
MediaRenderer:3 Device

For UPnP Version 1.0

Status: Standardized DCP (SDCP)

Date: March 31, 2013

Device Template Version 3.0

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP Forum, pursuant to Section 2.1(c)(ii) of the UPnP Forum Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP Compliant Devices. All such use is subject to all of the provisions of the UPnP Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

© 2014, UPnP Forum. All rights Reserved.

Authors	Company
Alan Presser	Allegrosoft
Wouter van der Beek	Cisco Systems
Gary Langille	Echostar
Gerrie Shults	HP
Raj Bopardikar	Intel
Nelson Kidd	Intel
John Ritchie	Intel
Mark Walker	Intel
Seung R. Yang	LG Electronics
Sungjoon Ahn	LG Electronics
Changhyun Kim	LG Electronics
Jack Unverferth	Microsoft
Keith Miller (Chair)	Nokia
Masatomo Hori	Panasonic
Matthew Ma	Panasonic
Russell Berkoff	Pioneer
Irene Shen	Pioneer
Wouter van der Beek	Philips
Wim Bronnenberg	Philips
Jeffrey Kang	Philips
Geert Knapen	Philips
Russell Berkoff (Vice-Chair)	Samsung Electronics
Richard Bardini	Sony
Norifumi Kikkawa	Sony
Jonathan Tourzan	Sony
Yasuhiro Morioka	Toshiba
Nicholas Frame	TP Vision
<p>Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.</p>	

CONTENTS

1	Scope	5
2	Normative references	5
3	Terms, definitions, symbols and abbreviations	8
3.1	Provisioning terms.....	8
3.2	Symbols	9
4	Notations and Conventions.....	9
4.1	Notation	9
4.1.1	Data Types.....	9
4.1.2	Strings Embedded in Other Strings.....	10
4.1.3	Extended Backus-Naur Form	10
4.2	Derived Data Types.....	11
4.2.1	Summary	11
4.2.2	CSV Lists	11
4.3	Management of XML Namespaces in Standardized DCPs	12
4.3.1	Namespace Prefix Requirements.....	16
4.3.2	Namespace Names, Namespace Versioning and Schema Versioning	17
4.3.3	Namespace Usage Examples	19
4.4	Vendor-defined Extensions.....	19
4.4.1	Vendor-defined Action Names	19
4.4.2	Vendor-defined State Variable Names	20
4.4.3	Vendor-defined XML Elements and attributes	20
4.4.4	Vendor-defined Property Names.....	20
5	Device Definitions	20
5.1	Device Type	20
5.2	Device Model	20
5.2.1	Description of Device Requirements	21
5.2.2	Relationships Between Services	21
6	XML Device Description	22
7	Test	23
	Annex A (informative) Theory of Operation.....	24
A.1	Device Discovery.....	24
A.2	Preparing to Transfer the Content.....	24
A.3	Controlling the Transfer of the Content	24
A.4	Controlling How the Content is Rendered	25
	Annex B (informative) Bibliography.....	26

List of Tables

Table 1 — EBNF Operators	11
Table 2 — CSV Examples	12
Table 3 — Namespace Definitions	13
Table 4 — Schema-related Information	15
Table 5 — Default Namespaces for the AV Specifications	17
Table 6 — Device Requirements	20

List of Figures

Figure 1 — MediaRenderer Functional Diagram	5
---	---

1 Scope

This device specification is compliant with the Universal Plug and Play Device Architecture version 1.0 [14]. It defines a device type referred to herein as MediaRenderer.

The MediaRenderer specification defines a general-purpose device template that can be used to instantiate any Consumer Electronics (CE) device that is capable of rendering AV content from the home network. It exposes a set of rendering controls in which a control point can control how the specified AV content is rendered. This includes controlling various rendering features such as brightness, contrast, volume, etc.

Example instances of a MediaRenderer include traditional devices such as TVs and stereo systems. Some more contemporary examples include digital devices such as MP3 players and Electronic Picture Frames (EPFs). Although most of these examples typically render one specific type of content (for example, a TV typically renders video content), a MediaRenderer is able to support a number of different data formats and transfer protocols. For example, a sophisticated implementation of a TV MediaRenderer could also support MP3 data so that its speakers could be used to play MP3 audio content.

The MediaRenderer device specification is very lightweight and is easy to implement on low-resource devices such as an MP3 player. However, it can also be used to expose the high-end capabilities of devices such as a PC.

A full-featured MediaRenderer exposes the following capabilities:

- Control various rendering characteristics
- Expose the supported transfer protocols and data formats
- Control the flow of the content (for example, FF, REW, etc), if appropriate depending on the transfer protocol.

The MediaRenderer DOES not enable control points to:

- Send AV content to another device

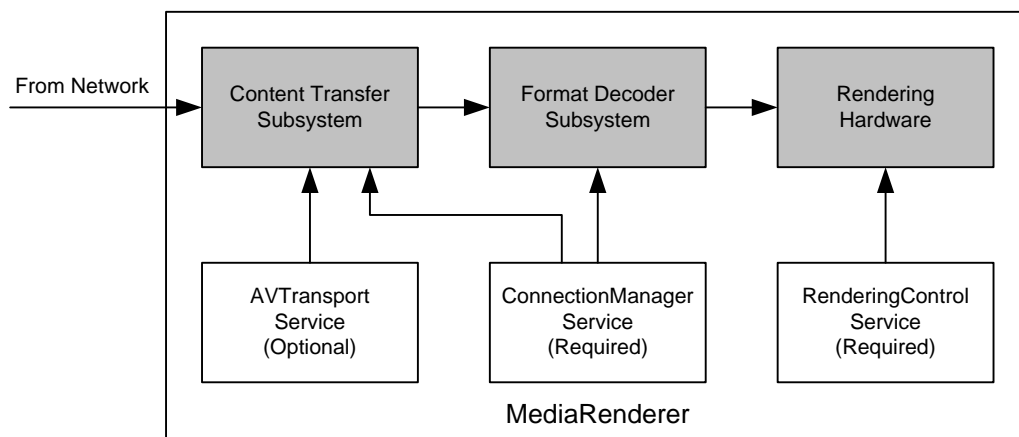


Figure 1 — MediaRenderer Functional Diagram

The un-shaded blocks represent the UPnP services that are contained by a MediaRenderer. The shaded blocks represent various device-specific modules that the UPnP services might interact with. However, the internal architecture of a MediaRenderer device is vendor specific.

2 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.

[1] – *XML Schema for RenderingControl AllowedTransformSettings*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/AllowedTransformSettings-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/AllowedTransformSettings.xsd>.

[2] – *AV Datastructure Template:1*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1-20130331.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1.pdf>.

[3] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/av-v3-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/av.xsd>.

[4] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/avs-v3-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/avs.xsd>.

[5] – *AVTransport:3*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVTransport-v3-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-AVTransport-v3-Service.pdf>.

[6] – *XML Schema for AVTransport LastChange Eventing*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/avt-event-v2-20080930.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/avt-event.xsd>.

[7] – *ContentDirectory:4*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf>.

[8] – *XML Schema for ContentDirectory LastChange Eventing*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/cds-event-v1-20080930.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/cds-event.xsd>.

[9] – *ConnectionManager:3*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service.pdf>.

[10] – *XML Schema for ConnectionManager DeviceClockInfoUpdates*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/cm-deviceClockInfoUpdates-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/cm-deviceClockInfoUpdates.xsd>.

[11] – *XML Schema for ConnectionManager Features*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/cm-featureList-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/cm-featureList.xsd>.

[12] – *XML Schema for UPnP AV Dublin Core*.

Available at: <http://www.dublincore.org/schemas/xmls/simpledc20020312.xsd>.

[13] – *DCMI term declarations represented in XML schema language*.

Available at: <http://www.dublincore.org/schemas/xmls>.

- [14] – *UPnP Device Architecture, version 1.0*, UPnP Forum, October 15, 2008.
Available at: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf>.
Latest version available at: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf>.
- [15] – *XML Schema for ContentDirectory Structure and Metadata (DIDL-Lite)*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/didl-lite-v3-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/didl-lite.xsd>.
- [16] – *XML Schema for ContentDirectory DeviceMode*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/dmo-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/dmo.xsd>.
- [17] – *XML Schema for ContentDirectory DeviceModeRequest*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/dmor-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/dmor.xsd>.
- [18] – *XML Schema for ContentDirectory DeviceModeStatus*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/dmos-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/dmos.xsd>.
- [19] – ISO/IEC 14977, *Information technology - Syntactic metalanguage - Extended BNF*, December 1996.
- [20] – *XML Schema for ContentDirectory PermissionsInfo*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/pi-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/pi.xsd>.
- [21] – *RenderingControl:3*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service.pdf>.
- [22] – *XML Schema for RenderingControl LastChange Eventing*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/rcs-event-v3-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rcs-event.xsd>.
- [23] – *XML Schema for ConnectionManager RendererInfo*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/rii-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rii.xsd>.
- [24] – *XML Schema for AVTransport PlaylistInfo*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/rpl-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rpl.xsd>.
- [25] – *ScheduledRecording:2*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service.pdf>.
- [26] – *XML Schema for ScheduledRecording Metadata and Structure*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/srs-v2-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/srs.xsd>.

[27] – *XML Schema for ScheduledRecording LastChange Eventing*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/srs-event-v1-20080930.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/srs-event.xsd>.

[28] – *XML Schema for RenderingControl TransformSettings*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/TransformSettings-v1-20130331.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/TransformSettings.xsd>.

[29] – *XML Schema for ContentDirectory Metadata*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/upnp-v4-20130331.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/upnp.xsd>.

[30] – *The “xml:” Namespace*, November 3, 2004.

Available at: <http://www.w3.org/XML/1998/namespace>.

[31] – *XML Schema for the “xml:” Namespace*.

Available at: <http://www.w3.org/2001/xml.xsd>.

[32] – *Namespaces in XML*, Tim Bray, Dave Hollander, Andrew Layman, eds., W3C Recommendation, January 14, 1999.

Available at: <http://www.w3.org/TR/1999/REC-xml-names-19990114>.

[33] – *XML Schema Part 1: Structures, Second Edition*, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation, 28 October 2004.

Available at: <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[34] – *XML Schema Part 2: Data Types, Second Edition*, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004.

Available at: <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

[35] – *XML Schema for XML Schema*.

Available at: <http://www.w3.org/2001/XMLSchema.xsd>.

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the terms and definitions given in [14] and the following subclauses 3.1 and 3.2 apply.

3.1 Provisioning terms

3.1.1

allowed

A

The definition or behavior is allowed.

3.1.2

conditionally allowed

CA

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

3.1.3

conditionally required

CR

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required. Otherwise the definition or behavior is allowed as default unless specifically defined as not allowed.

3.1.4 required

R

The definition or behavior is required.

3.1.5 R/A

Used in a table column heading to indicate that each abbreviated entry in the column declares the provisioning status of the item named in the entry's row.

3.1.6 **X**

Vendor-defined, non-standard.

3.1.7 **-D**

Declares that the item referred to is deprecated, when it is appended to any of the other abbreviated provisioning terms.

3.1.8 CSV list (or CSV)

Comma separated value list. List—or one-dimensional array—of values contained in a string and separated by commas

3.2 Symbols

3.2.1

::

Signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

4 Notations and Conventions

4.1 Notation

- UPnP interface names defined in the UPnP Device Architecture specification [14] are styled in **green bold underlined** text.
- UPnP interface names defined outside of the UPnP Device Architecture specification [14] are styled in **red italic underlined** text.
- Some additional non-interface names and terms are styled in *italic* text.
- Words that are emphasized are also styled in *italic* text. The difference between italic terms and italics for emphasis will be apparent by context.
- Strings that are to be taken literally are enclosed in “double quotes”.

4.1.1 Data Types

Data type definitions come from three sources:

- All state variable and action argument data types are defined in [14].
- Basic data types for properties are defined in [34].
- Additional data types for properties are defined in the XML schema(s) (see [3]) associated with this service.

For UPnP Device Architecture defined **boolean** data types, it is strongly recommended to use the value “**0**” for false, and the value “**1**” for true. However, when used as input arguments, the values “**false**”, “**no**”, “**true**”, “**yes**” may also be encountered and shall be accepted. Nevertheless, it is strongly recommended that all **boolean** state variables and output arguments be represented as “**0**” and “**1**”.

For XML Schema defined Boolean data types, it is strongly recommended to use the value “0” for false, and the value “1” for true. However, when used as input properties, the values “*false*”, “*true*” may also be encountered and shall be accepted. Nevertheless, it is strongly recommended that all Boolean properties be represented as “0” and “1”.

4.1.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that shall be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary characters in the string and/or its independent substrings. This document uses embedded strings in two contexts – Comma Separated Value (CSV) lists (see subclause 4.2.2) and property values in search criteria strings. Escaping conventions use the backslash character, “\” (character code U+005C), as follows:

- a) Backslash (“\”) is represented as “\\” in both contexts.
- b) Comma (“,”) is
 - 1) represented as “\,” in individual substring entries in CSV lists
 - 2) not escaped in search strings
- c) Double quote (“””) is
 - 1) not escaped in CSV lists
 - 2) not escaped in search strings when it appears as the start or end delimiter of a property value
 - 3) represented as “\”” in search strings when it appears as a character that is part of the property value

4.1.3 Extended Backus-Naur Form

Extended Backus-Naur Form is used in this document for a formal syntax description of certain constructs. The usage here is according to the reference [19].

4.1.3.1 Typographic conventions for EBNF

Non-terminal symbols are unquoted sequences of characters from the set of English upper and lower case letters, the digits “0” through “9”, and the hyphen (“-”). Character sequences between 'single quotes' are terminal strings and shall appear literally in valid strings. Character sequences between (*comment delimiters*) are English language definitions or supplementary explanations of their associated symbols. White space in the EBNF is used to separate elements of the EBNF, not to represent white space in valid strings. White space usage in valid strings is described explicitly in the EBNF. Finally, the EBNF uses the following operators in Table 1:

Table 1 — EBNF Operators

Operator	Semantics
::=	definition – the non-terminal symbol on the left is defined by one or more alternative sequences of terminals and/or non-terminals to its right.
	alternative separator – separates sequences on the right that are independently allowed definitions for the non-terminal on the left.
*	null repetition – means the expression to its left may occur zero or more times.
+	non-null repetition – means the expression to its left shall occur at least once and may occur more times.
[]	optional – the expression between the brackets is allowed.
()	grouping – groups the expressions between the parentheses.
-	character range – represents all characters between the left and right character operands inclusively.

4.2 Derived Data Types

4.2.1 Summary

Subclause 4.2 defines a derived data type that is represented as a string data type with special syntax. This specification uses string data type definitions that originate from two different sources. The UPnP Device Architecture defined [string](#) data type is used to define state variable and action argument [string](#) data types. The XML Schema namespace is used to define property xsd:string data types. The following definition in subclause 4.2.2 applies to both string data types.

4.2.2 CSV Lists

The UPnP AV services use state variables, action arguments and properties that represent lists – or one-dimensional arrays – of values. The UPnP Device Architecture, Version 1.0 [14], does not provide for either an array type or a list type, so a list type is defined here. Lists may either be homogeneous (all values are the same type) or heterogeneous (all values can be of different types). Lists may also consist of repeated occurrences of homogeneous or heterogeneous subsequences, all of which have the same syntax and semantics (same number of values, same value types and in the same order). The data type of a homogeneous list is [string](#) or xsd:string and denoted by CSV (*x*), where *x* is the type of the individual values. The data type of a heterogeneous list is also [string](#) or xsd:string and denoted by CSV (*x*, *y*, *z*), where *x*, *y* and *z* are the types of the individual values. If the number of values in the heterogeneous list is too large to show each type individually, that variable type is represented as CSV (heterogeneous), and the variable description includes additional information as to the expected sequence of values appearing in the list and their corresponding types. The data type of a repeated subsequence list is [string](#) or xsd:string and denoted by CSV ({*a,b,c*},{*x*, *y*, *z*}), where *a*, *b*, *c*, *x*, *y* and *z* are the types of the individual values in the subsequence and the subsequences may be repeated zero or more times.

- A list is represented as a [string](#) type (for state variables and action arguments) or xsd:string type (for properties).
- Commas separate values within a list.
- Integer values are represented in CSVs with the same syntax as the integer data type specified in [14] (that is: allowed leading sign, allowed leading zeroes, numeric US-ASCII)
- Boolean values are represented in state variable and action argument CSVs as either “[0](#)” for false or “[1](#)” for true. These values are a subset of the defined [boolean](#) data type values specified in [14]: [0](#), [false](#), [no](#), [1](#), [true](#), [yes](#).
- Boolean values are represented in property CSVs as either “[0](#)” for false or “[1](#)” for true. These values are a subset of the defined Boolean data type values specified in [34]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in 4.1.2.

- White space before, after, or interior to any numeric data type is not allowed.
- White space before, after, or interior to any other data type is part of the value.

Table 2 — CSV Examples

Type refinement of string	Value	Comments
CSV (string) or CSV (xsd:string)	"+artist,-date"	List of 2 property sort criteria.
CSV (int) or CSV (xsd:integer)	"1,-5,006,0,+7"	List of 5 integers.
CSV (boolean) or CSV (xsd:Boolean)	"0,1,1,0"	List of 4 booleans
CSV (string) or CSV (xsd:string)	"Smith\, Fred,Jones\, Davey"	List of 2 names, "Smith, Fred" and "Jones, Davey"
CSV (i4 , string , ui2) or CSV (xsd:int, xsd:string, xsd:unsignedShort)	"-29837, string with leading blanks,0"	Note that the second value is " string with leading blanks"
CSV (i4) or CSV (xsd:int)	"3, 4"	Illegal CSV. White space is not allowed as part of an integer value.
CSV (string) or CSV (xsd:string)	" , ,"	List of 3 empty string values
CSV (heterogeneous)	"Alice,Marketing,5,Sue,R&D,21,Dave,Finance,7"	List of unspecified number of people and associated attributes. Each person is described by 3 elements: a name string , a department string and years-of-service ui2 or a name xsd:string, a department xsd:string and years-of-service xsd:unsignedShort.

4.3 Management of XML Namespaces in Standardized DCPs

UPnP specifications make extensive use of XML namespaces. This enables separate DCPs, and even separate components of an individual DCP, to be designed independently and still avoid name collisions when they share XML documents. Every name in an XML document belongs to exactly one namespace. In documents, XML names appear in one of two forms: qualified or unqualified. An unqualified name (or no-colon-name) contains no colon (":") characters. An unqualified name belongs to the document's default namespace. A qualified name is two no-colon-names separated by one colon character. The no-colon-name before the colon is the qualified name's namespace prefix, the no-colon-name after the colon is the qualified name's "local" name (meaning local to the namespace identified by the namespace prefix). Similarly, the unqualified name is a local name in the default namespace.

The formal name of a namespace is a URI. The namespace prefix used in an XML document is *not* the name of the namespace. The namespace name shall be globally unique. It has a single definition that is accessible to anyone who uses the namespace. It has the same meaning anywhere that it is used, both inside and outside XML documents. The namespace prefix, however, in formal XML usage, is defined only in an XML document. It shall be locally unique to the document. Any valid XML no-colon-name may be used. And, in formal XML usage, different XML documents may use different namespace prefixes to refer to the same namespace. The creation and use of the namespace prefix was standardized by the W3C XML Committee in [32] strictly as a convenient local shorthand replacement for the full URI name of a namespace in individual documents.

All AV object properties are represented in XML by element and attribute names, therefore, all property names belong to an XML namespace.

For the same reason that namespace prefixes are convenient in XML documents, it is convenient in specification text to refer to namespaces using a namespace prefix. Therefore, this specification declares a “standard” prefix for all XML namespaces used herein. In addition, this specification expands the scope where these prefixes have meaning, beyond a single XML document, to all of its text, XML examples, and certain string-valued properties. This expansion of scope *does not* supersede XML rules for usage in documents, it only augments and complements them in important contexts that are out-of-scope for the XML specifications. For example, action arguments which refer to CDS properties, such as the [SearchCriteria](#) argument of the [Search\(\)](#) action or the [Filter](#) argument of the [Browse\(\)](#) action, shall use the predefined namespace prefixes when referring to CDS properties (“upnp:”, “dc:”, etc).

All of the namespaces used in this specification are listed in Table 3 and Table 4. For each such namespace, Table 3 gives a brief description of it, its name (a URI) and its defined “standard” prefix name. Some namespaces included in these tables are not directly used or referenced in this document. They are included for completeness to accommodate those situations where this specification is used in conjunction with other UPnP specifications to construct a complete system of devices and services. For example, since the ScheduledRecording service depends on and refers to the ContentDirectory service, the predefined “srs:” namespace prefix is included. The individual specifications in such collections all use the same standard prefix. The standard prefixes are also used in Table 4 to cross-reference additional namespace information. Table 4 includes each namespace’s valid XML document root element(s) (if any), its schema file name, versioning information (to be discussed in more detail below), and a link to the entry in Clause 2 for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 3. The schemas are designed to support these definitions for both human understanding and as test tools. However, limitations of the XML Schema language itself make it difficult for the UPnP-defined schemas to accurately represent all details of the namespace definitions. As a result, the schemas will validate many XML documents that are not valid according to the specifications.

The Working Committee expects to continue refining these schemas after specification release to reduce the number of documents that are validated by the schemas while violating the specifications, but the schemas will still be informative, supporting documents. Some schemas might become normative in future versions of the specifications.

Table 3 — Namespace Definitions

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
<i>AV Working Committee defined namespaces</i>			
attrs	urn:schemas-upnp-org:av:AllowedTransformSettings	AllowedTransformSettings and AllowedDefaultTransformSettings state variables for RenderingControl	[21]
av	urn:schemas-upnp-org:av:av	Common data types for use in AV schemas	[3]
avdt	urn:schemas-upnp-org:av:avdt	Datastructure Template	[2]
avs	urn:schemas-upnp-org:av:avs	Common structures for use in AV schemas	[4]
avt-event	urn:schemas-upnp-org:metadata-1-0/AVT/	Evented LastChange state variable for AVTransport	[5]
cds-event	urn:schemas-upnp-org:av:cds-event	Evented LastChange state variable for ContentDirectory	[7]

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
cm-dciu	urn:schemas-upnp-org:av:cm-deviceClockInfoUpdates	Evented DeviceClockInfoUpdates state variable for ConnectionManager	[9]
cm-ftrlist	urn:schemas-upnp-org:av:cm-featureList	FeatureList state variable for ConnectionManager	[9]
didl-lite	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[7]
dmo	urn:schemas-upnp.org:av:dmo	Evented DeviceMode state variable for ContentDirectory	[7]
dmor	urn:schemas-upnp.org:av:dmor	A_ARG_TYPE_DeviceModeRequest state variable for ContentDirectory	[7]
dmos	urn:schemas-upnp.org:av:dmos	DeviceModeStatus state variable for ContentDirectory	[7]
pi	urn:schemas-upnp.org:av:pi	PermissionsInfo state variable for ContentDirectory	[7]
rcs-event	urn:schemas-upnp-org:metadata-1-0/RCS/	Evented LastChange state variable for RenderingControl	[21]
rii	urn:schemas-upnp-org:av:rii	A_ARG_TYPE_RenderingInfoList state variable for ConnectionManager	[9]
rpl	urn:schemas-upnp-org:av:rpl	A_ARG_TYPE_PlaylistInfo state variable for AVTransport	[5]
srs	urn:schemas-upnp-org:av:srs	Metadata and structure for ScheduledRecording	[25]
srs-event	urn:schemas-upnp-org:av:srs-event	Evented LastChange state variable for ScheduledRecording	[25]
trs	urn:schemas-upnp-org:av:TransformSettings	TransformSettings and DefaultTransformSettings state variables for RenderingControl	[21]
upnp	urn:schemas-upnp-org:metadata-1-0/upnp/	Metadata for ContentDirectory	[7]
<i>Externally defined namespaces</i>			
dc	http://purl.org/dc/elements/1.1/	Dublin Core	[13]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[33], [34]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	[33] 2.6 & 3.2.7
xml	http://www.w3.org/XML/1998/namespace	The "xml:" Namespace	[30]

Table 4 — Schema-related Information

Standard Name-space Prefix	Relative URI and File Name ^a • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
<i>AV Working Committee Defined Namespaces</i>			
atrs	AllowedTransformSetting s-vn-yyyyymmdd.xsd AllowedTransformSetting s-vn.xsd AllowedTransformSetting s.xsd	<TransformList>	[1]
av	av-vn-yyyyymmdd.xsd av-vn.xsd av.xsd	n/a	[3]
avdt	avdt-vn-yyyyymmdd.xsd avdt-vn.xsd avdt.xsd	<AVDT>	[2]
avs	avs-vn-yyyyymmdd.xsd avs-vn.xsd avs.xsd	<Capabilities> <Features> <stateVariableValuePairs>	[4]
avt-event	avt-event-vn- yyyyymmdd.xsd avt-event-vn.xsd avt-event.xsd	<Event>	[6]
cds-event	cds-event-vn- yyyyymmdd.xsd cds-event-vn.xsd cds-event.xsd	<StateEvent>	[8]
cm-dciu	cm- deviceClockInfoUpdates- vn-yyyyymmdd.xsd cm- deviceClockInfoUpdates -vn.xsd cm- deviceClockInfoUpdates. xsd	<DeviceClockInfoUpdates>	[10]
cm-ftrlst	cm-featureList-vn- yyyyymmdd.xsd cm-featureList-vn.xsd cm-featureList.xsd	<Features>	[11]
didl-lite	didl-lite-vn- yyyyymmdd.xsd didl-lite-vn.xsd didl-lite.xsd	<DIDL-Lite>	[15]
dmo	dmo-vn-yyyyymmdd.xsd dmo-vn.xsd dmo.xsd	<DeviceMode>	[16]
dmor	dmor-vn-yyyyymmdd.xsd dmor-vn.xsd dmor.xsd	<DeviceModeRequest>	[17]
dmos	dmos-vn-yyyyymmdd.xsd dmos-vn.xsd dmos.xsd	<DeviceModeStatus>	[18]

Standard Name-space Prefix	Relative URI and File Name ^a • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
pi	pi-vn-yyyymmdd.xsd pi-vn.xsd pi.xsd	<PermissionsInfo>	[20]
rcs-event	rcs-event-vn-yyyymmdd.xsd rcs-event-vn.xsd rcs-event.xsd	<Event>	[22]
rii	rii-vn-yyyymmdd.xsd rii-vn.xsd rii.xsd	<rendererInfo>	[23]
rpl	rpl-vn-yyyymmdd.xsd rpl-vn.xsd rpl.xsd	<PlaylistInfo>	[24]
trs	TransformSettings-vn-yyyymmdd.xsd TransformSettings-vn.xsd TransformSettings.xsd	<TransformSettings>	[28]
srs	srs-vn-yyyymmdd.xsd srs-vn.xsd srs.xsd	<srs>	[26]
srs-event	srs-event-vn-yyyymmdd.xsd srs-event-vn.xsd srs-event.xsd	<StateEvent>	[27]
upnp	upnp-vn-yyyymmdd.xsd upnp-vn.xsd upnp.xsd	n/a	[29]
<i>Externally Defined Namespaces</i>			
dc	<i>Absolute URL:</i> http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[12]
xsd	n/a	<schema>	[35]
xsi	n/a		n/a
xml	n/a		[31]
^a Absolute URIs are generated by prefixing the relative URIs with " http://www.upnp.org/schemas/av/ "			

4.3.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings shall use the standard namespace prefixes as declared in Table 3. In order to properly process the XML documents described herein, control points and devices shall use namespace-aware XML processors [32] for both reading and writing. As allowed by [32], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document may be different from the standard prefix. All devices shall be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. However, it is strongly recommended that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP specifications. However, each

individual specification may assume a default namespace for its descriptive text. In that case, names from that namespace may appear with no prefix.

The assumed default namespace, if any, for each UPnP AV specification is given in Table 5.

Note: all UPnP AV schemas declare attributes to be “unqualified”, so namespace prefixes are never used with AV Working Committee defined attribute names.

Table 5 — Default Namespaces for the AV Specifications

AV Specification Name	Default Namespace Prefix
AVTransport	avt-event
ConnectionManager	n/a
ContentDirectory	didl-lite
MediaRenderer	n/a
MediaServer	n/a
RenderingControl	rcs-event
ScheduledRecording	srs

4.3.2 Namespace Names, Namespace Versioning and Schema Versioning

The UPnP AV service specifications define several data structures (such as state variables and action arguments) whose format is an XML instance document that complies with one or more specific XML schemas, which define XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespace names that are defined by the AV Working Committee are URNs. See Table 3 for a current list of namespace names. Additionally, each namespace corresponds to an XML schema document that provides a machine-readable representation of the associated namespace to enable automated validation of the XML (state variable or action parameter) instance documents.

Within an XML schema and XML instance document, the name of each corresponding namespace appears as the value of an `xmlns` attribute within the root element. Each `xmlns` attribute also includes a namespace prefix that is associated with that namespace in order to qualify and disambiguate element and attribute names that are defined within different namespaces. The schemas that correspond to the listed namespaces are identified by URI values that are listed in the `schemaLocation` attribute also within the root element (see subclause 4.3.3).

In order to enable both forward and backward compatibility, namespace names are permanently assigned and shall not change even when a new version of a specification changes the definition of a namespace. However, all changes to a namespace definition shall be backward-compatible. In other words, the updated definition of a namespace shall not invalidate any XML documents that comply with an earlier definition of that same namespace. This means, for example, that a namespace shall not be changed so that a new element or attribute becomes required in a conforming instance document. Although namespace names shall not change, namespaces still have version numbers that reflect a specific set of definitional changes. Each time the definition of a namespace is changed, the namespace's version number is incremented by one.

Whenever a new namespace version is created, a new XML schema document (.xsd) is created and published so that the new namespace definition is represented in a machine-readable form. Since a XML schema document is just a representation of a namespace definition, translation errors can occur. Therefore, it is sometime necessary to re-release a published schema in order to correct typos or other namespace representation errors. In order to easily identify the potential multiplicity of schema releases for the same namespace, the URI of each released schema shall conform to the following format (called Form 1):

Form 1: "http://www.upnp.org/schemas/av/" **schema-root-name** "-v" **ver** "-" **yyyymmdd** where

- **schema-root-name** is the name of the root element of the namespace that this schema represents.
- **ver** corresponds to the version number of the namespace that is represented by the schema.
- **yyyymmdd** is the year, month and day (in the Gregorian calendar) that this schema was released.

Table 4 identifies the URI formats for each of the namespaces that are currently defined by the UPnP AV Working Committee.

As an example, the original schema URI for the “rcs-event” namespace (that was released with the original publication of the UPnP AV service specifications in the year 2002) was “<http://www.upnp.org/schemas/av/rcs-event-v1-20020625.xsd>”. When the UPnP AV service specifications were subsequently updated in the year 2006, the URI for the updated version of the “rcs-event” namespace was “<http://www.upnp.org/schemas/av/rcs-event-v2-20060531.xsd>”. However, in 2006, the schema URI for the newly created “srs-event” namespace was “<http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd>”. Note the version field for the “srs-event” schema is “v1” since it was first version of that namespace whereas the version field for the “rcs-event” schema is “v2” since it was the second version of that namespace.

In addition to the dated schema URIs that are associated with each namespace, each namespace also has a set of undated schema URIs. These undated schema URIs have two distinct formats with slightly different meanings:

Form 2: “<http://www.upnp.org/schemas/av/>” *schema-root-name* “-v” **ver**
where **ver** is described above.

Form 3: “<http://www.upnp.org/schemas/av/>” *schema-root-name*

Form 2 of the undated schema URI is always linked to the most recent release of the schema that represents the version of the namespace indicated by **ver**. For example, the undated URI “[.../av/rcs-event-v2.xsd](http://www.upnp.org/schemas/av/rcs-event-v2.xsd)” is linked to the most recent schema release of version 2 of the “rcs-event” namespace. Therefore, on May 31, 2006 (20060531), the undated schema URI was linked to the schema that is otherwise known as “[.../av/rcs-event-v2-20060531.xsd](http://www.upnp.org/schemas/av/rcs-event-v2-20060531.xsd)”. Furthermore, if the schema for version 2 of the “rcs-event” namespace was ever re-released, for example to fix a typo in the 20060531 schema, then the same undated schema URI (“[.../av/rcs-event-v2.xsd](http://www.upnp.org/schemas/av/rcs-event-v2.xsd)”) would automatically be updated to link to the updated version 2 schema for the “rcs-event” namespace.

Form 3 of the undated schema URI is always linked to the most recent release of the schema that represents the highest version of the namespace that has been published. For example, on June 25, 2002 (20020625), the undated schema URI “[.../av/rcs-event.xsd](http://www.upnp.org/schemas/av/rcs-event.xsd)” was linked to the schema that is otherwise known as “[.../av/rcs-event-v1-20020625.xsd](http://www.upnp.org/schemas/av/rcs-event-v1-20020625.xsd)”. However, on May 31, 2006 (20060531), that same undated schema URI was linked to the schema that is otherwise known as “[.../av/rcs-event-v2-20060531.xsd](http://www.upnp.org/schemas/av/rcs-event-v2-20060531.xsd)”.

When referencing a schema URI within an XML instance document or a referencing XML schema document, the following usage rules apply:

- All instance documents, whether generated by a service or a control point, shall use Form 3.
- All UPnP AV published schemas that reference other UPnP AV schemas shall also use Form 3.

Within an XML instance document, the definition for the `schemaLocation` attribute comes from the XML Schema namespace “<http://www.w3.org/2002/XMLSchema-instance>”. A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values that is interpreted as a namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

In addition to the schema URI naming and usage rules described above, each released schema shall contain a `version` attribute in the `<schema>` root element. Its value shall correspond to the format:

ver “-” **yyyymmdd** where **ver** and **yyyymmdd** are described above.

The `version` attribute provides self-identification of the namespace version and release date of the schema itself. For example, within the original schema released for the “rcs-event” namespace (`.../rcs-event-v2-20020625.xsd`), the `<schema>` root element contains the following attribute: `version="2-20020625"`.

4.3.3 Namespace Usage Examples

The `schemaLocation` attribute for XML instance documents comes from the XML Schema instance namespace “`http://www.w3.org/2002/XMLSchema-instance`”. A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values: namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

Example 1:

Sample *DIDL-Lite XML Instance Document*. Note that the references to the UPnP AV schemas do not contain any version or release date information. In other words, the references follow Form 3 from above. Consequently, this example is valid for all releases of the UPnP AV service specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
  xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
      http://www.upnp.org/schemas/av/didl-lite.xsd
    urn:schemas-upnp-org:metadata-1-0/upnp/
      http://www.upnp.org/schemas/av/upnp.xsd">
  <item id="18" parentID="13" restricted="0">
    ...
  </item>
</DIDL-Lite>
```

4.4 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified below in subclauses 4.4.1 to 4.4.4.

4.4.1 Vendor-defined Action Names

Vendor-defined action names shall begin with “**X**”. Additionally, it should be followed by an ICANN assigned domain name owned by the vendor followed by the underscore character (“_”). It shall then be followed by the vendor-assigned action name. The vendor-assigned action name shall not contain a hyphen character (“-”, 2D Hex in UTF-8) nor a hash character (“#”, 23 Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters shall not be “XML” in any combination of case.

4.4.2 Vendor-defined State Variable Names

Vendor-defined state variable names shall begin with “**X**”. Additionally, it should be followed by an ICANN assigned domain name owned by the vendor, followed by the underscore character (“_”). It shall then be followed by the vendor-assigned state variable name. The vendor-assigned state variable name shall not contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters shall not be “XML” in any combination of case.

4.4.3 Vendor-defined XML Elements and attributes

UPnP vendors may add non-standard elements and attributes to a UPnP standard XML document, such as a device or service description. Each addition shall be scoped by a vendor-owned XML namespace. Arbitrary XML shall be enclosed in an element that begins with “**X**,” and this element shall be a sub element of a standard complex type. Non-standard attributes may be added to standard elements provided these attributes are scoped by a vendor-owned XML namespace and begin with “**X**”.

4.4.4 Vendor-defined Property Names

UPnP vendors may add non-standard properties to the ContentDirectory service. Each property addition shall be scoped by a vendor-owned namespace. The vendor-assigned property name shall not contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned property names are case sensitive. The first character of the name shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters shall be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters shall not be “XML” in any combination of case.

5 Device Definitions

5.1 Device Type

The following device type identifies a device that is compliant with this specification:

urn:[schemas-upnp-org:device:MediaRenderer:3](#)

The shorthand MediaRenderer is used herein to refer to this device type.

5.2 Device Model

MediaRenderer products shall implement minimum version numbers of all required embedded devices and services specified in Table 6 below. A MediaRenderer device can be either a [Root](#) device or can be [Embedded](#) in another UPnP device (MediaRenderer or other). A MediaRenderer device ([Root](#) or [Embedded](#)) can in turn contain other standard or non-standard [Embedded](#) UPnP devices.

Table 6 — Device Requirements

DeviceType	Root	R/A ^a	ServiceType	R/A	Service ID ^b
MediaRenderer:3	Root or Embedded	R	RenderingControl:3	R	RenderingControl
			ConnectionManager:3	R	ConnectionManager
			AVTransport:3	A	AVTransport
			<i>Standard non-AV services defined by UPnP (QoS, Security, etc.) go here.</i>	X	TBD

			Non-standard services embedded by a UPnP vendor go here.	X	TBD
Standard devices embedded by a UPnP vendor go here.	Embedded	A	Services as defined by the corresponding standard UPnP Device Definition go here.		
Non-standard devices embedded by a UPnP vendor go here.	Embedded	X	TBD	TBD	TBD
<p>^a R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).</p> <p>^b Prefixed by urn:upnp-org:serviceId:</p>					

5.2.1 Description of Device Requirements

Any instance of a MediaRenderer shall have a RenderingControl service and a ConnectionManager service. For a given instance (MediaRenderer), there shall only be one instance of these standard defined services. There may be one instance of a standard AVTransport service. The semantics of additional AV services are not defined. Other standard services, such as UPnP QoS, may be added with semantics defined by the relevant specifications.

It should be noted that MediaRenderer:3 implementations shall respond to all SSDP queries that specify MediaRenderer:2 or lower and shall respond to all actions defined by the MediaRenderer:2 specification or lower.

The RenderingControl service allows control points to control the various rendering capabilities of the device. The ConnectionManager service is used to enumerate and select a particular transfer protocol and data format to be used for transferring the content. Additionally, the ConnectionManager service also allows control points, such as a home network management application, to discover useful information about the content transfers that the device is actively participating in. Such information could be useful to a Quality of Service capability, which may be defined in the future.

The existence of the AVTransport service depends on the transfer protocols that are supported by the device. The ConnectionManager service specification includes a table that identifies which transfer protocols require an AVTransport service to be implemented on the MediaRenderer. If an implementation of the MediaRenderer supports any of these transfer protocols, then it shall implement the AVTransport service. However, no AVTransport service instances will be instantiated until a connection is made using one of those transfer protocols.

5.2.2 Relationships Between Services

The [ConnectionManager::PrepareForConnection\(\)](#) action provides the trigger point for creating a new virtual instance of the RenderingControl and AVTransport service (refer to the RenderingControl and AVTransport service specifications for a description of virtual instances of those services). When a new connection is established (one that requires an AVTransport service on the MediaRenderer, which is determined by the selected transfer protocol), the [ConnectionManager::PrepareForConnection\(\)](#) action returns the [InstanceID](#) of the RenderingControl and AVTransport services that are bound to that connection. The RenderingControl service virtual instance is used by the control point to control how the content from that connection is rendered. The AVTransport service virtual instance is used by the control point to control the flow (for example, [AVTransport::Play\(\)](#), [AVTransport::Seek\(\)](#), etc.) of the content received via that connection. As described in the RenderingControl and AVTransport service specifications, each virtual instance of these services operates independently from all other virtual instances.

6 XML Device Description

```

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>
      urn:schemas-upnp-org:device:MediaRenderer:3
    </deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>
          urn:schemas-upnp-org:service:RenderingControl:3
        </serviceType>
        <serviceId>
          urn:upnp-org:serviceId:RenderingControl
        </serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>
          urn:schemas-upnp-org:service:ConnectionManager:3
        </serviceType>
        <serviceId>
          urn:upnp-org:serviceId:ConnectionManager
        </serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>
          urn:schemas-upnp-org:service:AVTransport:3
        </serviceType>
        <serviceId>

```

```
urn:upnp-org:serviceId:AVTransport
</serviceId>
<SCPDUURL>URL to service description</SCPDUURL>
<controlURL>URL for control</controlURL>
<eventSubURL>URL for eventing</eventSubURL>
</service>
Declarations for standard non-AV services defined by UPnP
(if any) go here
Declarations for other services added by UPnP vendor
(if any) go here
</serviceList>
<deviceList>
Description of embedded devices added by UPnP vendor
(if any) go here
</deviceList>
<presentationURL>URL for presentation</presentationURL>
</device>
</root>
```

7 Test

There are no semantic tests defined for this device.

Annex A (informative)

Theory of Operation

MediaRenderer devices are used in conjunction with one or more MediaServer device(s) to enable a control point to render entertainment (AV) content (for example, video, music, images, etc.) that is discovered on a MediaServer device within the home network. In general terms, the process begins with the control point(s) discovering MediaServer and MediaRenderer devices within the home network. After a control point locates the desired content on a MediaServer, the control point needs to identify a common transfer protocol and data format that can be used to transfer the content from the MediaServer to the MediaRenderer. After these transfer parameters have been established, the control point controls the flow of the content (for example, [AVTransport::Play\(\)](#), [AVTransport::Pause\(\)](#), [AVTransport::Stop\(\)](#), [AVTransport::Seek\(\)](#), etc.). (Depending on the selected transfer protocol, these flow control operations are sent either to the MediaServer or the MediaRenderer, but not both). The actual transfer of the content is performed directly by the MediaServer and MediaRenderer. The content transfer happens independently from the control point and does not involve UPnP itself. The control point uses UPnP to setup the transfer of the content, but the transfer is performed using an out-of band transfer protocol.

A.1 Device Discovery

Control points can discover MediaRenderer devices using the standard UPnP SSDP-based device discovery mechanism to search for any device that is a member of the MediaRenderer device class including [Root](#) devices and/or [Embedded](#) devices.

A.2 Preparing to Transfer the Content

After the desired content has been identified, the control point needs to determine which transfer protocol and data format need to be used to transfer the content from the MediaServer to the MediaRenderer. (Transfer protocol examples include IEEE-1394, HTTP GET, RTSP/RTP, etc., and data format examples include MPEG2, MPEG4, MP3, WMA, JPEG, etc.) The control point makes this determination by comparing the content's protocol/format information (obtained via the MediaServer's ContentDirectory service) with the protocol/format information obtained via the MediaRenderer's [ConnectionManager::GetProtocolInfo\(\)](#) action.

After the transfer protocol and data format have been identified, the control point uses the [ConnectionManager::PrepareForConnection\(\)](#) action on each device to inform the device that the specified protocol/format are about to be used. Depending on which transfer protocol was selected, the [ConnectionManager::PrepareForConnection\(\)](#) action on either the MediaRenderer or MediaServer will return an AVTransport [InstanceID](#) to the control point. This AVTransport [InstanceID](#) is used by the control point to control the transfer of the content (for example, [AVTransport::Play\(\)](#), [AVTransport::Pause\(\)](#), [AVTransport::Stop\(\)](#), [AVTransport::Seek\(\)](#), etc). Refer to the subclauses A.3 and A.4 below for more details.

Depending on which transfer protocols are supported by the device (for example, devices that only support HTTP GET), a MediaRenderer and/or MediaServer may choose to not implement the [ConnectionManager::PrepareForConnection\(\)](#) action. In this case, the control point may not have been able to obtain an AVTransport [InstanceID](#) from either device. When this happens, the control point should use an AVTransport [InstanceID](#) of 0 (zero). If the MediaRenderer has implemented the AVTransport service, the control point should use it for all AVTransport actions. Otherwise, AVTransport actions should be sent to the MediaServer device. Refer to the ConnectionManager service [9] for more information.

A.3 Controlling the Transfer of the Content

In all cases, the control point uses the [InstanceID](#), obtained as described above, to control the flow of the content. For example, to begin transferring the content, the control point invokes the [AVTransport::Play\(\)](#) action. To skip to a specific location within the content, the control point invokes the [AVTransport::Seek\(\)](#) action. In most cases, the choice of AVTransport

actions that are actually invoked will likely be directed by the end-user while interacting with the control point's UI. Refer to the AVTransport service specification [5] for additional details about these and other AVTransport actions.

A.4 Controlling How the Content is Rendered

Similar to the allocation of AVTransport InstanceIDs, the MediaRenderer's ConnectionManager::PrepareForConnection() action will also return a RenderingControl InstanceID. This InstanceID is used in conjunction with the RenderingControl service to control how the content is to be rendered. For example, to change the loudness of the sound, the control point invokes the RenderingControl::SetVolume() action. The control point passes the RenderingControl InstanceID and the desired volume setting as input parameters. To get the current brightness of the MediaRenderer's display, the control point invokes the RenderingControl::GetBrightness() action. The InstanceID is passed as an input parameter and the current brightness setting is returned. Refer to the RenderingControl service for additional details on these and other actions that affect how content is rendered.

Annex B (informative)

Bibliography

The following documents, in whole or in part, may be useful for understanding this document but they are not essential 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.

[36] – *XML Schema for UPnP AV Datastructure Template*, UPnP Forum, September 30, 2008. Available at: <http://www.upnp.org/schemas/av/avdt-v1-20080930.xsd>. Latest version available at: <http://www.upnp.org/schemas/av/avdt.xsd>.

[37] – *AVArchitecture:2*, UPnP Forum, March 31, 2013. Available at: <http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2-20130331.pdf>. Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2.pdf>.

[38] – ISO/IEC CD 21000-2:2001, *Information Technology - Multimedia Framework - Part 2: Digital Item Declaration*, July 2001.

[39] – *DeviceProtection:1*, UPnP Forum, February 24, 2011. Available at: <http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service-20110224.pdf>. Latest version available at: <http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf>.

[40] – *HyperText Transport Protocol – HTTP/1.1*, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999. Available at: <http://www.ietf.org/rfc/rfc2616.txt>.

[41] – *IEC 61883 Consumer Audio/Video Equipment – Digital Interface - Part 1 to 5*. Available at: <http://www.iec.ch>.

[42] – *IEC-PAS 61883 Consumer Audio/Video Equipment – Digital Interface - Part 6*. Available at: <http://www.iec.ch>.

[43] – *IEEE P802.1AS™ (Draft 7.0) - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, Institute of Electrical and Electronics Engineers, March 23, 2010. Available at: <http://www.ieee802.org/1/pages/802.1as.html>.

[44] – *IEEE-P1733™ (Draft 2.2) – Audio Video Bridge Layer 3 Transport Protocol*, International Institute of Electrical and Electronics Engineers, April 20, 2009. Available at: <http://grouper.ieee.org/groups/1733>.

[45] – *Data elements and interchange formats – Information interchange -- Representation of dates and times*, International Standards Organization, December 21, 2000. Available at: [ISO 8601:2000](http://www.iso.org/iso/8601.html).

[46] – *IETF RFC 1341, MIME (Multipurpose Internet Mail Extensions)*, N. Borenstein, N. Freed, June 1992. Available at: <http://www.ietf.org/rfc/rfc1341.txt>.

[47] – *MediaRenderer:3*, UPnP Forum, March 31, 2013. Available at: <http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v3-Device-20130331.pdf>. Latest version available at: <http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v3-Device.pdf>.

[48] – *MediaServer:4*, UPnP Forum, March 31, 2013. Available at: <http://www.upnp.org/specs/av/UPnP-av-MediaServer-v4-Device-20130331.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-AV-MediaServer-v4-Device.pdf>.

[49] – *IETF RFC 1305, Network Time Protocol (Version 3) Specification, Implementation and Analysis*, David L. Mills, March 1992.

Available at: <http://www.ietf.org/rfc/rfc1305.txt>.

[50] – *IETF RFC 1321, The MD5 Message-Digest Algorithm*, R. Rivest, April 1992.

Available at: <http://tools.ietf.org/html/rfc1321>.

[51] – *IETF RFC 1738, Uniform Resource Locators (URL)*, Tim Berners-Lee, et. Al., December 1994.

Available at: <http://www.ietf.org/rfc/rfc1738.txt>.

[52] – *IETF RFC 2030, Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OS*, D Mills, October 1996.

Available at: <http://www.ietf.org/rfc/rfc2030.txt>.

[53] – *IETF RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part 1:Format of Internet Message Bodies*, N. Freed, N. Borenstein, November 1996.

Available at: <http://www.ietf.org/rfc/rfc2045.txt>.

[54] – *IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels*, S. Bradner, 1997.

Available at: <http://www.faqs.org/rfcs/rfc2119.html>.

[55] – *IETF RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax*, January 2005.

Available at: <http://www.ietf.org/rfc/rfc3986.txt>.

[56] – *IETF RFC 3174, US Secure Hash Algorithm 1 (SHA1)*, D. Eastlake et al, September 2001.

Available at: <http://tools.ietf.org/html/rfc3174>.

[57] – *IETF RFC 3339, Date and Time on the Internet: Timestamps*, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002.

Available at: <http://www.ietf.org/rfc/rfc3339.txt>.

[58] – *IETF RFC 4078, The TV-Anytime Content Reference Identifier (CRID)*, N. Earnshaw et al, May 2005.

Available at: <http://www.ietf.org/rfc/rfc4078.txt>.

[59] – *IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications*, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, July 2003.

Available at: <http://www.ietf.org/rfc/rfc3550.txt>.

[60] – *IETF RFC 2326, Real Time Streaming Protocol (RTSP)*, H. Schulzrinne, A. Rao, R. Lanphier, April 1998.

Available at: <http://www.ietf.org/rfc/rfc2326.txt>.

[61] – *Unicode Standard Annex #15, Unicode Normalization Forms, version 4.1.0, revision 25*, M. Davis, M. Dürst, March 25, 2005.

Available at: <http://www.unicode.org/reports/tr15/tr15-25.html>.

[62] – *Unicode Technical Standard #10, Unicode Collation Algorithm version 4.1.0*, M. Davis, K. Whistler, May 5, 2005.

Available at: <http://www.unicode.org/reports/tr10/tr10-14.html>.

[63] – *Unicode Technical Standard #10, Unicode Collation Algorithm, version 4.1.0, revision 14*, M. Davis, K. Whistler, May 5, 2005.

Available at: <http://www.unicode.org/reports/tr10/tr10-14.html>.

[64] – *Unicode Technical Standard #35, Locale Data Markup Language, version 1.3R1, revision 5*, M. Davis, June 2, 2005.

Available at: <http://www.unicode.org/reports/tr35/tr35-5.html>.

[65] – *IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace*, P. Leach, Microsoft, M. Mealling, Refactored Networks LLC, R. Salz, DataPower Technology, Inc., July 2005.

Available at: <http://www.ietf.org/rfc/rfc4122.txt>.

[66] – *Extensible Markup Language (XML) 1.0 (Third Edition)*, François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004.

Available at: <http://www.w3.org/TR/2004/REC-xml-20040204>.

[67] – *XML Path Language (XPath) 2.0*. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, Jerome Simeon. W3C Recommendation, 21 November 2006.

Available at: <http://www.w3.org/TR/xpath20>.

[68] – *XQuery 1.0 An XML Query Language*. W3C Recommendation, 23 January 2007.

Available at: <http://www.w3.org/TR/2007/REC-xquery-20070123>.