit socket description shall be specified as Web service partition. In the following sections, such a Web service partition is also referred to as "socket".

In a WSDL1 document, a socket shall be described by a <portType> element (underneath the <definitions> root element), its 'name' attribute, and its subelements (<operation>).

In a WSDL2 document, a socket shall be described by an <interface> element (underneath the <description> root element), its 'name' attribute, and its subelements (<operation>).

EXAMPLE 1 Implicit socket description in WSDL1.

```
  <portType name="socketname">
    <documentation>
      <dcterms:conformsTo>http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013</
dcterms:conformsTo>
      <!-- Other socket properties go here -->
    </documentation>
    <!-- Operations of the socket go here -->
  </portType>
```

EXAMPLE 2 Implicit socket description in WSDL2.

```
  <interface name="socketname">
    <documentation>
      <dcterms:conformsTo>http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013</
dcterms:conformsTo>
      <!-- Other socket properties go here -->
    </documentation>
    <!-- Operations of the socket go here -->
  </interface>
```

The Web service partition representing a socket shall include no other operations than those specified in this section.

7.6.2 Session-full vs. session-less sockets

A socket that is represented by a Web service partition can be session-full or session-less.

If the Web service partition represents a session-full socket:

— it shall define a session fault (see 7.6.14);

— it shall contain the following operations: open-session-request (see 7.6.15), close-session-request (see 7.6.16), suspend-session-request (see 7.6.17), resume-session-request (see 7.6.18), get-session-status (see 7.6.19);

— all of its operations, except for the open-session-request operation, shall have "sessionId" as the first input item; and

— all of its operations, except for the 'open-session-request' and the 'get-session-status' operations, shall reference the session fault.

If the Web service partition represents a session-less socket:

— it shall not define a session fault (see 7.6.14);

— it shall not contain the following operations: open-session-request (see 7.6.15), close-session-request (see 7.6.16), suspend-session-request (see 7.6.17), resume-session-request (see 7.6.18), get-session-status (see 7.6.19);

— none of its operations shall have "sessionId" as an input item; and

— none of its operations shall reference a session fault.

In the following subsections, a Web service partition representing a session-full socket is referred to as "session-full socket", and a Web service partition representing a session-less socket as "session-less socket".

NOTE 1 Most of the examples in this International Standard assume a session-less socket (e.g. a socket for a thermostat) for simplicity.

NOTE 2 Web service clients can use the presence of the session-related operations open-session-request, close-session-request, and get-session-status as an indicator for a session-full socket. Note that the operations suspend-session-request and resume-session-request are optional for session-full sockets.

7.6.3 The 'name' attribute

The 'name' attribute of a Web service partition is used to construct the implicit socket's name (URL), as follows (in Extended BNF notation):

implicit socket name = *implicit target name* , *name attribute value* ;

where *implicit target name* is the globally unique identifier of the target (see 7.5.2), and *name attribute value* is the value of the 'name' attribute on the Web service partition (<portType> in WSDL1, and <interface> in WSDL2).

EXAMPLE A WSDL1 document defines "http://openurc.org/res/devices/basic-thermostat/" as implicit target name, and "socket1" as the value of the 'name' attribute of the <portType> element, as follows (omissions indicated as elipses):

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  ...>
  ...
  <portType name="socket1">
  ...
  </portType>
</definitions
```

Therefore the implicit socket name is: http://openurc.org/TPL/basic-thermostat-1.0/socket1.

7.6.4 The <dcterms:conformsTo> element

A <dcterms:conformsTo> element shall be present as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). It shall have a URI as element content, specifying an established standard to which the implicit socket description conforms. The value http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013 indicates that the socket description conforms to this International Standard.

NOTE A conforming Web service description can contain both conforming and non-conforming partitions. Clients should always check for conformance of partitions when parsing a WSDL1 or WSDL2 document.

7.6.5 The <wsdl-urc:socketType> element

A socket may have a <wsdl-urc:socketType> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, its value shall be one of the following:

— "location-dependent". Location-dependent sockets have a defined location, and require the user to be close to the target and its socket.

— "location-informative". Location-informative sockets have a defined location, but can be controlled from anywhere.

— "location-free". Location-free sockets have no meaningful location, i.e. they exist in a virtual space.

Sockets whose type is not specified, shall be "location-dependent" (default value).

EXAMPLE 1 An ATM service would be location-dependent, a home security system would be location-informative, and an Internet-based currency rate information service would location-free.

EXAMPLE 2 A location-dependent ATM service (WSDL1 or WSDL2).

```
<documentation>
```



```
      <dcterms:conformsTo>http://openurc.org/ns/wsdl-urc/isoiec24752-6-2013</
dcterms:conformsTo>
      <wsdl-urc:socketType>location-dependent</wsdl-urc:socketType>
    </documentation>
```

7.6.6 The <wsdl-urc:hidden> element

A socket may have a <wsdl-urc:hidden> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have a Boolean value (i.e. either “true” or “false”). Its default value shall be “false”.

This corresponds to the ‘hidden’ attribute on a <td:socket> element in target descriptions (see ISO/IEC 24752-4, section 6.9.5).

EXAMPLE In this example, the socket is specified as hidden (WSDL1 or WSDL2).

```
    <documentation>
      <wsdl-urc:hidden>true</wsdl-urc:hidden>
    </documentation>
```

7.6.7 The <wsdl-urc:maxSessions> element

A session-full socket may have a <wsdl-urc:maxSessions> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have an integer equal to or greater than “1” as content.

This corresponds to the ‘maxSessions’ attribute on a <td:socket> element in target descriptions (see ISO/IEC 24752-4, section 6.9.6).

There is no default value for the <wsdl-urc:maxSessions> element. If it is not present, no information about the maximum number of sessions is available.

EXAMPLE In this example, the socket can only maintain one session at a time (WSDL1 or WSDL2).

```
    <documentation>
      <wsdl-urc:maxSessions>1</wsdl-urc:maxSessions>
    </documentation>
```

7.6.8 The <wsdl-urc:sharedSessions> element

A session-full socket may have a <wsdl-urc:sharedSessions> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have a Boolean value as content, i.e. “true” or “false”.

This corresponds to the ‘sharedSessions’ attribute on a <td:socket> element in target descriptions (see ISO/IEC 24752-4, section 6.9.7).

There is no default value for the <wsdl-urc:sharedSessions> element. If it is not present, no information about the sharing of sessions is available.

EXAMPLE In this example, the socket shares parallel sessions (WSDL1 or WSDL2).

```
    <documentation>
      <wsdl-urc:sharedSessions>true</wsdl-urc:sharedSessions>
    </documentation>
```

7.6.9 The <wsdl-urc:requestable> element

A socket may have a <wsdl-urc:requestable> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “true”.

This corresponds to the ‘requestable’ attribute on a <td:socket> element in target descriptions (see ISO/IEC 24752-4, 6.9.8).

EXAMPLE In this example, the socket can be requested to be opened directly by a URC (WSDL1 or WSDL2).

```
<documentation>
  <wsdl-urc:requestable>true</wsdl-urc:requestable>
</documentation>
```

7.6.10 The <wsdl-urc:sufficient> element

A socket may have a <wsdl-urc:sufficient> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have a Boolean value as content, i.e. “true” or “false”.

This corresponds to the ‘sufficient’ attribute on a <uis:uiSocket> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A socket marked as sufficient (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:sufficient>true</wsdl-urc:sufficient>
</documentation>
```

7.6.11 The <wsdl-urc:complete> element

A socket may have a <wsdl-urc:complete> element as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2). If present, it shall have a Boolean value as content, i.e. “true” or “false”.

This corresponds to the ‘complete’ attribute on a <uis:uiSocket> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A socket marked as complete (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:complete>true</wsdl-urc:complete>
</documentation>
```

7.6.12 Socket properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a socket, if appropriate (as specified in ISO 15836). Each of them may occur any number of times as subelement of the <documentation> element of the partition (<portType> in WSDL1, <interface> in WSDL2).

In particular, the following DCMI terms may be applied to a socket (exposed through a Web service’s partition):

— <dc:identifier> specifying the product code (or instance code) of the socket

— <dc:creator> specifying the manufacturer of the socket

— <dc:publisher> specifying the provider of the socket

— <dc:contributor> specifying co-manufacturers of the socket

The ‘xsi:type’ attribute should be used to identify the coding schema, if appropriate.

EXAMPLE In this example, the author of the socket is specified as “Gottfried Zimmermann” (WSDL1 and WSDL2).

```
<documentation>
  <dc:creator>Gottfried Zimmermann</dc:creator>
</documentation>
```

7.6.13 Set path

A user interface socket contains elements (variables, commands, and notifications) that can be nested within sets in a hierarchical structure. Each element has a “set path”, that is a list of set identifiers reflecting the way from the root to the element.

In the following subsections, the set path of an element shall consist of a dot-separated list of identifiers of all sets that form the path from the root to the element, with a trailing dot.

This can be expressed in Extended BNF notation (see ISO/IEC 14977) as follows:

set path = { set identifier, "." } ;

where *set identifier* is the value of the 'id' attribute of <set>.

EXAMPLE 1 A variable with id="power" is contained in a set "powerControls" under <uiSocket>, has the following set path: powerControls.

EXAMPLE 2 A variable with id="power" is contained in a set "powerControls" which is contained in a set "tvControls" under <uiSocket>, has the following set path: tvControls.powerControls.

An element that is directly underneath the socket (i.e. that is not contained in any set) shall have an empty path of sets (pathOfSets is an empty string).

7.6.14 The session fault

A session-full socket shall define a session fault, to be used by its operations as indicated for each operation.

The session fault shall have either one of the following values, each indicating a specific session-related error condition:

— "invalid": invalid session identifier;

— "expired": session has expired (client has closed it);

— "aborted": target has aborted the session;

— "forwarded": target has closed the session and wants the client to open a session with another socket. The client should call the operation get-session-status in order to learn about the forward socket.

In WSDL1, the session cannot be defined on <portType> level; it nevertheless shall occur on the pertinent operations as <fault> element with name="sessionFault".

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document, including the <fault> elements on operations, and the pertaining element, type and message definitions.

In WSDL2, the session management interface shall define an interface-level <fault> with name="sessionFault", to be referenced by an <outfault> element on the pertinent operations.

EXAMPLE In WSDL2, a session fault is defined within an <interface> element as follows:

```
    <fault name="sessionFault" element="wsdl-urc:sessionStatus">
      <!-- This fault is raised by the target if the operation cannot be concluded due to
a problem with
        the session.  The reason is provided as element content of <wsdl-
urc:sessionStatus>.
        Allowed values are:
          "invalid": invalid session identifier.
          "expired": session has expired (client has closed it).
          "aborted": target has aborted the session.
        "forwarded": target has closed the session and wants the client to open a session with
           another socket. The client should call the operation getSessionStatus in order to
            learn about the forward socket. -->
    </fault>
```

NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document, including the <fault> on <interface>, the <outfault> elements on operations, and the pertaining element and type definitions.

7.6.15 The open-session-request operation

An open-session-request operation shall be present in a session-full socket. Its 'name' attribute shall have the value "openSessionRequest". It shall have the following input and output items:

— An input item named "authorizationCode" of type xs:string, specifying an authorization code required for opening a session. authorizationCode may be empty.

— An output item named "sessionId" of type xs:NMTOKEN, specifying the session identifier, if the operation was successful. If the open session requested was unsuccessful, "sessionId" shall be empty.

— An output item named "forwardSocketName" of type xs:anyURI. forwardSocketName shall be empty if "sessionId" is non-empty; otherwise it may provide a forward target (WSDL file location) and forward socket (partition) in the following form: "[URL of target WSDL file]#wsdl.interface([partition name])", whereas

  — "[URL of target WSDL file]" being the URL of a WSDL file representing a web service target conforming to this International Standard;

  — "[partition name]" being the name of the web service partition representing the socket that the client is forwarded to;

  NOTE Although WSDL1 uses <portType> rather than <interface> for representing a socket, the syntax "wsdl.interface([partition name])" is nevertheless used to reference a port type.

— An output item named "forwardSocketAuthorizationCode" of type xs:string. forwardSocketAuthorizationCode may have a non-empty value only if the output item forwardSocketName is non-empty; otherwise it shall be empty. If provided, it shall specify an authorization code required to open a session with the socket specified by forwardSocketName.

EXAMPLE 1 In WSDL1, the open-session-request operation is specified as follows:

```
    <operation name="openSessionRequest">
      <dc:description xml:lang="en">For a session-full socket, a URC calls this
operation to open
      a session with the socket.</dc:description>
      <input message="authorizationCode" />
         <!-- <wsdl-urc:authorizationCode> carries an optional authorization code,
may be empty -->
      <output message="openSessionOutputMessage" />
         <!-- <wsdl-urc:sessionId> contains the identifier for the new session, if granted.
           If rejected or forwarded, <wsdl-urc:sessionId> is empty. -->
        <!-- <forwardSocketName> contains a reference to the forwarded socket, otherwise empty.
           The socket name for forward is provided in the following form:
           [URL of target WSDL file]#wsdl.interface([partition name]) -->
         <!-- <wsdl-urc:forwardSocketAuthorizationCode> contains an authorization code
for the socket,
           that the URC is forwarded to.  Empty if no forwarding or no authorization
required. -->
    </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its open-session-request operation, including the pertaining element, type and message definitions.

EXAMPLE 2 In WSDL2, the open-session-request operation is specified as follows:

```
    <operation name="openSessionRequest"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <dc:description xml:lang="en">For a session-full socket, a URC calls this
operation to open
        a session with the socket.</dc:description>
      <input messageLabel="In" element="wsdl-urc:authorizationCode" />
         <!-- Optional authorization code, may be empty -->
      <output messageLabel="Out" element="wsdl-urc:sessionId" />
```



```
        <!-- If request granted, <wsdl-urc:sessionId> contains the identifier for
the new session.
      If rejected or forwarded, <wsdl-urc:sessionId> is empty. -->
      <output messageLabel="Out" element="wsdl-urc:forwardSocketName" />
        <!-- If forwarded, <forwardSocketName> contains a reference to the forwarded
socket, in the following form:
        [URL of target WSDL file]#wsdl.interface([partition name]). -->
      <output messageLabel="Out" element="wsdl-urc:forwardSocketAuthorizationCode" />
        <!-- If forwarded, <wsdl-urc:forwardSocketAuthorizationCode> contains an
authorization code for the socket
        that the URC is forwarded to.  May be empty if no authorization required. -->
    </operation>
```

NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its open-session-request operation, including the pertaining element and type definitions.

7.6.16 The close-session-request operation

A close-session-request operation shall be present in a session-full socket. Its 'name' attribute shall have the value "closeSessionRequest". It shall have the following input and output items:

— An input item named "sessionId" of type xs:NMTOKEN, specifying the session that is requested to be closed.

— An output item named "requestResult" of type xs:boolean. It shall be "true" if the session has been close as a result of the request, otherwise "false".

— A session fault (see 7.6.14).

EXAMPLE 1 In WSDL1, the close-session-request operation is specified as follows:

```
    <operation name="closeSessionRequest">
      <dc:description xml:lang="en">For a session-full socket, a URC calls this
operation to close a
        session with the socket.</dc:description>
      <input message="sessionId" />
      <!-- <wsdl-urc:sessionId> carries the identifier of the active session. -->
      <output message="requestResult" />
      <!-- true if closed, false if session expired or otherwise invalid sessionId. -->
      <fault name="sessionFault" message="sessionStatus"/>
        <!-- This fault is raised by the target if the operation cannot be concluded due
to a problem with
          the session.  The reason is provided as element content of <sessionStatus>.
          Allowed values are:
          "invalid": invalid session identifier.
          "expired": session has expired (client has closed it).
          "aborted": target has aborted the session.
          "forwarded": target has closed the session and wants the client to open
a session with
            another socket.  The client should call the operation getSessionStatus in order to
              learn about the forward socket.
        This applies also to all following faults with name="sessionFault". -->
    </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its close session request operation, including the pertaining element, type and message definitions.

EXAMPLE 2 In WSDL2, the close-session-request operation is specified as follows:

```
    <operation name="closeSessionRequest"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <dc:description xml:lang="en">For a session-full socket, a URC calls this
operation to close a
        session with the socket.</dc:description>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
```

```
        <!-- Identifier of the active session. -->
      <output messageLabel="Out" element="wsdl-urc:requestResult" />
          <!-- true if closed, false if session expired or otherwise invalid sessionId. -->
       <outfault ref="sessionFault"/>
    </operation>
```

NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its close-session-request operation, including the pertaining element and type definitions.

7.6.17 The suspend-session-request operation

A suspend-session-request operation may be present in a session-full socket. Its 'name' attribute shall have the value "suspendSessionRequest". It shall have the following input and output items:

— An input item named "sessionId" of type xs:NMTOKEN, specifying the session that is requested to be suspended.

— An input item named "suggestedTimeout" of type xs:duration, specifying a timeout suggested by the client.

— An output item named "tentativeTimeout" of type xs:duration, specifying a tentative timeout granted by the web service. tentativeTimeout may be different from suggestedTimeout. It shall be empty if the request for suspension has been denied,

— A session fault (see 7.6.14).

EXAMPLE 1 In WSDL1, the suspend-session-request operation is specified as follows:

```
    <operation name="suspendSessionRequest">
       <dc:description xml:lang="en">For a session-full socket that supports
suspension of sessions,
        a URC calls this operation to request for a suspension.</dc:description>
      <input message="suspendSessionInputMessage" />
      <!-- <wsdl-urc:sessionId> carries the identifier of the active session. -->
       <!-- <wsdl-urc:suggestedTimeout> carries the timeout suggested by the URC. May
be empty. -->
      <output message="tentativeTimeout" />
      <!-- <wsdl-urc:tentativeTimeout> carries the tentative timeout granted by the target
(maybe different
        from suggestedTimeout). Empty if request for suspension denied. -->
      <fault name="sessionFault" message="sessionStatus"/>
        </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its suspend-session-request operation, including the pertaining element, type and message definitions.

EXAMPLE 2 In WSDL2, the suspend-session-request operation is specified as follows:

```
    <operation name="suspendSessionRequest"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
       <dc:description xml:lang="en">For a session-full socket that supports
suspension of sessions,
        a URC calls this operation to request for a suspension.</dc:description>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
         <!-- Identifier for the session to suspend. -->
      <input messageLabel="In" element="wsdl-urc:suggestedTimeout" />
         <!-- Timeout suggested by the URC. May be empty. -->
      <output messageLabel="Out" element="wsdl-urc:tentativeTimeout" />
         <!-- Tentative timeout granted by the target (maybe different from
suggestedTimeout.
          Empty if request for suspension denied. -->
        <outfault ref="sessionFault"/>
    </operation>
```


NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its suspend-session-request operation, including the pertaining element and type definitions.

7.6.18 The resume-session-request operation

A resume-session-request operation may be present in a session-full socket. Its 'name' attribute shall have the value "resumeSessionRequest". It shall have the following input and output items:

— An input item named "sessionId" of type xs:NMTOKEN, specifying the session that is requested to be resumed.

— An output item named "requestResult" of type xs:boolean, It shall be "true" if the session has been resumed, otherwise "false".

— A session fault (see 7.6.14).

EXAMPLE 1 In WSDL1, the resume-session-request operation is specified as follows:

```
    <operation name="resumeSessionRequest">
      <dc:description xml:lang="en">For a session-full socket that supports
suspension of sessions,
        a URC calls this operation to resume a suspended session.</dc:description>
      <input message="sessionId" />
      <!-- <wsdl-urc:sessionId> carries the identifier of the active session. -->
      <output message="requestResult" />
      <!-- true if resumed, false if expired or otherwise denied. -->
      <fault name="sessionFault" message="sessionStatus"/>
    </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its resume-session-request operation, including the pertaining element, type and message definitions.

EXAMPLE 2 In WSDL2, the resume-session-request operation is specified as follows:

```
    <operation name="resumeSessionRequest"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
       <dc:description xml:lang="en">For a session-full socket that supports
suspension of sessions,
        a URC calls this operation to resume a suspended session.</dc:description>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
        <!-- identifier of the suspended session. -->
      <output messageLabel="Out" element="wsdl-urc:requestResult" />
        <!-- true if resumed, false if denied. -->
      <outfault ref="sessionFault"/>
    </operation>
```

NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its resume-session-request operation, including the pertaining element and type definitions.

7.6.19 The get-session-status operation

A get-session-status operation shall be present in a session-full socket. Its 'name' attribute shall have the value "getSessionStatus". It shall have the following input and output items:

— An input item named "sessionId" of type xs:NMTOKEN.

— An output item named "sessionStatus" of enumeration type, with the following allowed values:

  — "invalid": invalid session identifier;

  — "active": the session is currently active;

- "forwardRequested": the session is still active, but the target wants the client to close it and open a session with the forwardSocketName, as soon as possible;
- "expired": the session has expired (closed by client);
- "aborted": the session has been aborted by the target;
- "forwarded": the target has closed the session and wants the client to open a session with the forwardSocketName.

— An output item named "forwardSocketName" of type xs:anyURI. forwardSocketName shall have a non-empty value only if sessionStatus has the value "forwardRequested" or "forwarded"; otherwise it shall be empty. If provided, it shall specify forward target (WSDL file location) and forward socket (partition) in the following form: "[URL of target WSDL file]#wsdl.interface([partition name])" (see 7.6.15, output item forwardSocketName of operation 'open-session-request', for further explanations).

— An output item named "forwardSocketAuthorizationCode" of type xs:string. forwardSocketAuthorizationCode may have a non-empty value only if the output item forwardSocketName is non-empty; otherwise it shall be empty. If provided, it shall specify an authorization code required to open a session with the socket specified by forwardSocketName.

EXAMPLE 1 The following value for the output item "forwardSocketName" represents the partition (socket) "control" on a web service target with its WSDL file located at "http://example.com/wsdl":

http://example.com/wsdl#wsdl.interface(control)

EXAMPLE 2 In WSDL1, the get-session-status operation is specified as follows:

```
    <operation name="getSessionStatus">
      <dc:description xml:lang="en">For a session full socket, a URC should call this
operation if its name
        is listed in the getOperations output item of the operation get-updates.</
dc:description>
      <input message="sessionId" />
      <!-- Identifier for the session. -->
      <output message="sessionStatus-forwardUri"/>
      <!-- <wsdl-urc:sessionStatus> with allowed values:
        "invalid": invalid session identifier.
        "active": session is active.
        "forwardRequested": session is still active, but the target wants the client to
close it and open
           a session with the forwardUri, as soon as possible.
        "expired": session has expired (client has closed it).
        "aborted": target has aborted the session.
       "forwarded": target has closed the session and wants the client to open a session with
           the forwardUri given as next output item. -->
      <!-- <wsdl-urc:forwardSocketName>: WSDL file location and socket name for forwarding
in the following form:
      [URL of target WSDL file]#wsdl.interface([partition name]) -->
      <!-- <wsdl-urc:forwardSocketAuthorizationCode>: If forwarded, <wsdl-urc:forwardSocke
tAuthorizationCode>
        contains an authorization code for the socket that the URC is forwarded to.  May
be empty if no
        authorization required. -->
      </output>
    </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its get-session-status operation, including the pertaining element, type and message definitions.

EXAMPLE 3 In WSDL2, the get-session-status operation is specified as follows:

```
    <operation name="getSessionStatus"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
```



```
      wsdlx:safe="true">
      <dc:description xml:lang="en">For a session-full socket, a URC should call this
operation if its name
        is listed in the getOperations output item of the operation get-updates.</
dc:description>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
        <!-- Identifier for the session. -->
      <output messageLabel="Out" element="wsdl-urc:sessionStatus"/>
        <!-- Possible values:
          "invalid": invalid session identifier.
          "active": session is active.
          "forwardRequested": session is still active, but the target wants the client to
close it and open
             a session with the forwardUri, as soon as possible.
          "expired": session has expired (client has closed it).
          "aborted": target has aborted the session.
         "forwarded": target has closed the session and wants the client to open a session with
             the forwardUri given as next output item. -->
      <output messageLabel="Out" element="wsdl-urc:forwardSocketName">
        <!-- WSDL file location and socket name for forwarding in the following form:
          [URL of target WSDL file]#wsdl.interface([partition name]) -->
      </output>
      <output messageLabel="Out" element="wsdl-urc:forwardSocketAuthorizationCode">
        <!-- If forwarded, <wsdl-urc:forwardSocketAuthorizationCode> contains an
authorization code for the
          socket that the URC is forwarded to.  May be empty if no authorization required. -->
      </output>
    </operation>
```

NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its get-session-status operation, including the pertaining element and type definitions.

7.6.20 The get-updates operation

A socket shall provide a get-updates operation with name="getUpdates" for clients to poll periodically to find out whether the session status has changed (only for session-full sockets), and/or updates of variable values, command states, notification states and/or dynamic atomic resources (for variables, notifications and indices) have occurred. Clients can then call these get, "getstat", check, get-index and/or get-resources operations to retrieve the updated values.

The Web service shall use the HTTP long polling technique for the get-updates operation, i.e. it shall only return the operation if an update has occurred or if a timeout has been reached. A timeout of 30 seconds is recommended, i.e. the server should send an empty response to the client if no updates have occurred within 30 seconds after the client request.

NOTE The timeout of 30 seconds is based on IETF RFC 6202, which specifies best practices for implementing the HTTP long polling technique on HTTP servers.

Also, the client should continuously send get-updates requests to the server, i.e. it should immediately send the next get-updates request as soon as it has received the response of the previous one.

The get-updates operation shall have the following input and output items:

— For session-full sockets: An input item named "sessionId" of type xs:NMTOKEN, specifying the session that this request pertains to.

— An output item named "getOperations" whose type shall be a space-separated list of xs:NCNames. getOperations shall contain a list of the names of those get and getidx operations to be called by the client to retrieve updated values. For session-full sockets, it shall also include the name of the get-session-status operation if the session status has changed. Once get-updates has been called for a specific session, the web service shall clear this list, and in the next call to get-updates shall reflect new updates only.

— For session-full sockets: A session fault (see 7.6.14).

EXAMPLE 1 In WSDL1, the get-updates operation for a session-full socket is specified as follows:

```
    <operation name="getUpdates">
      <dc:description xml:lang="en">Generic operation for identifying updated session
status, variables,
        notifications and indices.  Clients should poll this operation continuously to find
out where updates
        have occurred, and the server shall use long polling with 30 seconds timeout
(see RFC 6202).
        If the server returns a non-empty list of operations, the client should call
those operations
        to retrieve the updated values.</dc:description>
      <input message="sessionId" />     <!-- May be empty for session-less targets. -->
      <output message="getOperations" />
      <!-- Space-separated list of get, getidx and getres functions to be called to
receive updated values.
        May include getSessionStatus if socket is session-full.-->
      <fault name="sessionFault" message="sessionStatus"/>
    </operation>
```

NOTE 1 See Annex A for a complete listing of a conforming WSDL1 document and its get-updates operation, including the pertaining element, type and message definitions.

EXAMPLE 3 In WSDL2, the get-updates operation for a session-full socket is specified as follows:

```
    <operation name="getUpdates"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <dc:description xml:lang="en">Generic operation for identifying updated session
status, variables,
        notifications and indices.  Clients should poll this operation continuously to find
out where updates
        have occurred, and the server shall use long polling with 30 seconds timeout
(see RFC 6202).
        If the server returns a non-empty list of operations, the client should call
those operations
        to retrieve the updated values.</dc:description>
      <input messageLabel="In" element="sessionId" />
      <output messageLabel="Out" element="getOperations" />
        <!-- Space-separated list of get, getidx and getres functions to be called to
receive updated values.
        May include getSessionStatus if socket is session-full.-->
      <outfault ref="sessionFault"/>
    </operation>
```

EXAMPLE 4 In WSDL2, the get-updates operation for a session-less socket is specified as follows:

```
    <operation name="getUpdates"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <dc:description xml:lang="en">Generic operation for identifying updated session
status, variables,
        notifications and indices.  Clients should poll this operation continuously to find
out where updates
        have occurred, and the server shall use long polling with 30 seconds timeout
(see RFC 6202).
        If the server returns a non-empty list of operations, the client should call
those operations
        to retrieve the updated values.</dc:description>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
      <output messageLabel="Out" element="wsdl-urc:getOperations" />
        <!-- Space-separated list of get, getidx and getres functions to be called to
receive updated values.
        May include getSessionStatus if socket is session-full.-->
      <outfault ref="sessionFault"/>
    </operation>
```


NOTE 2 See Annex A for a complete listing of a conforming WSDL2 document and its get-updates operation, including the pertaining element and type definitions.

7.6.21 Variables

7.6.21.1 General

For each variable in the socket, a pair of get and set operations shall be included and a get-resources operation may be included in the Web service partition representing the socket (see 6.4). For final and read-only variables, only a get operation shall be included, and a get-resources operation may be included.

The name for the get operation shall have the prefix “get.”, followed by the set path of the variable, followed by the variable identifier. The name for the set operation shall have the prefix “set.”, followed by the set path of the variable, followed by the variable identifier. The name of the get-resources operation shall have the prefix “getres.”, followed by the set path of the variable, followed by the variable identifier.

In Extended BNF notation:

— get operation name = “get.” , set path, variable identifier ;

— set operation name = “set.” , set path, variable identifier ;

— get-resource operation name = “getres.” , set path, variable identifier ;

where *variable identifier* is the value of the ‘id’ attribute of <variable>.

EXAMPLE 1 For a variable with id=”power” that is directly underneath the <uiSocket> root element, the pertaining get operation is named “get.power”, the set operation “set.power”, and the get-resources operation “getres.power”.

EXAMPLE 2 For a variable with id=”power” in a set with id=”powerControls”, underneath <uiSocket>, the pertaining get operation is named “get.powerControls.power”, the set operation “set.powerControls.power”, and the get-resources operation “getres.powerControls.power”.

EXAMPLE 3 For a variable with id=”power” in a set with id=”powerControls”, in a set with id=”dvd”, underneath <uiSocket>, the pertaining get operation is named “get.dvd.powerControls.power”, the set operation “set.dvd.powerControls.power” and the get-resources operation “getres.dvd.powerControls.power”.

The names of the get, set and get-resources operations shall not occur as value of the ‘name’ attribute on any other element in the Web service description file.

EXAMPLE 4 Get and set operations for a socket variable with id=”targetTemp”, directly underneath <uiSocket> (WSDL1 for a session-less socket). This example assumes the definition of the “empty” and “targetTemp” messages earlier in the WSDL document.

```
<operation name=”get.targetTemp”>
  <input message=”empty”/>
  <output message=”targetTemp”/>
</operation>

<operation name=”set.targetTemp”>
  <input message=”targetTemp”/>
  <output message=”targetTemp”/>
</operation>
```

EXAMPLE 5 Get and set operations for a socket variable with id=”targetTemp”, directly underneath <uiSocket> (WSDL2 for a session-less socket). This example assumes the definition of the “targetTemp” element in the <types> section of the WSDL document.

```
<operation name=”get.targetTemp”
  pattern=”http://www.w3.org/ns/wsdl/in-out”
  style=”http://www.w3.org/ns/wsdl/style/iri”
```

```
      wsdlx:safe = "true">
      <output messageLabel="Out" element="targetTemp" />
    </operation>

    <operation name="set.targetTemp"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <input messageLabel="In" element="targetTemp" />
      <output messageLabel="Out" element="targetTemp" />
    </operation>
```

7.6.21.2 Value representation

Values of variables shall be represented based on the built-in data types of XML Schema Definition, where appropriate.

For string-based values, an empty string ("") shall be represented as empty element content, with attribute xsi:null="false" or not present.

For any types, the undefined value shall be represented as an empty element with attribute xsi:null="true".

NOTE Use nillable="true" in the corresponding type definitions.

7.6.21.3 The get operation

7.6.21.3.1 General

In WSDL1, a get operation of a variable shall have no input item, and an output item carrying the variable's value.

EXAMPLE 1 WSDL1 for a session-less socket: A get operation for the read-only variable "currentRoomTemp" of type xs:float. The variable is not contained in a set. All relevant parts of the WSDL1 document are listed. Ellipses (...) indicate omissions.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc" ... >
  ...
  <types>
     <xs:schema targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/">
       <xs:element name="currentRoomTemp" type="xs:float" nillable="true"/>
       ...
     </xs:schema>
  </types>

  <message name="currentRoomTemp">
    <part name="currentRoomTemp" element="currentRoomTemp"/>
  </message>
  ...
  <portType name="socket">
    <operation name="get.currentRoomTemp">
      <input message="empty"/>
      <output message="currentRoomTemp"/>
    </operation>
    ...
  </portType>

  <binding name="socketSoapBinding" type="socket">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="get.currentRoomTemp">
      <soap:operation soapAction="http://example.com/thermostat/socket/get.
currentRoomTemp"/>
      <input>
         <soap:body use="literal"/>
      </input>
      <output>
```



```
        <soap:body use="literal"/>
      </output>
      <fault name="sessionFault">
        <soap:body use="literal"/>
      </fault>
    </operation>
    ...
  </binding>

  <service name="socketService">
    <port name="socket" binding="tns:socketSoapBinding">
      <soap:address location="http://example.com/thermostat/socket"/>
    </port>
  </service>

</definitions>
```

In WSDL2, a get operation of a variable shall specify the pattern "http://www.w3.org/ns/wsdl/in-out", and the style "http://www.w3.org/ns/wsdl/style/iri".

EXAMPLE 2 WSDL2 for a session-less socket: A get operation for the read-only variable "currentRoomTemp" of type xs:float. The variable is not contained in a set. All relevant parts of the WSDL2 document are listed. Ellipses (...) indicate omissions.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc"
  xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap" ... >
  ...
  <types>
    <!-- Schema Types for input, output and outfault message elements.-->
    <xs:schema targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/">
      <xs:element name="currentRoomTemp" type="xs:float" nillable="true"/>
      ...
    </xs:schema>
  </types>
  ...
  <interface name="socket">
    ...
    <!-- Operation for variable with id="currentRoomTemp" -->

    <operation name="get.currentRoomTemp"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
      wsdlx:safe = "true">
      <output messageLabel="Out" element="currentRoomTemp" />
    </operation>
    ...
  </interface>

  <binding name="socketSoapBinding"
    interface="socket"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="get.currentRoomTemp"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    ...
  </binding>

  <service name="socketService"
    interface="socket">
    <endpoint name="socketEndpoint"
      binding="socketSOAPBinding"
      address ="http://example.com/thermostat/socket"/>
  </service>
  ...
</description>
```

7.6.21.3.2 The <wsdl-urc:secret> element

A get operation of a variable may have a <wsdl-urc:secret> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘secret’ attribute on a <uis:variable> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A variable marked as secret (WSDL1 and WSDL2).

```
<documentation>
<wsdl-urc:secret>true</wsdl-urc:secret>
</documentation>
```

7.6.21.3.3 The <wsdl-urc:sensitive> element

A get operation of a variable may have a <wsdl-urc:sensitive> element as subelement of the <documentation>. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘sensitive’ attribute on a <uis:variable> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A variable marked as sensitive (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:sensitive>true</wsdl-urc:sensitive>
</documentation>
```

7.6.21.3.4 The <wsdl-urc:optional> element

A get operation of a variable may have a <wsdl-urc:optional> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘optional’ attribute on a <uis:variable> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A variable marked as optional (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:optional>true</wsdl-urc:optional>
</documentation>
```

7.6.21.3.5 The <wsdl-urc:final> element

A get operation of a variable may have a <wsdl-urc:final> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘final’ attribute on a <uis:variable> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A variable marked as final (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:final>true</wsdl-urc:final>
</documentation>
```

NOTE A final variable is read-only and therefore has no set operation.

7.6.21.3.6 Variable dependencies

All dependencies of a variable shall be attached to its get operation, as follows.

A get operation of a variable may have a <wsdl-urc:dependency> element as subelement of the <documentation> element. If present, it may have any number of the following subelements, each occurring zero or one time:

— <wsdl-urc:relevant> with a valid XPath expression evaluating to a Boolean value, as defined in part 2, section “Variable dependencies”.

— <wsdl-urc:write> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

NOTE A read-only variable with a write dependency of “false()” has no set operation.

— <wsdl-urc:insert> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

NOTE A final variable with dimensions can have an insert dependency (for removing indices), but no set operation.

— <wsdl-urc:length> with a valid XPath expression evaluating to a number, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:minLength> with a valid XPath expression evaluating to a number, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:maxLength> with a valid XPath expression evaluating to a number, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:pattern> with a valid XPath expression evaluating to a string, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:minInclusive> with a valid XPath expression evaluating to a value of the same type as the variable it belongs to, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:maxInclusive> with a valid XPath expression evaluating to a value of the same type as the variable it belongs to, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:minExclusive> with a valid XPath expression evaluating to a value of the same type as the variable it belongs to, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

— <wsdl-urc:maxExclusive> with a valid XPath expression evaluating to a value of the same type as the variable it belongs to, as defined in ISO/IEC 24752-2, section “Variable dependencies”.

7.6.21.3.7 The <wsdl-urc:selection> element

All selection sets of a variable shall be attached to its get operation, as follows.

A get operation of a variable may have a <wsdl-urc:selection> element as subelement of the <documentation> element. If present, it shall conform to the <selection> element in a socket description (see ISO/IEC 24752-2).

NOTE A selection is used to either restrict a variable’s value space (closed selection) or to provide suggested values for user input (open selection).

The <wsdl-urc:selection> element may contain any number of static and dynamic selection sets as subelements, each zero or more times.

Static selection sets shall be represented as the <wsdl-urc:selectionSetStatic> element, with syntax as described in ISO/IEC 24752-2, section “The <selectionSetStatic> element”. However, the attribute ‘name’ shall be present in place of the ‘id’ attribute, and its value shall be unique among all ‘name’ values in the WSDL document.

The attribute 'typeRef' shall reference a type that is either defined in the WSDL file or otherwise included by type import (<xs:include>) or external namespace reference (<xs:import>). A fully qualified name shall be used to reference external type definitions.

EXAMPLE 1 A variable with a closed static selection set, referencing the local type operatingModeType (WSDL1 and WSDL2).

```
    <documentation>
      <wsdl-urc:selection closed="true">
        <wsdl-urc:selectionSetStatic name="operatingModeSelection"
typeRef="operatingModeType" />
      </wsdl-urc:selection>
    </documentation>
```

Dynamic selection sets shall be represented as the <wsdl-urc:selectionSetDynamic> element, with syntax as described in ISO/IEC 24752-2, section "The <selectionSetDynamic> element". However, the attribute 'name' shall be present in place of the 'id' attribute, and its value shall be unique among all 'name' values in the WSDL document.

The attribute 'varRef' shall reference a variable of the pertaining socket (as described by the Web service partition).

EXAMPLE 2 A variable with an open dynamic selection set, referencing a list variable (WSDL1 and WSDL2).

```
    <documentation>
      <wsdl-urc:selection closed="false">
        <wsdl-urc:selectionSetDynamic name="operatingModeSelection"
varRef="operatingModeList" />
      </wsdl-urc:selection>
    </documentation>
```

7.6.21.3.8 Properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a get operation, if appropriate (as specified in ISO 15836). Each of them may occur any number of times as subelement of the operation.

The 'xsi:type' attribute should be used to identify the coding schema, if appropriate.

EXAMPLE A description in English language is provided for a get operation (WSDL1 and WSDL2). Ellipses ("...") indicate omissions.

```
    <operation name="get.operatingMode">
      ...
      <dc:description xml:lang="en">Get operation for UIS variable id="operatingMode"</
dc:description>
      ...
    </operation>
```

7.6.21.3.9 The "sessionId" input item

A get operation that belongs to a session-full socket shall have an input item named "sessionId" of type xs:NMTOKEN, specifying the identifier of the pertaining session.

EXAMPLE 1 In WSDL1, if the input item "sessionId" is the only input item, it is specified as follows (see 7.6.21.3.1 for a full example of a get operation):

```
    <input message="sessionId" />
```

EXAMPLE 2 In WSDL2, the input item "sessionId" is specified as follows:

```
    <input messageLabel="In" element="sessionId" />
```


7.6.21.3.10 The output item

A get operation of a variable shall have one output item of the variable's type, specifying the current value of the variable.

EXAMPLE 1 In WSDL1, the output item "currentRoomTemp" for a get operation of the read-only variable "currentRoomTemp" of type xs:float is specified as follows (see 7.6.21.3.1 for a full example of a get operation):

```
<output message="currentRoomTemp"/>
```

EXAMPLE 2 In WSDL2, the output item "currentRoomTemp" for a get operation of the read-only variable "currentRoomTemp" of type xs:float is specified as follows (see 7.6.21.3.1 for a full example of a get operation):

```
<output messageLabel="Out" element="currentRoomTemp" />
```

NOTE To indicate unavailability of the variable at runtime, the Web service can return the undefined value as output item (see 7.6.21.2).

7.6.21.3.11 The fault item

A get operation that belongs to a session-full socket shall specify a session fault item (see 7.6.14) to indicate session faults.

EXAMPLE 1 In WSDL1, the fault item "sessionFault" for a get operation is specified as follows (see 7.6.21.3.1 for a full example of a get operation):

```
<fault name="sessionFault" message="sessionStatus"/>
```

EXAMPLE 2 In WSDL2, a get operation references the interface fault item "sessionFault", as follows (see 7.6.21.3.1 for a full example of a get operation):

```
<outfault ref="sessionFault"/>
```

7.6.21.4 The set operation

7.6.21.4.1 General

EXAMPLE 1 WSDL1 for a session-less socket: A set operation for the variable "targetTemp" of type xs:float. The variable is not contained in a set. All relevant parts of the WSDL1 document are listed. Ellipses (...) indicate omissions.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc" ... >
  ...
  <types>
    <xs:schema targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/">
      <xs:element name="targetTemp" type="xs:float" nillable="true"/>
      ...
    </xs:schema>
  </types>

  <message name="targetTemp">
    <part name="targetTemp" element="targetTemp"/>
  </message>
  ...
  <portType name="socket">
    <operation name="set.targetTemp">
      <input message="targetTemp"/>
      <output message="targetTemp"/>
    </operation>
    ...
  </portType>
```

```
  <binding name="socketSoapBinding" type="socket">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="set.targetTemp">
      <soap:operation soapAction="http://example.com/thermostat/socket/set.targetTemp"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    ...
  </binding>

  <service name="socketService">
    <port name="socket" binding="tns:socketSoapBinding">
      <soap:address location="http://example.com/thermostat/socket"/>
    </port>
  </service>

</definitions>
```

In WSDL2, a set operation of a variable shall specify the pattern "http://www.w3.org/ns/wsdl/in-out", and the style "http://www.w3.org/ns/wsdl/style/iri".

EXAMPLE 2 WSDL2 for a session-less socket: A set operation for the variable "targetTemp" of type xs:float. The variable is not contained in a set. All relevant parts of the WSDL2 document are listed. Ellipses (...) indicate omissions.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc"
  xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap" . . >
  ...
  <types>
    <xs:schema targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/">
      <xs:element name="targetTemp" type="xs:float" nillable="true"/>
      ...
    </xs:schema>
  </types>
  ...
  <interface name="socket">
    ...
    <operation name="set.targetTemp"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <input messageLabel="In" element="targetTemp" />
      <output messageLabel="Out" element="targetTemp" />
    </operation>
    ...
  </interface>

  <binding name="socketSoapBinding"
    interface="socket"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="set.targetTemp"
    wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    ...
  </binding>

  <service name="socketService"
    interface="socket">
    <endpoint name="socketEndpoint"
      binding="socketSOAPBinding"
      address ="http://example.com/thermostat/socket"/>
  </service>
  ...
```



```
</description>
```

7.6.21.4.2 Properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a set operation, if appropriate (as specified in ISO 15836). Each of them may occur any number of times as subelement of the operation.

The 'xsi:type' attribute should be used to identify the coding schema, if appropriate.

EXAMPLE A description in English language is provided for a set operation for a socket variable "volume" (WSDL1 and WSDL2).

```
    <operation name="set.volume">
      <dc:description xml:lang="en">Sets the volume in the range of 0 to 100.</
dc:description>
    </operation>
```

7.6.21.4.3 The "sessionId" input item

A set operation that belongs to a session-full session shall have an input item named "sessionId" of type xs:NMTOKEN, specifying the identifier of the pertaining session.

NOTE See 7.6.21.3.9 for examples in WSDL1 and WSDL2.

7.6.21.4.4 The input and output items for the variable's value

A set operation shall have an input and an output element, both of the variable's type. They shall carry the requested and the resulting value of the variable, respectively.

NOTE 1 See 7.6.21.4.1 for examples in WSDL1 and WSDL2.

NOTE 2 To indicate unavailability of the variable at runtime, the Web service can return the undefined value as output item (see 7.6.21.2).

7.6.21.4.5 The fault item

A set operation that belongs to a session-full socket shall specify a session fault item (see 7.6.14) to indicate session faults.

NOTE See 7.6.21.4.1 for examples in WSDL1 and WSDL2.

7.6.21.5 The get-resources operation

7.6.21.5.1 General

EXAMPLE 1 WSDL1 for a session-less socket: A get-resources operation for the variable "operatingMode" of type uiStringListItem. The variable is not contained in a set. All relevant parts of the WSDL1 document are listed. Ellipses (...) indicate omissions.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc" ... >
  ...
  <message name="resourceDescriptions">
    <part name="resItems" element="wsdl-urc:resItems"/>
  </message>
  ...
  <portType name="socket">
    <operation name="getres.operatingMode">
      <dc:description xml:lang="en">Operation for a client to retrieve dynamic atomic
resources pertaining
        to the socket variable 'operatingMode'.</dc:description>
```

```
      <input message="empty" />
      <output message="resourceDescriptions" />
    </operation>
    ...
  </portType>

  <binding name="socketSoapBinding" type="socket">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getres.operatingMode">
      <soap:operation soapAction="http://example.com/thermostat/socket/getres.
operatingMode"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    ...
  </binding>

  <service name="socketService">
  <port name="socket" binding="tns:socketSoapBinding">
    <soap:address location="http://example.com/thermostat/socket"/>
    </port>
  </service>

</definitions>
```

In WSDL2, a get-resources operation of a variable shall specify the pattern "http://www.w3.org/ns/wsdl/in-out", and the style "http://www.w3.org/ns/wsdl/style/iri".

EXAMPLE 2 WSDL2 for a session-less socket: A get-resources operation for the variable "operatingMode" of type uis:stringListItem. The variable is not contained in a set. All relevant parts of the WSDL2 document are listed. Ellipses (...) indicate omissions.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc"
  xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap" ... >
  ...
  <interface name="socket">
    ...
    <operation name="getres.operatingMode"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <dc:description xml:lang="en">Operation for a client to retrieve dynamic atomic
resources pertaining
        to the socket variable 'operatingMode'.</dc:description>
      <output messageLabel="Out" element="wsdl-urc:resItems" />
    </operation>
    ...
  </interface>

  <binding name="socketSoapBinding"
    interface="socket"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="getres.operatingMode"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    ...
  </binding>

  <service name="socketService"
    interface="socket">
    <endpoint name="socketEndpoint"
      binding="socketSOAPBinding"
      address ="http://example.com/thermostat/socketService"/>
```

```
  </service>
  ...
</description>
```

7.6.21.5.2 Properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a get-resources operation, if appropriate (as specified in ISO 15836). Each of them may occur any number of times as subelement of the operation.

The 'xsi:type' attribute should be used to identify the coding schema, if appropriate.

EXAMPLE A description is provided for a get-resources operation (WSDL1 and WSDL2).

```
  <operation name="getres.deviceStatus">
    <dc:description xml:lang="en">Get text and icon labels for the runtime values of
"deviceStatus".</dc:description>
  </operation>
```

7.6.21.5.3 The "sessionId" input item

A get-resources operation that belongs to a session-full session shall have an input item named "sessionId" of type xs:NMTOKEN, specifying the identifier of the pertaining session.

NOTE See 7.6.21.3.9 for examples in WSDL1 and WSDL2.

7.6.21.5.4 The output items for the dynamic atomic resources

A get-resources operation shall have one output item of element "wsdl-urc:resItems" (see 6.4.4).

NOTE 1 See 7.6.21.5.1 for examples in WSDL1 and WSDL2.

NOTE 2 To indicate unavailability of the variable at runtime, the Web service can return the undefined value as output item (see 7.6.21.2).

7.6.21.5.5 The fault item

A get-resources operation that belongs to a session-full socket shall specify a session fault item (see 7.6.14) to indicate session faults.

NOTE See 7.6.21.4.1 for examples in WSDL1 and WSDL2.

7.6.22 Commands

7.6.22.1 General

For each command in the socket, a command operation shall be included in the Web service partition representing the socket (see 6.5). If the command's type is other than 'uis:voidCommand', the partition shall also include a get-status operation. A get-resources operation may be included if the Web service wants to provide dynamic atomic resources for the command.

7.6.22.2 Value representation

Values of variables shall be represented based on the built-in data types of XML Schema Definition, where appropriate.

For string-based values, an empty string ("") shall be represented as empty element content, with attribute xsi:null="false" or not present.

For any types, the undefined value shall be represented as an empty element with attribute xsi:null="true".

NOTE Use nillable="true" in the corresponding type definitions.

7.6.22.3 The command operation

EXAMPLE 1 WSDL1 code snippet for a session-less socket: A get operation for a read-only variable "numberOfTicketsIssuedToday" of type "xs:integer", and a command operation for a command "orderTicket". The command "orderTicket" has a global output parameter "numberOfTicketsIssuedToday" (referencing the variable by name), one local input parameter "creditCardNumber", one local input-output parameter "time", and one local output parameter "transactionNumber". Note that the parameter "time" occurs twice as input and as output item.

```
<!-- Get operation for variable with id="numberOfTicketsIssuedToday" -->
<operation name="get.numberOfTicketsIssuedToday">
  <documentation>
    <wsdl-urc:dependency>
      <wsdl-urc:write>false()</wsdl-urc:write>
    </wsdl-urc:dependency>
  </documentation>
  <input message="empty" />
  <output message="numberOfTicketsIssuedToday" />
</operation>

<!-- Operation for command with id="orderTicket" -->
<operation name="cmd.orderTicket">
  <documentation>
    <wsdl-urc:globalParam idref="numberOfTicketsIssuedToday" dir="out"/>
  </documentation>
  <input message="cmd.orderTicket.inputmessage"/>
  <output message="cmd.orderTicket.outputmessage"/>
</operation>
```

EXAMPLE 2 Same as example 1, but for WSDL2:

```
<!-- Get operation for variable with id="numberOfTicketsIssuedToday" -->
<operation name="get.numberOfTicketsIssuedToday">
  <documentation>
    <wsdl-urc:dependency>
      <wsdl-urc:write>false()</wsdl-urc:write>
    </wsdl-urc:dependency>
  </documentation>
  <!-- No input item -->
  <output messageLabel="Out" element="numberOfTicketsIssuedToday" />
</operation>

<!-- Operation for command with id="orderTicket" -->
<operation name="cmd.orderTicket">
  <documentation>
    <wsdl-urc:globalParam idref="numberOfTicketsIssuedToday" dir="out"/>
  </documentation>
  <input messageLabel="In" element="creditCardNumber">
    <documentation>
      <wsdl-urc:secret>true</wsdl-urc:secret>
    </documentation>
  </input>
  <input messageLabel="In" element="time"/>
  <output messageLabel="Out" element="_status"/>
  <output messageLabel="Out" element="time"/>
  <output messageLabel="Out" element="transactionNumber">
    <documentation>
      <wsdl-urc:sensitive>true</wsdl-urc:sensitive>
    </documentation>
  </output>
</operation>
```

If the command is of type uis:voidCommand, the command operation may be synchronous or asynchronous. For all other command types, the command operation shall be synchronous.

The name of the command operation shall have the prefix "cmd.", followed by the set path of the command, followed by the command identifier.

In Extended BNF notation:

command operation name = “cmd.” , set path, command identifier ;

where *command identifier* is the value of the ‘id’ attribute of <command>.

EXAMPLE 1 For a command with id=”resetProgram” under <uiSocket>, the pertaining operation is named “cmd.resetProgram”.

EXAMPLE 2 For a command with id=”resetProgram”, in a set with id=”advancedFunctions”, under <uiSocket>, the pertaining operation is named “cmd.advancedFunctions.resetProgram”.

The name of the command operation shall not occur as value of the ‘name’ attribute on any other element in the Web service description file.

7.6.22.3.1 The <wsdl-urc:sensitive> element

A command operation may have a <wsdl-urc:sensitive> element as subelement of the <documentation>. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘sensitive’ attribute on a <uis:command> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A command marked as sensitive (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:sensitive>true</wsdl-urc:sensitive>
</documentation>
```

7.6.22.3.2 The <wsdl-urc:sufficient> element

A command operation may have a <wsdl-urc:sufficient> element as subelement of the <documentation>. If present, it shall have a Boolean value as content, i.e. “true” or “false”.

This corresponds to the ‘sufficient’ attribute on a <uis:command> element in a socket description (see ISO/IEC 24752-2).

If present, the <wsdl-urc:sufficient> element of the command operation overrides the <wsdl-urc:sufficient> element of the socket (see 7.6.10), but only for that particular command. If not present, the value of the socket’s <wsdl-urc:sufficient> element applies to the command.

EXAMPLE A command marked as sufficient (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:sufficient>true</wsdl-urc:sufficient>
</documentation>
```

7.6.22.3.3 The <wsdl-urc:complete> element

A command operation may have a <wsdl-urc:complete> element as subelement of the <documentation>. If present, it shall have a Boolean value as content, i.e. “true” or “false”.

This corresponds to the ‘complete’ attribute on a <uis:command> element in a socket description (see ISO/IEC 24752-2).

If present, the <wsdl-urc:complete> element of the command operation overrides the <wsdl-urc:complete> element of the socket (see 7.6.11), but only for that particular command. If not present, the value of the socket’s <wsdl-urc:complete> element applies to the command.

EXAMPLE A command marked as complete (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:complete>true</wsdl-urc:complete>
</documentation>
```

7.6.22.3.4 The <wsdl-urc:optional> element

A command operation may have a <wsdl-urc:optional> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. “true” or “false”. The default value is “false”.

This corresponds to the ‘optional’ attribute on a <uis:command> element in a socket description (see ISO/IEC 24752-2).

EXAMPLE A command marked as optional (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:optional>true</wsdl-urc:optional>
</documentation>
```

7.6.22.3.5 Command dependencies

A command operation may have a <wsdl-urc:dependency> element as subelement of the <documentation> element. If present, it may have any number of the following subelements, each occurring at most once:

— <wsdl-urc:relevant> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Command dependencies”.

— <wsdl-urc:write> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Command dependencies”.

— <wsdl-urc:insert> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Command dependencies”.

— <wsdl-urc:assert> with a valid XPath expression evaluating to a Boolean value, as defined in ISO/IEC 24752-2, section “Command dependencies”.

7.6.22.3.6 Command properties from DCMI

Any element and element refinement from the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a command operation, if appropriate (as specified in ISO 15836). Each of them may occur any number of times as subelement of the <operation> element of the command operation.

The ‘xsi:type’ attribute should be used to identify the coding schema, if appropriate.

EXAMPLE A description in English language is provided for a command operation (WSDL1 and WSDL2) - omissions are indicated by elipses.

```
<operation name=”cmd:resetProgram”>
  <documentation>
  ...
  </documentation>
  <dc:description xml:lang=”en”>This command resets all programmed thermostat
settings.</dc:description>
  ...
</operation>
```

7.6.22.3.7 The <wsdl-urc:globalParam> element

A command operation may have one or more <wsdl-urc:param> elements as subelement of the <documentation> element. If present, any of them shall reflect a global command parameter, as specified in ISO/IEC 24752-2, section “Command parameters”.

EXAMPLE A global input parameter referencing the variable “power” in the socket (WSDL1 and WSDL2).

```
<documentation>
  <wsdl-urc:globalParam idref=”power” dir=”in” />
</documentation>
```

NOTE The order of the input items is not significant since they are identified by names.

7.6.22.3.7.1 The 'idref' attribute

A <wsdl-urc:globalParam> element shall have an 'idref' attribute, with its value referencing the identifier of a variable, as specified in part 2, section "The 'idref' attribute (global parameter)".

7.6.22.3.7.2 The 'dir' attribute

A <wsdl-urc:globalParam> element shall have an 'dir' attribute, with one of the values: "in", "out", or "inout".

This corresponds to the 'dir' attribute on a global parameter in a socket description (see part 2, section "The 'dir' attribute").

7.6.22.3.8 The "sessionId" input item

A command operation that belongs to a session-full socket shall have an input item named "sessionId" of type xs:NMTOKEN, specifying the identifier of the pertaining session.

NOTE See 7.6.22.1 for examples in WSDL1 and WSDL2.

7.6.22.3.9 The input items for the command's parameters

The input items of the command operation shall reflect the local input and local input-output parameters of the command, as an ordered list. In the WSDL document, there shall be one input item for each local input and local input-output parameter.

NOTE See 7.6.22.1 for examples in WSDL1 and WSDL2.

7.6.22.3.9.1 The <wsdl-urc:secret> element

An input item reflecting a command parameter may be marked as secret. In WSDL1, the pertinent <part> element of the input item may have a <documentation> element with a <wsdl-urc:secret> element. In WSDL2, the pertinent <input> element may have a <documentation> element with a <wsdl-urc:secret> element.

If such a <wsdl-urc:secret> element is present, it shall have a Boolean value as content, i.e. "true" or "false". The default value is "false".

This corresponds to the 'secret' attribute on a command parameter in a socket description (see ISO/IEC 24752-2).

7.6.22.3.9.2 The <wsdl-urc:sensitive> element

An input item reflecting a command parameter may be marked as sensitive. In WSDL1, the pertinent <part> element of the input item may have a <documentation> element with a <wsdl-urc:sensitive> element. In WSDL2, the pertinent <input> element may have a <documentation> element with a <wsdl-urc:sensitive> element.

If such a <wsdl-urc:sensitive> element is present, it shall have a Boolean value as content, i.e. "true" or "false". The default value is "false".

This corresponds to the 'sensitive' attribute on a command parameter in a socket description (see ISO/IEC 24752-2).

7.6.22.3.9.3 The <wsdl-urc:selection> element

An input item reflecting a command parameter may have a set of selections. In WSDL1, the pertinent <part> element of the input item may have a <documentation> element with a <wsdl-urc:selection> element. In WSDL2, the pertinent <input> element may have a <documentation> element with a <wsdl-urc:selection> element.

If such a <wsdl-urc:selection> element is present, it shall conform to the <selection> element on a command parameter in a socket description (see ISO/IEC 24752-2), but with subelements <wsdl-urc:selectionSetStatic> and <wsdl-urc:selectionSetDynamic> in place of <selectionSetStatic> and <selectionSetDynamic>.

NOTE A selection is used to either restrict a command parameter's value space (closed selection) or to provide suggested values for user input (open selection).

7.6.22.3.10 The output item for the command's status

A command operation shall have an output item reflecting the command's status (return value), depending on the command type as follows:

— If the command is of type "uis:voidCommand", there shall be no output items corresponding to the command's status.

— If the command is of type "uis:basicCommand" or "uis:timedCommand", there shall be an output item of element "wsdl-urc:commandStatus".

— If the command is of type "uis:timedCommand", there shall be an additional output item of element "wsdl-urc:timeToComplete".of type xs:duration.

NOTE 1 See 7.6.22.1 for examples in WSDL1 and WSDL2.

In the pertaining user interface socket description, none of the command parameters (input, output, and input-output) shall have the name "commandStatus" or "timeToComplete".

NOTE 2 To indicate unavailability of the command at runtime, the Web service can return the undefined value as 'status' output item (see 7.6.22.2).

7.6.22.3.11 The output items for the command's parameters

For each local output and input-output parameter of the command, an output item shall be be provided by the command operation. For each of them, the element shall be named after the parameter name. In WSDL1, this is the 'element' attribute value of the pertaining <part> element in a <message> element referenced from the <output> element. In WSDL2, this is the 'element' attribute value on the pertaining <output> element.

NOTE 1 An input-output parameters occurs twice within a command operation, first as input item, and second as output item.

NOTE 2 The order of the output items is not significant since they are identified by names.

NOTE 3 See 7.6.22.1 for examples in WSDL1 and WSDL2.

7.6.22.3.11.1 The <wsdl-urc:secret> element

An output item reflecting a local output or input-output command parameter may have a <wsdl-urc:secret> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. "true" or "false". The default value is "false".

This corresponds to the 'secret' attribute on a command parameter in a socket description (see ISO/IEC 24752-2).

7.6.22.3.11.2 The <wsdl-urc:sensitive> element

An output item reflecting a local output or input-output command parameter may have a <wsdl-urc:sensitive> element as subelement of the <documentation> element. If present, it shall have a Boolean value as content, i.e. "true" or "false". The default value is "false".

This corresponds to the 'sensitive' attribute on a command parameter in a socket description (see ISO/IEC 24752-2).

7.6.22.3.12 The fault item

A command operation that belongs to a session-full socket shall specify a session fault item (see 7.6.14) to indicate session faults.

NOTE See 7.6.22.1 for examples in WSDL1 and WSDL2.

7.6.22.4 The get-status operation

For every socket command other than of type uis:voidCommand, there shall be a get-status operation defined, as follows:

— The name of the operation shall be as follows (in Extended BNF notation): operation name = "getstat." , set path of command , command identifier;

— if the operation belongs to a session-full socket, it shall have an input item named "sessionId" of type xs:NMTOKEN, specifying the identifier of the pertaining session;

— the operation shall have an output item of element <wsdl-urc:commandStatus>, indicating the current status of the command;

— if the command is of type uis:timedCommand, the operation shall have a second output item <wsdl-urc:timeToComplete> of type xs:duration, reflecting the target's estimate on the time needed to complete the pertaining command (see ISO/IEC 24752-2, section "uis:timedCommand");

— if the operation belongs to a session-full socket, it shall have a fault item (see 7.6.14) to indicate session faults;

vthe operation may have properties from DCMI, as specified in 7.6.21.3.8.

NOTE 1 Changes in the status or time-to-complete value of a command will be indicated when a client calls the get-updates operation (7.6.20).

NOTE 2 To indicate unavailability of the command at runtime, the Web service can return the undefined value as 'commandStatus' output item (see 7.6.22.2).

EXAMPLE 1 A get-status operation for the command "orderTicket" in WSDL1.

```
    <operation name="getstat.orderTicket">
      <dc:description xml:lang="en">Get the current status of the 'orderTicket' command.</
dc:description>
      <input message="sessionId"/>
      <output message="commandStatus"/>
      <fault name="sessionFault" message="sessionStatus"/>
    </operation>
```

EXAMPLE 2 A get-status operation for the command "orderTicket" in WSDL2.

```
    <operation name="getstat.orderTicket">
      <input messageLabel="In" element="sessionId" />
      <output messageLabel="Out" element="wsdl-urc:commandStatus"/>
      <output messageLabel="Out" element="wsdl-urc:timeToComplete"/>
      <outfault ref="sessionFault"/>
    </operation>
```

7.6.22.5 The get-resources operation

The get-resources operation for a socket command, if present, shall be the same as for variables (see 7.6.21.5).

NOTE To indicate unavailability of the command at runtime, the Web service can return the undefined value as output item (see 7.6.22.2).

7.6.23 Notifications

7.6.23.1 General

For each notification in the socket, a check operation shall be included in the Web service partition representing the socket (see 6.6). If the notification has a 'timeout' attribute, an extend-timeout operation shall be included, and otherwise it may be included. If the notifications's type is other than "show", the partition shall also include an acknowledge operation. A get-resources operation may be included if the Web service wants to provide dynamic atomic resources for the notification.

The name of the check operation shall have the prefix "chk.", followed by the set path of the notification, followed by the notification identifier. The name of the acknowledge operation shall have the prefix "ack.", followed by the set path of the notification, followed by the notification identifier. The name of the extend-timeout operation shall have the prefix "ext.", followed by the set path of the notification, followed by the notification identifier. The name of the get-resources operation shall have the prefix "getres.", followed by the set path of the notification, followed by the notification identifier.

In Extended BNF notation:

— check operation name = "chk." , set path, notification identifier ;

— acknowledge operation name = "ack." , set path, notification identifier ;

— extend-timeout operation name = "ext." , set path, notification identifier ;

— get-resources operation name = "getres." , set path, notification identifier ;

where *notification identifier* is the value of the 'id' attribute of <notify>.

EXAMPLE 1 For a notification with id="errorOccurred" under <uiSocket>, the pertaining check operation is named "chk.errorOccurred", and the acknowledge operation "ack.errorOccurred".

EXAMPLE 2 For a notification with id="errorOccurred" in a set with id="errors", under <uiSocket>, the pertaining check operation is named "chk.errors.errorOccurred", and the acknowledge operation "ack.errors.errorOccurred".

The names of the check, extend-timeout, acknowledge and get-resources operations shall not occur as value of the 'name' attribute on any other element in the Web service description file.

7.6.23.2 Value representation

Values of notifications shall be represented based on the built-in data types of XML Schema Definition, where appropriate.

For string-based values, an empty string ("") shall be represented as empty element content, with attribute xsi:null="false" or not present.

For any types, the undefined value shall be represented as an empty element with attribute xsi:null="true".

NOTE Use nillable="true" in the corresponding type definitions.

7.6.23.3 The check operation

7.6.23.3.1 General

EXAMPLE 1 WSDL1 code for a check operation for a notification with id="confirmReset", of type "confirmCancel", and of category "alert". The notification is not contained in a set. The socket is session-full. Ellipses ("...") indicate omissions.

```
<definitions targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```



```
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc" ...>
  ...
  <message name="sessionId">
    <part name="sessionId" element="wsdl-urc:sessionId"/>
  </message>
  <message name="notifyStatus">
    <part name="notifyStatus" element="wsdl-urc:notifyStatus" />
  </message>
  ...
  <portType name="socket">
    ...
    <operation name="chk.confirmReset">
      <documentation>
        <wsdl-urc:category>alert</wsdl-urc:category>
        <wsdl-urc:sensitive>false</wsdl-urc:sensitive>
        <wsdl-urc:optional>false</wsdl-urc:optional>
      </documentation>
      <input message="sessionId" />
      <output message="notifyStatus" />
      <fault name="sessionFault" message="sessionStatus"/>
    </operation>
    ...
  </portType>
  ...
</definitions>
```

EXAMPLE 2 WSDL2 code for example 1. Ellipses ("...") indicate omissions.

```
<description targetNamespace="http://openurc.org/TPL/basic-thermostat-1.0/"
  xmlns="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl-urc="http://openurc.org/ns/wsdl-urc" ...>
  ...
  <interface name="socket">
    ...
    <operation name="chk.confirmReset"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
      wsdlx:safe="true">
      <documentation>
        <wsdl-urc:category>alert</wsdl-urc:category>
        <wsdl-urc:sensitive>false</wsdl-urc:sensitive>
        <wsdl-urc:optional>false</wsdl-urc:optional>
      </documentation>
      <input messageLabel="In" element="wsdl-urc:sessionId" />
      <output messageLabel="Out" element="wsdl-urc:notifyStatus" />
      <outfault ref="sessionFault"/>
    </operation>
    ...
  </interface>
  ...
</description>
```

EXAMPLE 3 WSDL1 code snippet for a check operation for a notification with id="connectionError", of type "show", and of category "error". The notification has a default timeout of 10 seconds. It is not contained in a set. The socket is session-full.

```
    <operation name="chk.connectionError">
      <documentation>
        <wsdl-urc:category>error</wsdl-urc:category>
        <wsdl-urc:sensitive>false</wsdl-urc:sensitive>
        <wsdl-urc:optional>false</wsdl-urc:optional>
        <wsdl-urc:timeout>PT10S</wsdl-urc:timeout>
      </documentation>
      <input message="sessionId" />
      <output message="notifyStatus" />
      <fault name="sessionFault" message="sessionStatus"/>
    </operation>
```