UPnP AV Datastructure Template:1

For UPnP™ Version 1.0
Status: Approved Standard
Date: May 31, 2006
Document Version: 1.00

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 Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP 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 Membership Agreement.

Authors	Company
Alan Presser	Allegrosoft
Gary Langille	Echostar
Gerrie Shults	HP
John Ritchie (Co-Chair)	Intel
Mark Walker	Intel
Changhyun Kim	LGE Electronics
Sungjoon Ahn	LGE Electronics
Masatomo Hori	Matsushita Electric (Panasonic)
Matthew Ma	Matsushita Electric (Panasonic)
Jack Unverferth	Microsoft
Wim Bronnenberg	Philips
Geert Knapen (Co-Chair)	Philips
Russell Berkoff	Pioneer
Irene Shen	Pioneer
Norifumi Kikkawa	Sony

Authors	Company
Jonathan Tourzan	Sony
Yasuhiro Morioka	Toshiba

Contents

List of Tables

List of Figures

1 Introduction

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 must 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.

1.1 Notation

- In this document, features are described as Required, Recommended, or Optional as follows:

  The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in [RFC 2119].

  In addition, the following keywords are used in this specification:

  PROHIBITED – The definition or behavior is an absolute prohibition of this specification. Opposite of REQUIRED.

CONDITIONALLY REQUIRED – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is REQUIRED, otherwise it is PROHIBITED.

CONDITIONALLY OPTIONAL – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is OPTIONAL, otherwise it is PROHIBITED.

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

- Strings that are to be taken literally are enclosed in "double quotes".
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP AV Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.
- A double colon delimiter, "::", 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.

1.1.1 Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types [DEVICE]. The XML Schema namespace is used to define property data types [XML SCHEMA-2].

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 MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all 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 MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all properties be represented as "*0*" and "*1*".

1.1.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that MUST 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 Section 1.2.1, "Comma Separated Value (CSV) Lists") 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

1.1.3 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 [EBNF].

1.1.3.1 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 MUST 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:

Table 1-1: 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 MUST occur at least once and MAY occur more times.
`[ ]`	**optional –** the expression between the brackets is optional.
`( )`	**grouping –** groups the expressions between the parentheses.
`-`	**character range** – represents all characters between the left and right character operands inclusively.

1.2 Derived Data Types

This section 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 applies to both string data types.

1.2.1 Comma Separated Value (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 [DEVICE], 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 (values of different types are allowed). 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 ({*x*, *y*, *z*}), where *x*, *y* and *z* are the types of the individual values in the subsequence and the subsequence 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 [DEVICE] (that is: optional leading sign, optional leading zeroes, numeric 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 [DEVICE]: **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 [XML SCHEMA-2]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in Section 1.1.2, "Strings Embedded in Other Strings".
- 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 1-2: 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

Type refinement of string	Value	Comments
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.

1.3 Management of XML Namespaces in Standardized DCPs

UPnP specifications make extensive use of XML namespaces. This allows 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 is, or should 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 must be locally unique to the document. Any valid XML no-colon-name may be used. And, in formal XML usage, no two XML documents are ever required to use the same namespace prefix to refer to the same namespace. The creation and use of the namespace prefix was standardized by the W3C XML Committee in [XML-NMSP] 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* supercede XML rules for usage in documents, it only augments and complements them in important contexts that are out-of-scope for the XML specifications.

All of the namespaces used in this specification are listed in the Tables “Namespace Definitions” and “Schema-related Information”. For each such namespace, Table 1-3, “Namespace Definitions” 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. The individual specifications in such collections all use the same standard prefix. The standard prefixes are also used in Table 1-4, “Schema-related Information”, to cross-reference additional namespace information. This second table includes each namespace’s valid XML document root elements (if any), its schema file name, versioning information (to

be discussed in more detail below), and links to the entries in the Reference section for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 1-3. 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 1-3: Namespace Definitions

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
AV Working Committee defined namespaces			
av:	urn:schemas-upnp-org:av:av	Common data types for use in AV schemas	[AV-XSD]
avs:	urn:schemas-upnp-org:av:avs	Common structures for use in AV schemas	[AVS-XSD]
avdt:	urn:schemas-upnp-org:av:avdt	Datastructure Template	[AVDT]
avt-event:	urn:schemas-upnp-org:metadata-1-0/AVT/	Evented *LastChange* state variable for AVTransport	[AVT]
didl-lite:	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[CDS]
rcs-event:	urn:schemas-upnp-org:metadata-1-0/RCS/	Evented *LastChange* state variable for RenderingControl	[RCS]
srs:	urn:schemas-upnp-org:av:srs	Metadata and structure for ScheduledRecording	[SRS]
srs-event:	urn:schemas-upnp-org:av:srs-event	Evented *LastChange* state variable for ScheduledRecording	[SRS]
upnp:	urn:schemas-upnp-org:metadata-1-0/upnp/	Metadata for ContentDirectory	[CDS]
Externally defined namespaces			
dc:	http://purl.org/dc/elements/1.1/	Dublin Core	[DC-TERMS]
xsd:	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[XML SCHEMA-1] [XML SCHEMA-2]
xsi:	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	Sections 2.6 & 3.2.7 of [XML SCHEMA-1]
xml:	http://www.w3.org/XML/1998/namespace	The “xml:” Namespace	[XML-NS]

Table 1-4: Schema-related Information

Standard Name-space Prefix	Relative URI and File Name • Form 1 • Form 2	Valid Root Element(s)	Schema Reference
AV Working Committee Defined Namespaces			
av:	• av-v*n-yyyymmdd*.xsd • av-v*n*.xsd	*n/a*	[AV-XSD]

Standard Name-space Prefix	Relative URI and File Name • Form 1 • Form 2	Valid Root Element(s)	Schema Reference
avs:	• avs-v*n*-*yyyymmdd*.xsd • avs-v*n*.xsd	`<Features>` `<stateVariableValuePairs>`	[AVS-XSD]
avdt:	• avdt-v*n*-*yyyymmdd*.xsd • avdt-v*n*.xsd	`<AVDT>`	[AVDT]
avt-event:	• avt-event-v*n*-*yyyymmdd*.xsd • avt-event-v*n*.xsd	`<Event>`	[AVT-EVENT-XSD]
didl-lite:	• didl-lite-v*n*-*yyyymmdd*.xsd • didl-lite-v*n*.xsd	`<DIDL-Lite>`	[DIDL-LITE-XSD]
rcs-event:	• rcs-event-v*n*-*yyyymmdd*.xsd • rcs-event-v*n*.xsd	`<Event>`	[RCS-EVENT-XSD]
srs:	• srs-v*n*-*yyyymmdd*.xsd • srs-v*n*.xsd	`<srs>`	[SRS-XSD]
srs-event:	• srs-event-v*n*-*yyyymmdd*.xsd • srs-event-v*n*.xsd	`<StateEvent>`	[SRS-EVENT-XSD]
upnp:	• upnp-v*n*-*yyyymmdd*.xsd • upnp-v*n*.xsd	*n/a*	[UPNP-XSD]
Externally Defined Namespaces			
dc:	*Absolute URL*: http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[DC-XSD]
xsd:	*n/a*	`<schema>`	[XMLSCHEMA-XSD]
xsi:	*n/a*		*n/a*
xml:	*n/a*		[XML-XSD]

1.3.1 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 MUST use the standard namespace prefixes as declared in Table 1-3. In order to properly process the XML documents described herein, control points and devices MUST use namespace-aware XML processors [XML-NMSP] for both reading and writing. As allowed by [XML-NMSP], 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 MUST be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. 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. Also, 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 1-5, “Default Namespaces for the AV Specifications”.

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

Table 1-5: Default Namespaces for the AV Specifications

AV Specification Name	Default Namespace Prefix
AVTransport:2	avt-event:
ConnectionManager:2	*n/a*
ContentDirectory:2	didl-lite:
MediaRenderer:2	*n/a*
MediaServer:2	*n/a*
RenderingControl:2	rcs-event:
ScheduledRecording:1	srs:

1.3.2 Namespace Names, Namespace Versioning and Schema Versioning

Each namespace that is defined by the AV Working Committee is named by a URN.

In order to enable both forward and backward compatibility, the UPnP TC has established the general policy that namespace names will not change with new versions of specifications, even when the specification changes the definition of a namespace. But, namespaces still have version numbers that reflect definitional changes. Each time the definition of a namespace is changed, the namespace's version number is incremented by one. Therefore, namespace version information must be provided with each XML instance document so that the document's receiver can properly understand its meaning. This is achieved by the following rules:

- Every release of a schema is identified by a version number and date of the form "*n-yyyymmdd*", where *n* corresponds to the namespace definition version number and *yyyymmdd* is the year, month and day in the Gregorian calendar that the schema is released.

  For example, the new version numbers of the pre-existing "DIDL-Lite" and "upnp" schemas are "2". Versions for new schemas, such as "srs" are "1".

  For each schema, the version-date will appear in two places:

  1. In the schema file name, according to the naming structure shown in Table 1-4, "Schema-related Information".
  2. As the value of the `version` attribute of each schema's `schema` root element.

Namespaces are referenced in both schema and XML instance documents by namespace name. The namespace name appears as the value of an `xmlns` attribute. The `xmlns` attribute also declares a namespace prefix that will be used to qualify names from each namespace. Schemas are referenced in both schema and XML instance documents by URI in the `schemaLocation` attribute. See section 1.3.3, "Namespace Usage Examples" . Two different forms of URI are available, each with a different meaning. All UPnP AV-defined schema URIs share a common base path of "http://www.upnp.org/schemas/av/". Each schema URI has two unique relative forms (see Table 1-4, "Schema-related Information"), according to which version of a namespace and its representative schema is of interest. The allowed relative URI forms are:

1. *schema-root-name* "-v" *version-date*
   where *version-date* is a full version-date of the form *n-yyyymmdd*. This form references the schema whose "root" name (typically the standardized prefix name used for the namespace that the schema represents) and version-date match *schema-root-name* and *version-date*, respectively.
2. *schema-root-name* "-v" *version*
   where *version* is an integer representing the namespace's version number. This form references the most recent version of the schema whose root name and namespace version number match *schema-root-name* and the *version*, respectively.

Usage rules for schema location URIs are as follows:

- All instance documents, whether generated by a service or a control point, MUST use Form 1.
- All UPnP AV published schemas that reference other UPnP AV schemas will also use Form 1.
- Validation of XML instance documents in UPnP AV systems potentially serves two purposes. The first is based on standard XML and XML Schema semantics: the document's creator asserts that the document is syntactically correct with respect to the referenced schema. The receiving processor can confirm this with a validating parser that uses the referenced schema(s). The second is based on UPnP AV namespace semantics. The receiving processor knows that the XML instance document is supposed to conform to one or more specific UPnP AV specifications. Since the second context is actually the more important context for instance document processing, the receiving processor MAY validate the instance document against any version of a schema that satisfies its needs in assessing the acceptability of the received instance document.

1.3.3 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 Document*. This document assumes version-date 2-20060531 of the "didl-lite:" namespace/schema combination and (a possible later) version 2-20061231 of "upnp:". The lines with the gray background show how to express this versioning information in the instance document.

```
<?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-v2-20060531.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp-v2-20061231.xsd">
   <item id="18" parentID="13" restricted="0">
      ...
   </item>
</DIDL-Lite>
```

Example 2:

Sample *srs XML Document*. This document assumes version 1-20060531 of the "srs:" namespace/schema combination. Again, the lines with the gray background show how to express this versioning information in the instance document.

```
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:srs
    http://www.upnp.org/schemas/av/srs-v1-20060531.xsd">
   ...
```

```
</srs>
```

1.4 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation MUST follow the naming conventions and XML rules as specified in [DEVICE], Section 2.5, “Description: Non-standard vendor extensions”.

1.5 References

This section lists the normative references used in the UPnP AV specifications and includes the tag inside square brackets that is used for each such reference:

[AVARCH] – *AVArchitecture:1*, UPnP Forum, June 25, 2002.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20020625.pdf.

[AVDT] – *AV DataStructure Template:1*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructure-v1-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructure-v1.pdf.

[AVDT-XSD] – *XML Schema for UPnP AV Datastructure Template:1*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/avdt-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/avdt-v1.xsd.

[AV-XSD] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/av-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/av-v1.xsd.

[AVS-XSD] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/avs-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/avs-v1.xsd.

[AVT] – *AVTransport:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service.pdf.

[AVT-EVENT-XSD] – *XML Schema for AVTransport:2 LastChange Eventing*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/avt-event-v2-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/avt-event-v2.xsd.

[CDS] – *ContentDirectory:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v2-Service-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v2-Service.pdf.

[CM] – *ConnectionManager:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service.pdf.

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

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

[DEVICE] – *UPnP Device Architecture, version 1.0*, UPnP Forum, June 13, 2000.
Available at: http://www.upnp.org/specs/architecture/UPnP-DeviceArchitecture-v1.0-20000613.htm.
Latest version available at: http://www.upnp.org/specs/architecture/UPnP-DeviceArchitecture-v1.0.htm.

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

[DIDL-LITE-XSD] – *XML Schema for ContentDirectory:2 Structure and Metadata (DIDL-Lite)*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/didl-lite-v2-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/didl-lite-v2.xsd.

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

[HTTP/1.1] – *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.

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

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

[ISO 8601] – *Data elements and interchange formats – Information interchange -- Representation of dates and times*, International Standards Organization, December 21, 2000.
Available at: ISO 8601:2000.

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

[MR] – *MediaRenderer:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v2-Device-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v2-Device.pdf.

[MS] – *MediaServer:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-MediaServer-v2-Device-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaServer-v2-Device.pdf.

[RCS] – *RenderingControl:2*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service.pdf.

[RCS-EVENT-XSD] –*XML Schema for RenderingControl:2 LastChange Eventing*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/rcs-event-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/rcs-event-v1.xsd.

[RFC 1738] – *IETF RFC 1738, Uniform Resource Locators (URL)*, Tim Berners-Lee, et. Al., December 1994.
Available at: http://www.ietf.org/rfc/rfc1738.txt.

[RFC 2119] – *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.

[RFC 2396] – *IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax*, Tim Berners-Lee, et al, 1998.
Available at: http://www.ietf.org/rfc/rfc2396.txt.

[RFC 3339] – *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.

[RTP] – *IETF RFC 1889, Realtime Transport Protocol (RTP)*, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, January 1996.
Available at: http://www.ietf.org/rfc/rfc1889.txt.

[RTSP] – *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.

[SRS] – *ScheduledRecording:1*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v1-Service-20060531.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v1-Service-20060531.pdf.

[SRS-XSD] – *XML Schema for ScheduledRecording:1 Metadata and Structure*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/srs-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/srs-v1.xsd.

[SRS-EVENT-XSD] – *XML Schema for ScheduledRecording:1 LastChange Eventing*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/srs-event-v1.xsd.

[UAX 15] – *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.

[UNICODE COLLATION] – *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.

[UPNP-XSD] – *XML Schema for ContentDirectory:2 Metadata*, UPnP Forum, May 31, 2006.
Available at: http://www.upnp.org/schemas/av/upnp-v2-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/upnp-v2.xsd.

[UTS 10] – *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.

[UTS 35] – *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.

[XML] – *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.

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

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

[XML-NMSP] – *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.

[XML SCHEMA-1] – *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.

[XML SCHEMA-2] – *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.

[XMLSCHEMA-XSD] – *XML Schema for XML Schema*.

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

2 Overview

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.

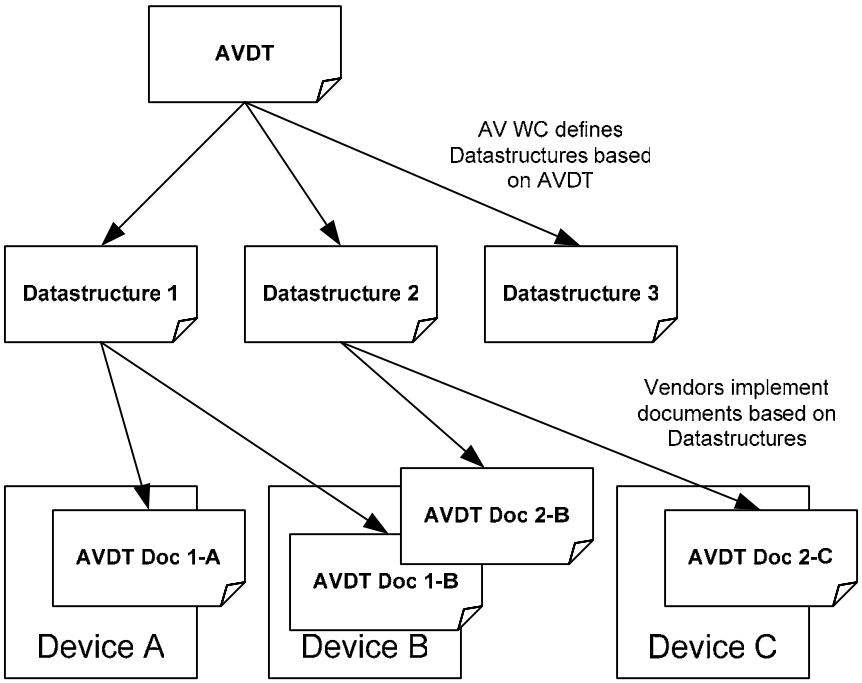

Figure 1: Typical Usage of AVDT

3 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-v1-20060531.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. Must 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 MUST 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 MUST be a QName as defined in section 3 of the W3C document "Namespaces in XML" [XML-NMSP]. Identical data structure types MUST be identified by the same name. Likewise, data structure types that are different MUST have different names.

fieldTable

REQUIRED. Begins the description section 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. MUST be one of the following formats:

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

where QName and NCName are defined in Section 3 of the W3C document "Namespaces in XML" [XML-NMSP]. For **field**s that correspond to an XML element (within the data structure's (**dataStructType**) XML document) **name** MUST 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** MUST 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**. MUST be a QName with a namespace prefix of "xsd". QName is defined in Section 3 of the W3C document "Namespaces in XML" [XML-NMSP]. MUST be one, and only one, of the data types defined by "XML Schema Part-2" [XML SCHEMA-2]. Contains the following attributes:

@csv

OPTIONAL. 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. MUST comply with the CSV data type notation identified in Section 1.2.1, "Comma Separated Value (CSV) Lists". 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. MUST 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

OPTIONAL. xsd:unsignedInt. Meaningful only when **datatype** equals "string". Indicates the maximum number of bytes allowed for this **field**. Note: 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

OPTIONAL. xsd:unsignedInt. Minimum number of occurrences of this **field** within the entire XML document. The default value is 0 which means this **field** is optional 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 MUST be present in every instance of this data structure at least the specified number of times.

maxCountTotal

OPTIONAL. xsd:string. Maximum number of occurrences of this **field** within the entire XML document. Its value MUST be either an unsigned integer or the value "UNBOUNDED". The default value is 1 which means this **field** MUST NOT be present more than once within any instance of this data structure (**dataStructType**). A value of 0 indicates that this **field** is prohibited and MUST 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** MUST be greater than or equal to **minCountTotal**.

minListSizeTotal

OPTIONAL. 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, MUST contain at least the specified number of entries in the CSV list.

maxListSizeTotal

OPTIONAL. 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 MUST be either a positive integer or the value "UNBOUNDED". The default value is 1 which means this CSV **field** MUST 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** MUST 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**. MUST contain either

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

Contains the following sub element(s):

dependentField

OPTIONAL. 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** MUST 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**
MUST contain either

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

Contains the following attributes and sub element(s):

@defaultDependency
OPTIONAL. 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**. MUST follow the format rules defined in the **name** sub element of **field**.

anyValue
OPTIONAL. 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 MUST be empty. **anyValue** MUST NOT be included along with the **valueList** or **valueRange** elements.

valueList
OPTIONAL. 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 MUST NOT be specified. MUST 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** MUST contain one of the allowed values defined within this **allowedValueDescriptor**.

valueRange
OPTIONAL. 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. MUST NOT be included along with the **anyValue** element. Contains the following sub elements:

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

step
OPTIONAL. xsd:string. Single positive numeric value. Indicates the numeric difference between adjacent supported values within the **valueRange**. The value of step MUST 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 must be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

minCount
OPTIONAL. 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** MUST 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 MUST be present at least the specified number of times. Additionally, each of those occurrences MUST be set to one of the values defined within this **allowedValueDescriptor**. Other instances of this **field** MAY occur but they MUST have a value defined within a different **allowedValueDescriptor**. For each **field**, the value of **minCount** MUST be less than or equal to **minCountTotal**.

maxCount
OPTIONAL. 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** MUST be either an unsigned integer or the value "UNBOUNDED". A value of 1 or greater indicates that this **field** MUST 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 MUST NOT include any occurrences of this **field** other than those occurences whose value is defined within a different **allowedValueDescriptor**. In this case, the **allowedValueDescriptor** MUST 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** MUST be greater than or equal to **minCount**. For each occurrence of **allowedValueDescriptor** the value of **maxCount** MUST be less than or equal to **maxCountTotal** for this **field**. Other instances of this

field MAY occur but they MUST have a value defined within a different **allowedValueDescriptor**.

minListSize

OPTIONAL. 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 MUST 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, MUST contain at least the specified number of entries whose value is defined within this **allowedValueDescriptor**. Other instances of this **field** MAY occur but they MUST have a value defined within a different **allowedValueDescriptor**.

maxListSize

OPTIONAL. 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** MUST be either an unsigned integer or the value "UNBOUNDED". The default value is 1 which means this CSV **field** MUST 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** MUST 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 MUST have a value defined within a different **allowedValueDescriptor**. The value of **maxListSize** MUST be greater than or equal to **minListSize** and less than or equal to **maxListSizeTotal**.

defaultValue

OPTIONAL. xsd:anyType. Identifies the default value assigned to this **field** if no value is present in the XML document. The contents MUST match the data type (**datatype**) of this **field** and it MUST 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 MUST contain the **defaultDependency** attribute with a value of 1.

allowAny

OPTIONAL. 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 MUST be empty. **allowAny** MUST NOT be included along with the **allowedValueList** or **allowedValueRange** elements.

allowedValueList

OPTIONAL. Enumerates a set of values that are allowed for this **field** subject to the contraints defined by the **dependentField** element, if present. Multiple **allowedValueLists** MUST NOT be specified. **allowedValueList** MUST 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 MUST 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: For a heterogeneous CSV-type **field**, it may not be practical to enumerate all of the allowed values that are possible. In this case, it is recommended to specify **allowAny**.

allowedValueRange

OPTIONAL. 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. MUST NOT be included along with the **allowAny** element. Contains the following sub elements:

minimum

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

step

OPTIONAL. xsd:string. Single positive numeric value. Indicates the numeric difference between adjacent valid values within the **allowedValueRange**. The value of step MUST 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** MUST be supported.
Note: Care must be taken when dealing with floating-point values so that conversion and/or rounding errors do not cause inaccurate comparison operations.

4 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 [AVDT-XSD]. 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 MUST 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 Section 1.3.2, “Namespace Names, Namespace Versioning and Schema Versioning”):

```
xsi:schemaLocation="
   urn:schemas-upnp-org:av:avdt
      http://www.upnp.org/schemas/av/avdt-v1-20060531.xsd"
```

where the number 1 after the “v” is the version number. Each AVDT schema version update must be backward compatible with the previous version. Specifically, XML elements and/or attributes may be added to more recent AVDT schema versions, but must 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.