
MediaServer:3 Device Template Version 1.01

For UPnP™ Version 1.0

Status: Standardized DCP

Date: September 30, 2008

Document Version: 1.0

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

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

Copyright © 2008, Contributing Members of the UPnP Forum. All rights Reserved.

Authors*	Company
Gary Langille	Echostar
John Ritchie (Co-Chair)	Intel
Julius Szakolczay	Matsushita Electric (Panasonic)
Keith Miller	Nokia
Vlad Stirbu	Nokia
Jeffrey Kang	Philips
Geert Knapen (Co-Chair)	Philips
Wouter van der Beek	Philips
Russell Berkoff	Pioneer
Wonseok Kwon	Samsung Electronics
SJae Oh	Samsung Electronics
Mahfuzur Raman	Samsung Electronics
Richard Bardini	Sony

***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	Overview and Scope	6
1.1	Introduction	6
1.2	Notation.....	7
1.2.1	Data Types	7
1.2.2	Strings Embedded in Other Strings.....	8
1.2.3	Extended Backus-Naur Form.....	8
1.3	Derived Data Types.....	9
1.3.1	Comma Separated Value (CSV) Lists.....	9
1.4	Management of XML Namespaces in Standardized DCPs.....	10
1.4.1	Namespace Prefix Requirements	13
1.4.2	Namespace Names, Namespace Versioning and Schema Versioning	13
1.4.3	Namespace Usage Examples.....	15
1.5	Vendor-defined Extensions	16
1.5.1	Vendor-defined Action Names	16
1.5.2	Vendor-defined State Variable Names.....	16
1.5.3	Vendor-defined XML Elements and attributes	16
1.5.4	Vendor-defined Property Names.....	16
1.6	References.....	17
2	Device Definitions.....	21
2.1	Device Type	21
2.2	Device Model.....	21
2.2.1	Description of Device Requirements	21
2.2.2	Relationships between Services	22
2.3	Theory of Operation.....	22
2.3.1	Device Discovery.....	22
2.3.2	Locating Desired Content	23
2.3.3	Preparing to Transfer the Content	23
2.3.4	Controlling the Transfer of the Content	23
2.3.5	Recording Content	23
3	XML Device Description.....	24
4	Test	26

List of Tables

Table 1-1:	EBNF Operators	8
Table 1-2:	CSV Examples	9
Table 1-3:	Namespace Definitions	11
Table 1-4:	Schema-related Information.....	12
Table 1-5:	Default Namespaces for the AV Specifications	13
Table 2-6:	Device Requirements	21

List of Figures

Figure 1: MediaServer Functional Diagram6

1 Overview and Scope

1.1 Introduction

This device specification is compliant with the UPnP™ Device Architecture version 1.0. It defines a device type referred to herein as MediaServer.

The MediaServer specification defines a general-purpose device that can be used to instantiate any Consumer Electronics (CE) device that provides AV content (for example, media) to other UPnP devices on the home network. It is based on the UPnP AV Architecture Framework (described in another document). It exposes its content via the ContentDirectory service (refer to the ContentDirectory service specification for details). The MediaServer MAY also provide functionality to record content using the ScheduledRecording service (refer to the ScheduledRecording service specification). As such, the MediaServer can handle any specific type of media, any data format, and transfer protocol.

Example instances of a MediaServer include traditional devices such as VCRs, CD Players, DVD Players, audio-tape players, still-image cameras, camcorders, radios, TV Tuners, and set-top boxes. Additional examples of a MediaServer also include new digital devices such as MP3 servers, PVRs, and Home MediaServers such as the PC. Although these devices contain diverse (AV) content in one form or another, the MediaServer (via the ContentDirectory service) is able to expose this content to the home network in a uniform and consistent manner. This ability allows the MediaServer to instantiate traditional single-function devices as well as more recent multi-function devices such as VCR-DVD players and the general purpose Home MediaServer, which contains a wide variety of content such as MPEG2 video, CD audio, MP3 and/or WMA audio, JPEG images, etc.

The MediaServer specification is very lightweight and can easily be implemented on low-resource devices such as still-image cameras or MP3 players that want to expose their local content to the home network. The MediaServer can also be used for high-end Home MediaServers that contain dozens of Gigabytes of heterogeneous content. Refer to the Theory Of Operation section for some specific examples of the MediaServer.

A full-featured MediaServer device provides clients with the following capabilities:

- Enumerate and query any of the content that the MediaServer can provide to the home network.
- Negotiate a common transfer protocol and data format between the MediaServer and target device.
- Control the flow of the content (for example, FF, REW, etc).
- Copy (import) content to the MediaServer from another device.
- Record content using the ScheduledRecording service.

This device specification does not provide:

- The ability to render AV content.

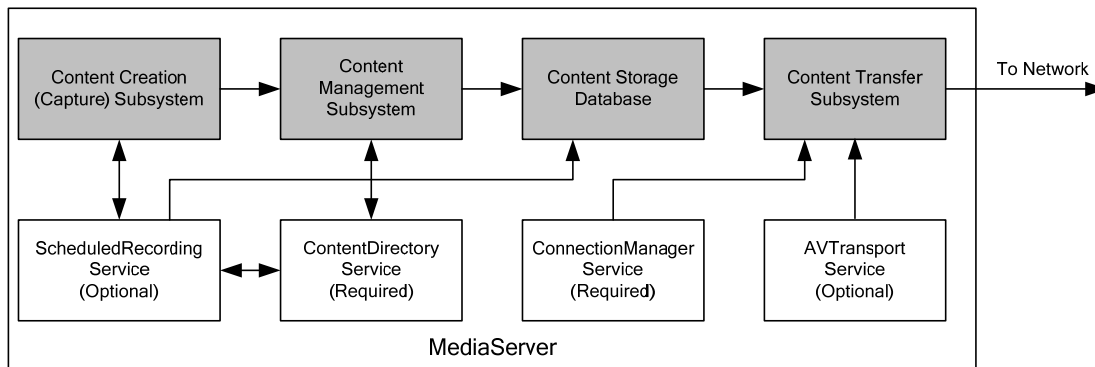


Figure 1: MediaServer Functional Diagram

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

1.2 Notation

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

The keywords “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 prohibited by 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 specification are printed using the *arch* character style [DEVICE].
- 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.

1.2.1 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 **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 **MUST** be accepted. Nevertheless, it is strongly **RECOMMENDED** that all Boolean properties be represented as “**0**” and “**1**”.

1.2.2 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 1.3.1, “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

1.2.3 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].

1.2.3.1 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.

1.3 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.

1.3.1 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 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 [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 1.2.2, “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 (<code>xsd:string</code>)	“+artist,-date”	List of 2 property sort criteria.
CSV (int) or CSV (<code>xsd:integer</code>)	“1,-5,006,0,+7”	List of 5 integers.
CSV (boolean) or CSV (<code>xsd:Boolean</code>)	“0,1,1,0”	List of 4 booleans

Type refinement of string	Value	Comments
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.

1.4 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. 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, MUST use the predefined namespace prefixes when referring to CDS properties (“upnp:”, “dc:”, etc).

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. For example, since the Scheduled Recording Service depends on and refers to the Content Directory 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 1-4, “Schema-related Information”, to cross-reference additional namespace information. This second table 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 Section 1.4.3 “Namespace Usage Examples” 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
<i>AV Working Committee defined namespaces</i>			
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]
cds-event	urn:schemas-upnp-org:av:cds-event	Evented LastChange state variable for ContentDirectory	[CDS]
didl-lite	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[CDS]
rscs-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]

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
<i>Externally defined namespaces</i>			
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 ¹ • Form 1, Form 2, Form 3	Valid Root Element(s)	Schema Reference
<i>AV Working Committee Defined Namespaces</i>			
av	<ul style="list-style-type: none"> • av-vn-yyyyymmdd.xsd • av-vn.xsd • av.xsd 	<i>n/a</i>	[AV-XSD]
avs	<ul style="list-style-type: none"> • avs-vn-yyyyymmdd.xsd • avs-vn.xsd • avs.xsd 	<Capabilities> <Features> <stateVariableValuePairs>	[AVS-XSD]
avdt	<ul style="list-style-type: none"> • avdt-vn-yyyyymmdd.xsd • avdt-vn.xsd • avdt.xsd 	<AVDT>	[AVDT]
avt-event	<ul style="list-style-type: none"> • avt-event-vn-yyyyymmdd.xsd • avt-event-vn.xsd • avt-event.xsd 	<Event>	[AVT-EVENT-XSD]
cds-event	<ul style="list-style-type: none"> • cds-event-vn-yyyyymmdd.xsd • cds-event-vn.xsd • cds-event.xsd 	<StateEvent>	[CDS-EVENT-XSD]
didl-lite	<ul style="list-style-type: none"> • didl-lite-vn-yyyyymmdd.xsd • didl-lite-vn.xsd • didl-lite.xsd 	<DIDL-Lite>	[DIDL-LITE-XSD]
racs-event	<ul style="list-style-type: none"> • rcs-event-vn-yyyyymmdd.xsd • rcs-event-vn.xsd • rcs-event.xsd 	<Event>	[RCS-EVENT-XSD]
srs:	<ul style="list-style-type: none"> • srs-vn-yyyyymmdd.xsd • srs-vn.xsd • srs.xsd 	<srs>	[SRS-XSD]
srs-event	<ul style="list-style-type: none"> • srs-event-vn-yyyyymmdd.xsd • srs-event-vn.xsd • srs-event.xsd 	<StateEvent>	[SRS-EVENT-XSD]
upnp	<ul style="list-style-type: none"> • upnp-vn-yyyyymmdd.xsd • upnp-vn.xsd • upnp.xsd 	<i>n/a</i>	[UPNP-XSD]

Standard Name-space Prefix	Relative URI and File Name ¹		Valid Root Element(s)	Schema Reference
	• Form 1, Form 2, Form3			
<i>Externally Defined Namespaces</i>				
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]

¹Absolute URIs are generated by prefixing the relative URIs with "<http://www.upnp.org/schemas/av/>".

1.4.1 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. 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. 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	avt-event
ConnectionManager	n/a
ContentDirectory	didl-lite
MediaRenderer	n/a
MediaServer	n/a
RenderingControl	rcs-event
ScheduledRecording	srs

1.4.2 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 must comply with one or more specific XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespaces that are defined by the AV Working Committee **MUST** be named by a URN. See Table 1-3, “Namespace

Definitions” 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 disambiguate (a.k.a. qualify) 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 Section 1.4.3 “Namespace Usage Examples”)

In order to enable both forward and backward compatibility, namespace names are permanently assigned and **MUST NOT** change even when a new version of a specification changes the definition of a namespace. However, all changes to a namespace definition **MUST** be backward-compatible. In other words, the updated definition of a namespace **MUST NOT** invalidate any XML documents that comply with an earlier definition of that same namespace. This means, for example, that a namespace **MUST NOT** be changed so that a new element or attribute is required. Although namespace names **MUST 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.

Each time 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 **MUST** 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 1-4, “Schema-related Information” identifies the URI formats for each of the namespaces that are currently defined by the UPnP AV Working Committee.

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

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

Form 2: “http://www.upnp.org/schemas/av/" *schema-root-name* “-v” *ver*

where *ver* is described above.

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

Form 2 of the undated schema URI is always linked to the most recent release of the schema that represents the version of the namespace indicated by *ver*. For example, the undated URI “.../av/rcs-event-v2.xsd” 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, MUST use Form 3.
- All UPnP AV published schemas that reference other UPnP AV schemas MUST 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 MUST contain a `version` attribute in the `<schema>` root element. Its value MUST correspond to the format:

ver “-” *yyymmdd* where *ver* and *yyymmdd* 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"`.

1.4.3 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>

```

1.5 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 below.

1.5.1 Vendor-defined Action Names

Vendor-defined action names MUST begin with “X”. Additionally, it SHOULD be followed by an ICANN assigned domain name owned by the vendor followed by the underscore character (“_”). It MUST then be followed by the vendor-assigned action name. The vendor-assigned action name MUST 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 MUST 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 MUST 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 MUST NOT be “XML” in any combination of case.

1.5.2 Vendor-defined State Variable Names

Vendor-defined state variable names MUST begin with “X”. Additionally, it SHOULD be followed by an ICANN assigned domain name owned by the vendor, followed by the underscore character (“_”). It MUST then be followed by the vendor-assigned state variable name. The vendor-assigned state variable name MUST NOT contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name MUST 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 MUST 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 MUST NOT be “XML” in any combination of case.

1.5.3 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 MUST be scoped by a vendor-owned XML namespace. Arbitrary XML MUST be enclosed in an element that begins with “X” and this element MUST 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”.

1.5.4 Vendor-defined Property Names

UPnP vendors MAY add non-standard properties to the ContentDirectory service. Each property addition MUST be scoped by a vendor-owned namespace. The vendor-assigned property name MUST NOT contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned property names are case sensitive. The first character of the name MUST 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 MUST 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 MUST NOT be “XML” in any combination of case.

1.6 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, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20080930.pdf>.

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

[AVDT] – *AV DataStructure Template:1*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVDataStructure-v1-20071231.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, 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-v1.xsd>.

[AV-XSD] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/av-v2-20080930.xsd>.

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

[AVS-XSD] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/avs-v2-20080930.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/avs-v2.xsd>.

[AVT] – *AVTransport:2*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service-20080930.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, 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-v1.xsd>.

[CDS] – *ContentDirectory:3*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v3-Service-20080930.pdf>.

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

[CDS-EVENT-XSD] – *XML Schema for ContentDirectory:3 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-v1.xsd>.

[CM] – *ConnectionManager:2*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service-20080930.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, July 20, 2006.

Available at: <http://www.upnp.org/specs/architecture/UPnP-DeviceArchitecture-v1.0-20060720.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:3 Structure and Metadata (DIDL-Lite)*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/didl-lite-v2-20080930.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](http://www.iso.org/iso/8601).

[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, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v2-Device-20071231.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v2-Device.pdf>.

[MS] – *MediaServer:3*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-MediaServer-v3-Device-20080930.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-AV-MediaServer-v3-Device.pdf>.

[RCS] – *RenderingControl:2*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service-20080930.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, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/rcs-event-v1-20080930.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 2045] – *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>.

[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:2*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service-20080930.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service.pdf>.

[SRS-XSD] – *XML Schema for ScheduledRecording:2 Metadata and Structure*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/srs-v2-20080930.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/srs-v2.xsd>.

[SRS-EVENT-XSD] – *XML Schema for ScheduledRecording:2 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-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:3 Metadata*, UPnP Forum, September 30, 2008.

Available at: <http://www.upnp.org/schemas/av/upnp-v3-20080930.xsd>.

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

[XPath20] – *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>.

[XQuery10] – *XQuery 1.0 An XML Query Language*. W3C Recommendation, 23 January 2007. Available at: <http://www.w3.org/TR/2007/REC-xquery-20070123>.

2 Device Definitions

2.1 Device Type

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

urn:schemas-upnp-org:device:MediaServer:3

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

2.2 Device Model

MediaServer products MUST implement minimum version numbers of all REQUIRED embedded devices and services specified in the table below. A MediaServer device can be either a Root device or can be Embedded in another UPnP device (MediaServer or other). A MediaServer device (Root or Embedded) can in turn contain other standard or non-standard Embedded UPnP devices.

Table 2-6: Device Requirements

DeviceType	Root	R/O ¹	ServiceType	R/O	Service ID ²
<u>MediaServer:3</u>	<u>Root</u> or <u>Embedded</u>	<u>R</u>	<u>ContentDirectory:3</u>	<u>R</u>	<u>ContentDirectory</u>
			<u>ConnectionManager:2</u>	<u>R</u>	<u>ConnectionManager</u>
			<u>AVTransport:2</u>	<u>O</u>	<u>AVTransport</u>
			<u>ScheduledRecording:2</u>	<u>O</u>	<u>ScheduledRecording</u>
			<i>Standard non-AV services defined by UPnP (QoS, Security, etc.) go here.</i>	<u>X</u>	<i>TBD</i>
			<i>Non-standard services embedded by a UPnP vendor go here.</i>	<u>X</u>	<i>TBD</i>
<u>Standard devices embedded by a UPnP vendor go here.</u>	<u>Embedded</u>	<u>O</u>	<u>Services as defined by the corresponding standard UPnP Device Definition go here.</u>		
<i>Non-standard devices embedded by a UPnP vendor go here.</i>	<u>Embedded</u>	<u>X</u>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>

¹ R = REQUIRED, O = OPTIONAL, X = Non-standard.

² Prefixed by urn:upnp-org:serviceId:

2.2.1 Description of Device Requirements

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

It should be noted that a MediaServer:3 implementation MUST respond to all SSDP queries that specify MediaServer:2 or lower and MUST respond to all actions defined by the MediaServer:2 or lower.

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

The existence of the AVTransport service depends on the transfer protocols that are supported by the device. The ConnectionManager specification includes a table that identifies which transfer protocols REQUIRE an AVTransport service to be implemented on the MediaServer. If an implementation of the MediaServer supports any of these transfer protocols, then it MUST implement the AVTransport service.

2.2.2 Relationships between Services

The [*ConnectionManager::PrepareForConnection\(\)*](#) action provides the trigger point for creating a new virtual instance of the AVTransport service (refer to the AVTransport service specification for a description of virtual instances of the AVTransport service). When a new connection is established (one that REQUIRES an AVTransport service on the MediaServer, which is determined by the selected transfer protocol), the [*ConnectionManager::PrepareForConnection\(\)*](#) action returns the [*InstanceID*](#) of the virtual instance of the AVTransport service that is bound to that connection. This virtual instance is used by the control point to control the flow (for example, [*AVTransport::Play\(\)*](#), [*AVTransport::Seek\(\)*](#), etc.) of the content to the network. As described in the AVTransport service specification, each virtual instance of the AVTransport service operates independently.

The [*ScheduledRecording::CreateRecordSchedule\(\)*](#) action may use objects (such as a User Channel item or an EPG item) exposed by the ContentDirectory service to specify what broadcast content will be recorded. The recorded content may then be exposed by the associated ContentDirectory service.

2.3 Theory of Operation

MediaServer devices are used in conjunction with one or more MediaRenderer devices to allow a control point to discover entertainment (AV) content (for example, video, music, images, etc) on the MediaServer and to render that content on any appropriate MediaRenderer within the home network. In general terms, the process begins with the control points discovering MediaServer and MediaRenderer devices within the home network. The control point interacts with a MediaServer(s) to locate a desired piece of content (for example, a movie, a song, a playlist, a photo album, etc). After the content has been identified, the control point needs to identify a common transfer protocol and data format that can be used to transfer the content from the MediaServer to the desired MediaRenderer. After these transfer parameters have been established, the control point controls the flow of the content (for example, [*AVTransport::Play\(\)*](#), [*AVTransport::Pause\(\)*](#), [*AVTransport::Stop\(\)*](#), [*AVTransport::Seek\(\)*](#), etc.) . (Depending on the selected transfer protocol, these flow control operations are sent either to the MediaServer or MediaRenderer, but not both). The actual transfer of the content is performed directly by the MediaServer and MediaRenderer. The content transfer happens independently from the control point and does not involve UPnP itself at all. The control point uses UPnP to setup the transfer of the content, but the transfer is performed using a transfer protocol other than UPnP. MediaServer devices also allow a control point to create a set of selection criteria to record content via the ScheduledRecording service. A control point can also discover recorded content on the MediaServer that was created by such a set of selection criteria.

2.3.1 Device Discovery

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

2.3.2 Locating Desired Content

Control points use the MediaServer's ContentDirectory service to locate desired content. The ContentDirectory service exposes both a search capability and a browse capability. Searching is useful when the control point (via the end-user) knows something about the content it wants to find (for example, its name, artist, type, date created, etc). Browsing is useful for blindly discovering what content the device has to offer. Each content item that is referenced by the ContentDirectory service includes various information about that content including the transfer protocol(s) and file format(s) that the MediaServer can use to transfer the content to the MediaRenderer.

2.3.3 Preparing to Transfer the Content

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

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

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

2.3.4 Controlling the Transfer of the Content

In all cases, the control point uses the [*InstanceID*](#), obtained as described above, to control the flow of the content. For example, to begin transferring the content, the control point invokes the [*AVTransport::Play\(\)*](#) action. To skip to a specific location within the content, the control point invokes the [*AVTransport::Seek\(\)*](#) action. In most cases, the choice of AVTransport actions that are actually invoked will likely be directed by the end-user while interacting with the control point's UI. Refer to the AVTransport service specification for additional details of these and other AVTransport actions.

2.3.5 Recording Content

A control point invokes the [*ScheduledRecording::CreateRecordSchedule\(\)*](#) action on a MediaServer device with a set of selection criteria (called a [*recordSchedule*](#)) that the device will use to identify the content to record. These criteria are then used to create individual objects (called [*recordTask*](#) objects), each containing the information necessary to create a single recording. For example, a [*recordSchedule*](#) might hold criteria that select every occurrence of a broadcast series for recording. The ScheduledRecording service then creates a [*recordTask*](#) instance for every episode in the series.

After creating one or more [*recordSchedule*](#) instances, the control point can browse the list of [*recordSchedule*](#) instances and their associated [*recordTask*](#) instances using the [*ScheduledRecording::BrowseRecordSchedules\(\)*](#) action.

3 XML Device Description

```

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>
      urn:schemas-upnp-org:device:MediaServer:3
    </deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>
          urn:schemas-upnp-org:service:ContentDirectory:3
        </serviceType>
        <serviceId>
          urn:upnp-org:serviceId:ContentDirectory
        </serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>
          urn:schemas-upnp-org:service:ConnectionManager:2
        </serviceType>
        <serviceId>
          urn:upnp-org:serviceId:ConnectionManager
        </serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
    </serviceList>
  </device>
</root>

```



```

<service>
  <serviceType>
    urn:schemas-upnp-org:service:AVTransport:2
  </serviceType>
  <serviceId>urn:upnp-org:serviceId:AVTransport</serviceId>
  <SCPDURL>URL to service description</SCPDURL>
  <controlURL>URL for control</controlURL>
  <eventSubURL>URL for eventing</eventSubURL>
</service>
<service>
  <serviceType>
    urn:schemas-upnp-org:service:ScheduledRecording:2
  </serviceType>
  <serviceId>
    urn:upnp-org:serviceId:ScheduledRecording
  </serviceId>
  <SCPDURL>URL to service description</SCPDURL>
  <controlURL>URL for control</controlURL>
  <eventSubURL>URL for eventing</eventSubURL>
</service>
  Declarations for standard non-AV services defined by UPnP
  (if any) go here
  Declarations for other services added by UPnP vendor
  (if any) go here
</serviceList>
<deviceList>
  Description of embedded devices added by UPnP vendor
  (if any) go here
</deviceList>
  <presentationURL>URL for presentation</presentationURL>
</device>
</root>

```

4 Test

There are no semantic tests defined for this device.