UPnP AV Datastructure Template:1

For UPnP Version 1.0
Status: Standardized DCP (SDCP)
Date: March 31, 2013
Service 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.

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
Wouter van der Beek	Philips
Wim Bronnenberg	Philips
Jeffrey Kang	Philips
Geert Knapen	Philips
Russell Berkoff	Pioneer
Irene Shen	Pioneer
Russell Berkoff (Vice-Chair)	Samsung Electronics
Norifumi Kikkawa	Sony
Jonathan Tourzan	Sony
Yasuhiro Morioka	Toshiba
Nicholas Frame	TP Vision
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.	

CONTENTS

1 Scope 4
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 9
4.1.3 Extended Backus-Naur Form 10
4.2 Derived Data Types 10
4.2.1 Summary 10
4.2.2 CSV Lists 10
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 AV Datastructure Template 20
6 AV Datastructure Schema 26
Annex A (informative) Bibliography 27

List of Tables

Table 1 — EBNF Operators 10
Table 2 — CSV Examples 11
Table 3 — Namespace Definitions 13
Table 4 — Schema-related Information 15
Table 5 — Default Namespaces for the AV Specifications 17

List of Figures

Figure 1 — Typical Usage of AVDT 5

1 Scope

This document defines the layout of the AV Datastructure Template (AVDT) XML document. An AVDT document describes the format requirements and restrictions of various data structures used within the UPnP AV specifications. Although these data structures are defined very precisely in the appropriate service specification, in most cases, each data structure definition allows for a certain degree of variation in order to accommodate differences between individual devices.

The purpose of an AVDT document is to enable each device to describe (at run-time) its particular variation of these AV data structures. AVDT documents allow users of AV data structures (e.g. UPnP control points) to reduce the number of instances of those data structures that comply with the service specification but are not compatible with the device's particular capabilities. The ultimate goal of an AVDT document is to reduce those error conditions that are caused by control points creating instances of a data structure that exceed the static (known) capabilities of the device. Unfortunately, the AVDT mechanism will never eliminate all preventable error conditions, but it will help to reduce them by giving the client more information about the device's particular capabilities.

As described above, an AVDT document is a machine readable, implementation-specific variant of an AV data structure defined by one of the UPnP AV specifications. For a given device, each instance of that data structure shall conform to both the specification definition and the device's AVDT definition of that data structure.

Ironically, an AVDT document is both a more-restrictive and more-permissive variant of the specification definition. AVDT documents are more restrictive because they limit certain aspects of the data structure (e.g. such as the allowed values for each field) that are otherwise permitted by the specification definition. However, due to limitations of the AVDT constructs, it is simply not possible to express some of the more intricate requirements defined by the specification (e.g. subtle interdependencies between data structure fields). Consequently, instances of a data structure that comply with a given AVDT description may not fully comply with all of the requirements defined in the specification.

The types of data structures that can be described by an AVDT document represent a (non-hierarchical) set of named property values. The set of allowed property names and their allowed values for a given data structure are defined by one of the UPnP AV specifications. Individual instances of these data structures are manifested via an XML document whose elements and attributes correspond to the set of named properties. In other words, within the XML document that corresponds to a given instance of a certain data structure, each XML element and attribute contains the value of a specific named property.

An AVDT document is conceptually similar to an XML schema in that both entities identify the XML elements and attributes that appear in any given document instance. Additionally, both AVDT documents and XML schemas identify the allowed values that are permitted for each element and/or attribute which corresponds to a specific property. However, unlike an XML schema, an AVDT document can also identify certain dependencies between two or more properties. For example, the set of allowed values of one property may depend on the actual value of another property. This type of interrelationship is difficult to represent using an XML schema. Hence, the AVDTdocument structure is needed.

In the various AV Architecture scenarios, sometimes there is a need to exchange device capabilities to ensure high level interoperability. In order to express the parameterized capability, an AV specification defines various templates for each purpose. A device uses the template and populates it with values to reflect its capabilities at run-time.

The AV Datastructure Template (AVDT) is a common structure to define various templates, which are called "Datastructure". This is written in XML and each data structure uses a subset of the AVDT to meet the necessary requirement.

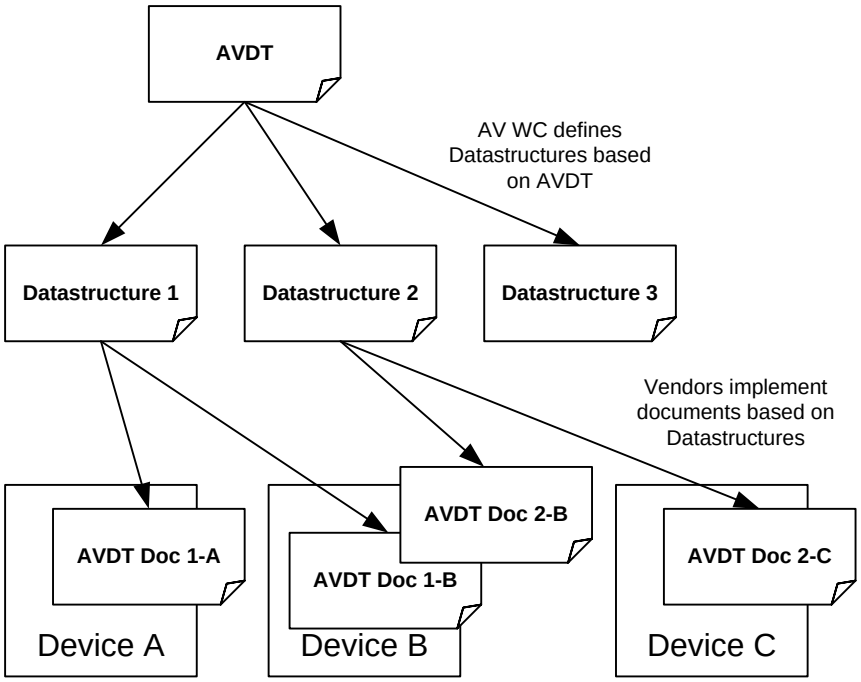

Figure 1 — Typical Usage of AVDT

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 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.

[4] – *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.

[5] – *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.

[6] – *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.

[7] – *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.

[8] – *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.

[9] – *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.

[10] – *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.

[11] – *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.

[12] – *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.

[13] – *XML Schema for UPnP AV Dublin Core*.
Available at: http://www.dublincore.org/schemas/xmls/simpledc20020312.xsd.

[14] – *DCMI term declarations represented in XML schema language*.
Available at: http://www.dublincore.org/schemas/xmls.

[15] – *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.

[16] – *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.

[17] – *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.

[18] – *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.

[19] – *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.

[20] – ISO/IEC 14977, *Information technology - Syntactic metalanguage - Extended BNF*, December 1996.

[21] – *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.

[22] – *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.

[23] –*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.

[24] – *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.

[25] – *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.

[26] – *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.

[27] – *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.

[28] – *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.

[29] – *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.

[30] – *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.

[31] – *The "xml:" Namespace*, November 3, 2004.
Available at: http://www.w3.org/XML/1998/namespace.

[32] – *XML Schema for the "xml:" Namespace*.
Available at: http://www.w3.org/2001/xml.xsd.

[33] – *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.

[34] – *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.

[35] – *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.

[36] – *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 [15] 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 [15] are styled in **green bold underlined** text.
- UPnP interface names defined outside of the UPnP Device Architecture specification [15] 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 [15].
- Basic data types for properties are defined in [35].
- Additional data types for properties are defined in the XML schema(s) (see [4]) 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 [20].

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 [15], 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 [15] (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 [15]: **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 [35]: 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 [33] 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
AV Working Committee defined namespaces			
atrs	urn:schemas-upnp-org:av:AllowedTransformSettings	*AllowedTransformSettings* and *AllowedDefaultTransformSettings* state variables for RenderingControl	[22]
av	urn:schemas-upnp-org:av:av	Common data types for use in AV schemas	[4]
avdt	urn:schemas-upnp-org:av:avdt	Datastructure Template	[2]
avs	urn:schemas-upnp-org:av:avs	Common structures for use in AV schemas	[5]
avt-event	urn:schemas-upnp-org:metadata-1-0/AVT/	Evented *LastChange* state variable for AVTransport	[6]
cds-event	urn:schemas-upnp-org:av:cds-event	Evented *LastChange* state variable for ContentDirectory	[8]
cm-dciu	urn:schemas-upnp-org:av:cm-deviceClockInfoUpdates	Evented *DeviceClockInfoUpdates* state variable for ConnectionManager	[10]
cm-ftrlst	urn:schemas-upnp-org:av:cm-featureList	*FeatureList* state variable for ConnectionManager	[10]
didl-lite	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[8]
dmo	urn:schemas-upnp.org:av:dmo	Evented *DeviceMode* state variable for ContentDirectory	[8]
dmor	urn:schemas-upnp.org:av:dmor	*A_ARG_TYPE_DeviceModeRequest* state variable for ContentDirectory	[8]
dmos	urn:schemas-upnp.org:av:dmos	*DeviceModeStatus* state variable for ContentDirectory	[8]
pi	urn:schemas-upnp.org:av:pi	*PermissionsInfo* state variable for ContentDirectory	[8]
rcs-event	urn:schemas-upnp-org:metadata-1-0/RCS/	Evented *LastChange* state variable for RenderingControl	[22]
rii	urn:schemas-upnp-org:av:rii	*A_ARG_TYPE_RenderingInfoList* state variable for ConnectionManager	[10]
rpl	urn:schemas-upnp-org:av:rpl	*A_ARG_TYPE_PlaylistInfo* state variable for AVTransport	[6]
srs	urn:schemas-upnp-org:av:srs	Metadata and structure for ScheduledRecording	[26]
srs-event	urn:schemas-upnp-org:av:srs-event	Evented *LastChange* state variable for ScheduledRecording	[26]
trs	urn:schemas-upnp-org:av:TransformSettings	*TransformSettings* and *DefaultTransformSettings* state variables for RenderingControl	[22]
upnp	urn:schemas-upnp-org:metadata-1-0/upnp/	Metadata for ContentDirectory	[8]

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
Externally defined namespaces			
dc	http://purl.org/dc/elements/1.1/	Dublin Core	[14]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[34], [35]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	[34] 2.6 & 3.2.7
xml	http://www.w3.org/XML/1998/namespace	The “xml:” Namespace	[31]

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
AV Working Committee Defined Namespaces			
atrs	AllowedTransformSettings-v*n*-*yyyymmdd*.xsd AllowedTransformSettings-v*n*.xsd AllowedTransformSettings.xsd	`<TransformList>`	[1]
av	av-v*n*-*yyyymmdd*.xsd av-v*n*.xsd av.xsd	*n/a*	[4]
avdt	avdt-v*n*-*yyyymmdd*.xsd avdt-v*n*.xsd avdt.xsd	`<AVDT>`	[2]

avs	avs-v*n*-*yyyymmdd*.xsd
	avs-v*n*.xsd
	avs.xsd
		`<Capabilities>`
		`<Features>`
		`<stateVariableValuePairs>`
			[5]

avt-event	avt-event-v*n*-*yyyymmdd*.xsd avt-event-v*n*.xsd avt-event.xsd	`<Event>`	[7]
cds-event	cds-event-v*n*-*yyyymmdd*.xsd cds-event-v*n*.xsd cds-event.xsd	`<StateEvent>`	[9]
cm-dciu	cm-deviceClockInfoUpdates-v*n*-*yyyymmdd*.xsd cm-deviceClockInfoUpdates-v*n*.xsd cm-deviceClockInfoUpdates.xsd	`<DeviceClockInfoUpdates>`	[11]
cm-ftrlst	cm-featureList-v*n*-*yyyymmdd*.xsd cm-featureList-v*n*.xsd cm-featureList.xsd	`<Features>`	[12]
didl-lite	didl-lite-v*n*-*yyyymmdd*.xsd didl-lite-v*n*.xsd didl-lite.xsd	`<DIDL-Lite>`	[16]
dmo	dmo-v*n*-*yyyymmdd*.xsd dmo-v*n*.xsd dmo.xsd	`<DeviceMode>`	[17]
dmor	dmor-v*n*-*yyyymmdd*.xsd dmor-v*n*.xsd dmor.xsd	`<DeviceModeRequest>`	[18]
dmos	dmos-v*n*-*yyyymmdd*.xsd dmos-v*n*.xsd dmos.xsd	`<DeviceModeStatus>`	[19]

Standard Name-space Prefix	Relative URI and File Name [a] • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
pi	pi-v*n*-*yyyymmdd*.xsd pi-v*n*.xsd pi.xsd	<PermissionsInfo>	[21]
rcs-event	rcs-event-v*n*-*yyyymmdd*.xsd rcs-event-v*n*.xsd rcs-event.xsd	<Event>	[23]
rii	rii-*vn-yyyymmdd*.xsd rii-v*n*.xsd rii.xsd	<rendererInfo>	[24]
rpl	rpl-*vn-yyyymmdd*.xsd rpl-v*n*.xsd rpl.xsd	<PlaylistInfo>	[25]
trs	TransformSettings-*vn-yyyymmdd*.xsd TransformSettings-v*n*.xsd TransformSettings.xsd	<TransformSettings>	[29]
srs	srs-v*n-yyyymmdd*.xsd srs-v*n*.xsd srs.xsd	<srs>	[27]
srs-event	srs-event-v*n*-*yyyymmdd*.xsd srs-event-v*n*.xsd srs-event.xsd	<StateEvent>	[28]
upnp	upnp-v*n-yyyymmdd*.xsd upnp-v*n*.xsd upnp.xsd	*n/a*	[30]
Externally Defined Namespaces			
dc	*Absolute URL*: http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[13]
xsd	*n/a*	<schema>	[36]
xsi	*n/a*		*n/a*
xml	*n/a*		[32]

[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 [33] for both reading and writing. As allowed by [33], 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 namepace 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" 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". 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") 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" was linked to the schema that is otherwise known as ".../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".

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 AV Datastructure Template

The following shows the generalized layout of an AVDT Template. More elements and/or attributes may be added in future versions of AVDT templates.

The *forum* character style is used to indicate names defined by the AVWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0"?>
<AVDT
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:avdt
      http://www.upnp.org/schemas/av/avdt.xsd"
 xmlns="urn:schemas-upnp-org:av:avdt">
   <contextID>data structure identification context</contextID>
   <dataStructType>data structure name</dataStructType>
   <fieldTable>
      <field>
         <name>field name</name>
         <dataType csv="csv data type" maxSize="max length">
            field data type
         </dataType>
         <minCountTotal>minimum total occurrences</minCountTotal>
         <maxCountTotal>maximum total occurrences</maxCountTotal>
         <minListSizeTotal>min # of entries in CSV</minListSizeTotal>
         <maxListSizeTotal>max # of entries in CSV</maxListSizeTotal>
         <allowedValueDescriptor>
```

```
            <dependentField defaultDependency="1|0">
               <name>field name</name>
               <anyValue></anyValue>
               <valueList>
                  <value>enumerated value</value>
                  // Other values go here
               </valueList>
               <valueRange>
                  <minimum>minimum value</minimum>
                  <maximum>maximum value</maximum>
                  <step>increment value</step>
               </valueRange>
               // Other value ranges go here
            </dependentField>
            // Other dependent fields go here
            <minCount>minimum occurrences of these values</minCount>
            <maxCount>maximum occurrences of these values</maxCount>
            <minListSize>minimum # of these values in CSV</minListSize>
            <maxListSize>maximum # of these values in CSV</maxListSize>
            <defaultValue>default value</defaultValue>
            <allowAny></allowAny>
            <allowedValueList>
               <allowedValue>enumerated value</allowedValue>
               // Other allowed values go here
            </allowedValueList>
            <allowedValueRange>
               <minimum>minimum value</minimum>
               <maximum>maximum value</maximum>
               <step>increment value</step>
            </allowedValueRange>
            // Other allowed value Ranges go here
         </allowedValueDescriptor>
         // Other allowed value descriptors go here
      </field>
      // Other field declarations go here
   </fieldTable>
</AVDT>
```

xml
Required for all XML documents. Case sensitive.

AVDT
Required. Shall have “urn:schemas-upnp-org:av:avdt” as the value for the xmlns attribute; this references the UPnP AV Working Committee Datastructure Template Schema. As long as the same xmlns is used, the data structure template shall be backward compatible, i.e. usable by legacy implementations. Contains all other elements describing the service, i.e., contains the following sub elements:

contextID
Required. xsd:anyType. Identifies the context in which the data structure type has meaning. Typically, this element contains a unique identifier for the device-specific service instance that contains this data structure.
For example, uuid:*device-UUID*::urn:**schemas-upnp-org**:**service**:scheduleRecording:1.

dataStructType
Required. xsd:QName. Identifies the data structure type. The name of the data structure type is vendor-dependent. It shall be a QName as defined in clause 3 of the W3C document “Namespaces in XML” [33]. Identical data structure types shall be identified by the same name. Likewise, data structure types that are different shall have different names.

fieldTable
Required. Begins the description for the fields that are defined for this data structure type. Contains zero or more of the following sub element(s):

field
Required. Repeat once for each **field** that is contained within this data structure type. Contains the following sub elements:

name

Required. xsd:string. Identifies the name of the **field** that is described within this **field** element. Shall be one of the following formats:

- QName
- QName "@" NCName
- "@" NCName
- NCName ":@" NCName

where QName and NCName are defined in clause 3 of the W3C document "Namespaces in XML" [33]. For **field**s that correspond to an XML element (within the data structure's (**dataStructType**) XML document) **name** shall contain the name of the XML element using the QName format e.g. element-name. For **field**s that correspond to an XML attribute (within the data structure's (**dataStructType**) XML document) **name** shall contain the name of the XML attribute using any of the forms other than the QName format e.g. element-name@attribute-name.

datatype

Required. xsd:string. Identifies the data type of this **field**. Shall be a QName with a namespace prefix of "xsd". QName is defined in clause 3 of the W3C document "Namespaces in XML" [33]. Shall be one, and only one, of the data types defined by "XML Schema Part-2" [35]. Contains the following attributes:

@csv

Allowed. xsd:string. If present, indicates that this string **field** contains a CSV list of values (called "entries") of the data type specified by the CSV attribute. Shall comply with the CSV data type notation identified in subclause 4.2.2. For example, a value of "xsd:int" indicates a CSV of integer values. AVDT does not impose any restrictions on the data type value that may be specified. However, each data structure defined by an AVDT instance (**dataStructType**) will use only a limited number of CSV data types. Shall only be specified when **datatype** equals "string" and the **field** is intended to contain a CSV list of values. The minimum and maximum number of entries in the CSV list are specified by **minListSizeTotal**, **maxListSizeTotal**, **minListSize**, and **maxListSize** defined below.

@maxSize

Allowed. xsd:unsignedInt. Meaningful only when **datatype** equals "string". Indicates the maximum number of bytes allowed for this **field**. Note that since some character sets consume multiple bytes per character (e.g. UTF-16), **maxSize** does not necessarily indicate the maximum number of characters that are allowed.

minCountTotal

Allowed. xsd:unsignedInt. Minimum number of occurrences of this **field** within the entire XML document. The default value is 0 which means this **field** is allowed and might not be included in some instances of this data structure (**dataStructType**). A value of 1 or more means that this **field** is required and shall be present in every instance of this data structure at least the specified number of times.

maxCountTotal

Allowed. xsd:string. Maximum number of occurrences of this **field** within the entire XML document. Its value shall be either an unsigned integer or the value "UNBOUNDED". The default value is 1 which means this **field** shall not be present more than once within any instance of this data structure (**dataStructType**). A value of 0 indicates that this **field** is not allowed and shall not be present in any instance of this data structure. A value of "UNBOUNDED" indicates that there is no predetermined limit on the number of times this **field** may be present. The value of **maxCountTotal** shall be greater than or equal to **minCountTotal**.

minListSizeTotal

Allowed. xsd:unsignedInt. Valid only for a CSV-type **field** i.e. when the **@csv** attribute is specified within **name**. Minimum number of entries in each instance of this CSV **field**. The default value is 0 which means this **field**, when present, may contain an empty CSV list. A value of 1 or more means that this **field**, when present, shall contain at least the specified number of entries in the CSV list.

maxListSizeTotal

Allowed. xsd:string. Valid only for a CSV-type **field** i.e. when the **@csv** attribute is specified within **name**. Maximum number of entries in each instance of this CSV **field**. Its value shall be either a positive integer or the value "UNBOUNDED". The default value is 1 which means this CSV **field** shall not contain more than one entry at a time. A value of "UNBOUNDED" indicates that there is no predetermined limit on the number of entries in the CSV list. The value of **maxListSizeTotal** shall be greater than or equal to **minListSizeTotal**.

allowedValueDescriptor

Required. Begins the description of an allowed value data set for this **field**. Multiple **allowedValueDescriptor** elements are permitted. The total span of

allowed values for this **field** is simply a concatenation of the individual allowed values within each **allowedValueDescriptor**. Shall contain either

- **allowAny** or
- **allowedValueList** and/or **allowedValueRange**

Contains the following sub element(s):

dependentField

Allowed. Identifies the values of a "dependent" **field** which define a "validity context" for the allowed value data set being defined within this **allowedValueDescriptor**. In other words, when the **dependentField** is set to one of the values defined within the **dependentField**'s sub elements **anyValue**, **valuelist** and/or **valueRange** sub element, then this **field** shall contain one of the values identified by the **allowedValueDescriptor**'s sub elements **allowAny**, **allowedValueList** and/or **allowedValueRange**. If multiple **dependentField** elements exist within a given **allowedValueDescriptor** element, the "validity context" for the allowed value data set exists whenever all of the **dependentField**s are set to their specified value/range i.e. multiple **dependentField** entries are "ANDed" together to define a specific "context" for the allowed values that follow. A missing **dependentField** element indicates that the allowed values of this **allowedValueDescriptor** are valid in all contexts except for those contexts that are identified by other peer **allowedValueDescriptor** blocks defined within this **field**

Shall contain either

- **anyValue** or
- **valueList** and/or **valueRange**

Contains the following attributes and sub element(s):

@defaultDependency

Allowed. xsd:boolean. A value of 1 indicates that the **value**/**valueRange**(s) defined within this **dependentField** include the default value (**defaultValue**) of the **dependentField**. The default value for **defaultDependency** is 0 which means that the default value of this **dependentField** is not included in the **value**/**valueRange**(s) defined within this **dependentField**. Used by control points that do not support the **dependentField** in order to identify the set of allowed values that reflect the device's capabilities when the **dependentField** contains its default value.

name

Required. xsd:string. Identifies the name of a **dependentField** whose value affects the set of allowed values for this **field**. In other words, the set of allowed values for this **field** depends on the value of the **dependentField**. Shall follow the format rules defined in the **name** sub element of **field**.

anyValue

Allowed. xsd:string. The existence of this element indicates that this **dependentField** may be set to any value allowed by the **dependentField**'s data type. The content of this element shall be empty. **anyValue** shall not be included along with the **valueList** or **valueRange** elements.

valueList

Allowed. Enumerates a set of values for the **dependentField** that constrain this **field** to the set of allowed values defined within this **allowedValueDescriptor**. Multiple **valueList** elements shall not be specified. Shall not be included along with the **anyValue** element. Contains the following sub elements:

value

Required. xsd:anyType. Identifies a legal value of this **field**. Legal values are typically defined by the UPnP Forum AV Working Committee. However, vendors may, if the working committee permits it, add vendor-specific allowed values. An empty **value** element means that when the **dependentField** is empty, then this **field** shall contain one of the allowed values defined within this **allowedValueDescriptor**.

valueRange

Allowed. Defines a range and resolution for a set of values for the **dependentField** that constrain this **field** to the set of allowed values defined within this **allowedValueDescriptor**. Valid only for numeric data types. Multiple **valueRange** elements may be specified. Shall not be included along with the **anyValue** element. Contains the following sub elements:

minimum
Required. xsd:string. Single numeric value. Inclusive lower bound. Shall be less than **maximum**.
Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

maximum
Required. xsd:string. Single numeric value. Inclusive upper bound. Shall be greater than **minimum**.
Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

step
Allowed. xsd:string. Single positive numeric value. Indicates the numeric difference between adjacent supported values within the **valueRange**. The value of step shall divide the inclusive range from minimum to maximum into an integral number of equal parts. In other words, **maximum** = **minimum** + N***step** where N is a positive integer. When **step** is omitted AND the data type of the **dependentField** is an integer, the default value of **step** is 1. Otherwise, when **step** is omitted, all values within the inclusive range from **minimum** to **maximum** are valid.
Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

minCount
Allowed. xsd:unsignedInt. Minimum number of occurrences of this **field** that has one of the values defined within this **allowedValueDescriptor**. Indicates the minimum number of times (within the entire XML document) that this **field** shall be set to one of the values defined within this **allowedValueDescriptor**. The default value is 0 which means that the XML document might not contain any occurrences of this **field** whose value is set to one of the values defined within this **allowedvalueDescriptor**. A value of 1 or greater means that this **field** is required and shall be present at least the specified number of times. Additionally, each of those occurrences shall be set to one of the values defined within this **allowedValueDescriptor**. Other instances of this **field** may occur but they shall have a value defined within a different **allowedValueDescriptor**. For each **field**, the value of **minCount** shall be less than or equal to **minCountTotal**.

maxCount
Allowed. xsd:string. Maximum number of occurrences of this **field** that has one of the values defined within this **allowedValueDescriptor**. Indicates the maximum number of times this **field** can be set to one of the values defined within this **allowedValueDescriptor**. The value of **maxCount** shall be either an unsigned integer or the value “UNBOUNDED”. A value of 1 or greater indicates that this **field** shall not be present more than the specified number of times with a value set to one of the values defined within this **allowedValueDescriptor**. The default value is 1. A value of 0 means that the data structure shall not include any occurrences of this **field** other than those occurences whose value is defined within a different **allowedValueDescriptor**. In this case, the **allowedValueDescriptor** shall contain an empty **allowedValueList** and no **allowedValueRange**. A value of “UNBOUNDED” indicates that there is no predetermined limit on the number of occurences of this **field** that may contain one of the values defined within this **allowedValueDescriptor**.The value of **maxCount** shall be greater than or equal to **minCount**. For each occurrence of **allowedValueDescriptor** the value of **maxCount** shall be less than or equal to **maxCountTotal** for this **field**. Other instances of this **field** may occur but they shall have a value defined within a different **allowedValueDescriptor**.

minListSize
Allowed. xsd:unsignedInt. Valid only for a CSV-type **field** i.e. when the **@csv** attribute is specified within the **name** sub element of **field**. Minimum number of entries in each instance of this CSV **field** that shall contain one of the values defined within this **allowedValueDescriptor**. The default value is 0 which means this **field**, when present, might not contain any entries that are defined within this **allowedValueDescriptor**. A value of 1 or more means that this **field**, when present, shall contain at least the specified number of entries whose value is defined within this

allowedValueDescriptor. Other instances of this **field** may occur but they shall have a value defined within a different **allowedValueDescriptor**.

maxListSize

Allowed. xsd:string. Valid only for a CSV-type **field** i.e. when the **@csv** attribute is specified within the **name** sub element of **field**. Maximum number of entries in each instance of this CSV **field** that are allowed to contain one of the values defined within this **allowedValueDescriptor**. The value of **maxListSize** shall be either an unsigned integer or the value "UNBOUNDED". The default value is 1 which means this CSV **field** shall not contain more than one entry whose value is defined within this **allowedValueDescriptor**. A value of 0 means that no entries within any instance of this CSV **field** are allowed to contain one of the values defined within this **allowedValueDescriptor**. In this case, the **allowedValueDescriptor** shall contain an **allowedValueList** with a single, empty **allowedValue** and no **allowedValueRange**. A value of "UNBOUNDED" indicates that there is no predetermined limit on the number entries that contain one of the values defined within this **allowedValueDescriptor**. Other instances of this **field** may occur but they shall have a value defined within a different **allowedValueDescriptor**. The value of **maxListSize** shall be greater than or equal to **minListSize** and less than or equal to **maxListSizeTotal**.

defaultValue

Allowed. xsd:anyType. Identifies the default value assigned to this **field** if no value is present in the XML document. The contents shall match the data type (**datatype**) of this **field** and it shall belong to the set of allowed values defined within this **allowedValueDescriptor** i.e. **allowAny**, **allowedValueList,** and/or **allowedValueRange**. If this **field** appears as a **dependentField** within another **field**, then that **dependentField** element shall contain the **defaultDependency** attribute with a value of 1.

allowAny

Allowed. xsd:string. The existence of this element indicates that this field may be set to any value allowed by this **field**'s data type. The content of this element shall be empty. **allowAny** shall not be included along with the **allowedValueList** or **allowedValueRange** elements.

allowedValueList

Allowed. Enumerates a set of values that are allowed for this **field** subject to the contraints defined by the **dependentField** element, if present. Multiple **allowedValueLists** shall not be specified. **allowedValueList** shall not be included along with the **allowAny** element. Contains the following sub elements:

allowedValue

Required. xsd:anyType. Identifies one of the values that are allowed for this **field**. Legal values are typically defined by the UPnP Forum AV Working Committee. However, vendors may, if the working committee permits it, add vendor-specific allowed values. An empty **allowedValue** element means that the content of this **field** is permitted to be empty. An **allowedValueList** with only an empty **allowedValue** means that when this **field** exists, its value shall be empty. For a CSV-type **field** (**@csv**), an **allowedValue** entry indicates one possible value of an entry in the CSV list. It does not indicate one of the possible combinations of values for the entire CSV list.

Note that for a heterogeneous CSV-type **field**, it might not be practical to enumerate all of the allowed values that are possible. In this case, it is recommended to specify **allowAny**.

allowedValueRange

Allowed. Defines a range and resolution for a set of numeric values that are allowed for this **field** subject to the contraints defined by the **dependentField** element, if present. Valid only for numeric data types. Multiple **allowedValueRange** elements may be specified. Shall not be included along with the **allowAny** element. Contains the following sub elements:

minimum

Required. xsd:string. Single numeric value. Inclusive lower bound. Shall be less than **maximum**.

Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

maximum
Required. xsd:string. Single numeric value. Inclusive upper bound. Shall be greater than **minimum**.
Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

step
Allowed. xsd:string. Single positive numeric value. Indicates the numeric difference between adjacent valid values within the **allowedValueRange**. The value of step shall divide the inclusive range from minimum to maximum into an integral number of equal parts. In other words, **maximum** = **minimum** + N***step** where N is a positive integer. When **step** is omitted and the data type of the **field** is an integer, the default value of **step** is 1. Otherwise, if **step** is omitted, all values within the inclusive range from **minimum** to **maximum** shall be supported.

Note: Care needs to be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

6 AV Datastructure Schema

The AV Datastructure XML schema defines the structure of an AVDT document. A machine readable file containing this schema can be found at [3]. Although the XML schema does not include any extensibility mechanisms (e.g. the inclusion of <xsd:any> tags), AVDT documents are permitted to include additional XML elements and/or attributes beyond those defined in this schema. This allows for vendor-defined extensions and/or for future (standardized) enhancements to the AVDT structure. Consequently, when parsing an AVDT document any unrecognized elements and/or attributes shall be gracefully ignored.

Each Datastructure (identified by the <contextID> and <dataStructType> elements) is described by an AVDT Document which in turn is an instantiation of a particular version of the AVDT schema. As the AVDT schema is enhanced over time, each version is assigned a unique number as indicated by the latter part of the schema URN as follows (see subclause 4.3.2):

```
xsi:schemaLocation="
   urn:schemas-upnp-org:av:avdt
      http://www.upnp.org/schemas/av/avdt.xsd"
```

where the number 1 after the “v” is the version number. Each AVDT schema version update shall be backward compatible with the previous version. Specifically, XML elements and/or attributes may be added to more recent AVDT schema versions, but shall not ever be removed. As a result, when examining the schema version value, implementations will likely want to perform a greater-than-or-equal-to comparison rather than just a plain equality check.

Annex A
(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.

[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.

[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.