ScheduledRecording:1 Service Template Version 1.01

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

Authors	Company
Alan Presser	Allegrosoft
Gary Langille	Echostar
Gerrie Shults	HP
John Ritchie (Co-Chair)	Intel
Mark Walker	Intel
Nelson Kidd	Intel
Changhyun Kim	LG Electronics
Sungjoon Ahn	LG Electronics
Dennis Bushmitch	Matsushita Electric (Panasonic)
Minobu Abe	Matsushita Electric (Panasonic)
Masatomo Hori	Matsushita Electric (Panasonic)
Matthew Ma	Matsushita Electric (Panasonic)
Jack Unverferth	Microsoft
Wim Bronnenberg	Philips

Authors	Company
Geert Knapen (Co-Chair)	Philips
John Gildred	Pioneer
Elias Kesh	Pioneer
Russell Berkoff	Pioneer
Irene Shen	Pioneer
Norifumi Kikkawa	Sony
Jonathan Tourzan	Sony
Yasuhiro Morioka	Toshiba

Contents

List of Tables

List of Figures

1 Overview and Scope

This service definition is compliant with the UPnP Device Architecture version *1.0*. It defines a service type referred to herein as ScheduledRecording service.

1.1 Introduction

The ScheduledRecording service is a UPnP service that allows control points to schedule the recording of content. Generally, this content is broadcast content, but this specification does not limit itself to broadcast content. This service type enables the following functions:

- Create a *recordSchedule* so that it is added to the list of *recordSchedule* instances. Each *recordSchedule* describes user-level recording instructions for the ScheduledRecording service.
- Browse a list of *recordSchedule* instances stored by the ScheduledRecording service.
- Delete a *recordSchedule* so that it is removed from the list of *recordSchedule* instances.
- Browse a list of *recordTask* instances, stored by the ScheduledRecording service. The ScheduledRecording service may create zero or more *recordTask* instances for each *recordSchedule*. A *recordTask* represents a discrete recording operation of a *recordSchedule*.
- Enable or disable individual *recordTask* instances.
- Enable or disable a *recordSchedule*.
- Receive notifications indicating change of *recordSchedule* or *recordTask* list.

The ScheduledRecording service does not require a dependency on any UPnP services other than a co-located ContentDirectory service, which provides the following functions:

- A ContentDirectory service provides channel line-up to allow users to find recordable channels. A control point may use this metadata when creating a *recordSchedule* on a ScheduledRecording service.
- A ContentDirectory service may provide Electronic Program Guide (EPG) features to allow users to find recordable content. A control point may use this metadata when creating a *recordSchedule* on a ScheduledRecording service.
- Contents recorded by the ScheduledRecording service may be exposed by a ContentDirectory service.

The architectural relationship among the different concepts, defined by the ScheduledRecording service can be summarized as follows: A ScheduledRecording service owns a flat (that is: non-nested) list of *recordSchedule* instances, meaning that the ScheduledRecording service may create, destroy, or change *recordSchedule* instances. A *recordSchedule* represents user-level instructions to perform recording operations. Generally, a user constructs his instructions to a ScheduledRecording service via a control point that invokes UPnP actions that affect the list of *recordSchedule* instances. In all cases, the ScheduledRecording service MUST be able to describe discrete recording operations for a *recordSchedule* through a list of associated *recordTask* instances. A *recordTask* can only exist with a *recordSchedule* (that is: never orphaned). Thus when a *recordTask* is created by the ScheduledRecording service, its lifetime depends on its parent *recordSchedule*. An individual *recordTask* can be selectively enabled or disabled.

This service template does not address:

- Implementations where the ScheduledRecording service and its associated ContentDirectory service are not co-located in the same device.

1.2 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.

1.2.1 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 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*".

1.2.2 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 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 (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"

Type refinement of string	Value	Comments
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.

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]

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
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]
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]

1.4.1 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. 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:

1.4.2 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 1.4.3, "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.

1.4.3 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 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>
```

1.5 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 in [DEVICE], Section 2.5, "Description: Non-standard vendor extensions".

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, 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 Service Modeling Definitions

2.1 ServiceType

The following service type identifies a service that is compliant with this template:

urn:schemas-upnp-org:service: *ScheduledRecording:1*

ScheduledRecording service is used herein to refer to this service type.

2.2 Terms and Abbreviations

2.2.1 Abbreviations

Table 2-1: Abbreviations

Definition	Description
CDS	ContentDirectory Service.
EPG	Electronic Program Guide.
SRS	ScheduledRecording service.

2.2.2 Terms

2.2.2.1 CDS object

An object in a ContentDirectory service metadata hierarchy; that is: *item* or *container*.

2.2.2.2 User Channel

A User Channel is a ContentDirectory service object that exposes the (continuous) content stream of a particular broadcast channel. Usually, the actual channel that the User Channel exposes is determined by the user through some device-specific interaction. Examples are: manual programming of a number of channel presets; invoking of the auto-scan functionality of a device; predefined fixed channel assignments by the device manufacturer.

2.2.2.3 Channel Group

A Channel Group is a ContentDirectory service *container* that holds a number of User Channel items. Typically, a Channel Group contains User Channel items that are bound to a particular hardware resource. Examples include: a single analog cable TV tuner, a HDTV digital tuner, an AM/FM radio tuner, etc.

2.2.2.4 Channel Line-up

A service provider-generated list of channels with their associated content provider.

2.2.2.5 object

A *recordSchedule* or a *recordTask* (see definition of *recordSchedule* and *recordTask* below).

2.2.2.6 class

As defined in the ContentDirectory service specification, a class is used to assign a type to an object. It also identifies the minimum REQUIRED set of properties that MUST be present on that object and the OPTIONAL properties that MAY be present. Classes are organized in a hierarchy with certain classes being derived from others as in a typical object-oriented system. This specification defines two base classes (*recordSchedule* and *recordTask*) from which all other classes are derived.

2.2.2.7 object Modification

An object is considered modified when one of its properties (or its list of properties) is modified; that is: added, removed or changed in value (see definition of property below).

2.2.2.8 *recordSchedule*

A ScheduledRecording service construct that represents a complete set of recording instructions to the service, which allows the service to generate *recordTask* objects as necessary to record the desired content. The creator of the *recordSchedule* object assigns it a specific class, based on the type and complexity of the instructions, used to identify the content.

A *recordSchedule* is represented in XML as an `<item>...</item>` element.

2.2.2.9 Conflicting *recordSchedule*

A conflicting *recordSchedule* exists when one or more of its associated *recordTask* instances is in conflict with another *recordTask* instance.

2.2.2.10 *recordTask*

A ScheduledRecording service construct that represents a discrete recording operation of the underlying recording system. A *recordTask* is created by its parent *recordSchedule* and can not be directly created by the user. The parent-child relationship of *recordSchedule* and *recordTask* can be 1-to-zero or more.

A *recordTask* is represented in XML as an `<item>...</item>` element.

2.2.2.11 Conflicting *recordTask*

A conflicting *recordTask* exists when it overlaps in time with one or more other *recordTask* instances and the ScheduledRecording service has insufficient resources to record all of them. Existing pre-roll and post-roll adjustments (as defined by the *scheduledStartDateTimeAdjust* and *scheduledDurationAdjust* properties) MUST be taken into account when determining conflicts.

2.2.2.12 *recordScheduleParts*

A ScheduledRecording service construct that represents user-level recording instructions to the service, which provide a template to generate complete *recordSchedule* objects. The creator of the *recordScheduleParts* object assigns it a specific class, based on the type and complexity of the instructions, used to identify the content.

A *recordScheduleParts* is represented in XML as an `<item>...</item>` element.

2.2.2.13 Property-set Data Types

Certain ScheduledRecording service actions use *property-set* arguments that contain information about a set of properties, typically expressed in the form of an *srs XML Document* (for example, the *Elements* argument of the *CreateRecordSchedule()* action). The set of properties that can exist in a *property-set* argument is implementation dependent. Indeed, the set of optional properties that a particular ScheduledRecording service chooses to implement is vendor dependent.

This specification currently defines three different *property-set* data types:

- *A_ARG_TYPE_RecordSchedule*
- *A_ARG_TYPE_RecordTask*
- *A_ARG_TYPE_RecordScheduleParts*

Although these three types are different, they are very similar in nature and are defined using the same SRS schema [SRS-XSD], which defines all the properties that can ever occur in any of the three *property-set* data types. They differ only in the set of properties that can appear in them and in the values that are allowed for these properties.

2.2.2.14 Property

A property in the ScheduledRecording service represents a characteristic of an object. Properties are distinguished by their names. The ScheduledRecording service defines two kinds of properties – independent and dependent. Each independent property has zero or more dependent properties associated with it. Independent property names contain no "@" symbol; they may contain an XML namespace prefix (see below for an explanation of the relationship between properties and XML). Each dependent property is associated either with exactly one independent property or directly with a ScheduledRecording service class. The name of a dependent property that is associated with an independent property is the concatenation of three parts: its associated independent property name, the "@" symbol, and a name for the relationship between the two properties' values. The name of a dependent property that is associated directly with a class is just the "@" symbol followed by the relationship name. Their data types and meanings are defined in Appendix B, "AV Working Committee Extended Properties".

Even though ScheduledRecording service properties are not XML objects, XML is used to express them in all exchanges between a control point and a ScheduledRecording service implementation. This creates an unavoidable relationship between XML syntax and property names and values. In XML, an independent property is represented as an element. The property name is used as the element name. The property value is the element content. A dependent property is represented as an attribute in XML. The dependent property's relationship name is used as the attribute name. The dependent property's value is the attribute value. For dependent properties that are associated with an independent property, the attribute appears in the start tag of the element that represents its associated independent property. For dependent properties that are associated directly with a class, the attribute appears in the top-level start tag for each object of that class.

Examples:

Table 2-1: Properties in XML

Property Name	XML Representation (srs declared as default namespace)
title	`<title>…</title>`
taskProgramCode	`<taskProgramCode>…</taskProgramCode>`
taskProgramCode@type	`<taskProgramCode type="…">…</taskProgramCode>`
@id	`<item id="…">…</item>`

2.2.2.15 Member Property

A property is a member of a particular class when the property is declared to be either REQUIRED or OPTIONAL for that class.

2.2.2.16 Supported Member Property

A supported member property is a member property that is supported by a particular ScheduledRecording service implementation, according to the information returned by the *GetPropertyList()* action.

2.2.2.17 Multi-valued property

Some independent properties are multi-valued. This means that the property MAY occur more than once in an object.

2.2.2.18 Single-valued property

Most independent properties are single-valued. This means thatthe property MUST occur at most once in an object. Some single-valued properties can contain a CSV list of values. A dependent property is always considered single-valued, because it can occur at most once with each occurrence of its associated independent property, even though the independent property may be multi-valued.

2.2.2.19 XML Document

A string that represents a valid XML 1.0 document according to a specific schema. Every occurrence of the phrase "*XML Document*" is preceded by the appropriate root element name, italicized, as listed in column 3, "Valid Root Element(s)" of Table 1-4, "Schema-related Information".

For example, the phrase "*srs XML Document*" refers to an XML document based on the SRS Schema as defined in [SRS-XSD]. Such a document comprises a single `<srs …>` root element, optionally preceded by the XML declaration: `<?xml version="1.0" …?>`.

Therefore, the string containing the *srs XML Document* will have one of the following two forms:

"`<srs …>…</srs>`"

or

"`<?xml …?>`
`<srs …>…</srs>`"

2.2.2.20 XML Fragment

An *XML Fragment* is a sequence of XML elements that are valid direct or indirect child elements of the root element according to a specific schema. Every occurrence of the phrase "*XML Fragment*" is preceded by the appropriate root element name, italicized, as listed in column 3, "Valid Root Element(s)" of Table 1-4, "Schema-related Information".

For example, the phrase "*srs XML Fragment*" refers to a sequence of XML elements that are defined in the SRS Schema as defined in [SRS-XSD]:

"`<item id="…" …>…</item>`"
or

"`<recordDestination mediaType="…" preference="…">`
`…`
`</recordDestination>`"
or

"`<title>…</title>`
`<class>…</class>`
`<…>…</…>`
`…`
`<…>…</…>`"

2.2.2.21 actualScheduledStartDateTime

The actual scheduled start date&time of a program item is defined as:

actualScheduledStartDateTime = *scheduledStartDateTime* + *scheduledStartDateTimeAdjust*

where *scheduledStartDateTime* is the scheduled broadcast start date&time of the program item and *scheduledStartDateTimeAdjust* is a user-supplied adjustment to that date&time, for example for pre-roll purposes.

2.2.2.22 actualStartDateTime

The actual start date&time of a program item is defined as:

actualStartDateTime = *actualScheduledStartDateTime* + any device-specific record startup latency.

2.2.2.23 actualScheduledEndDateTime

The actual scheduled end time of a program item is defined as:

actualScheduledEndDateTime = *scheduledStartDateTime* + *scheduledDuration* + *scheduledDurationAdjust*

where *scheduledStartDateTime* is the scheduled broadcast start date&time of the program item, *scheduledDuration* is the scheduled broadcast duration of the program item and *scheduledDurationAdjust* is a user-supplied adjustment to that duration, for example to select just a part of the program for recording.

2.2.2.24 actualEndDateTime

The actual end date&time of a program item is defined as:

actualEndDateTime = *actualScheduledEndDateTime* + any device-specific record teardown latency.

2.2.2.25 actualScheduledDuration

The actual scheduled duration of a program item is defined as:

actualScheduledDuration = *actualScheduledEndDateTime* – *actualScheduledStartDateTime*

= *scheduledDuration* + *scheduledDurationAdjust* – *scheduledStartDateTimeAdjust*

where *scheduledDuration* is the scheduled broadcast duration of the program item, *scheduledDurationAdjust* is a user-supplied adjustment to that duration, and *scheduledStartDateTimeAdjust* is a user-supplied adjustment to the scheduled start date&time.

2.2.2.26 Lexical Sort Order

Lexical sort order refers to string sorting – also called collation – based on language and regional conventions. It is *not* based on the binary codes of the characters in strings. Furthermore, lexical sorting is not based on character sets; a single character set may have multiple sort orders, again according to language and regional conventions. It is also possible to have lexical sorts that are further refined according to user preference. For a complete discussion of this topic see [UTS 10], and the related standards [UAX 15] and [UTS 35]. [UTS 10] defines the lexical sort algorithms. It uses a secondary algorithm defined in [UAX 15] and supporting data tables defined in [UTS 35]. These three references together – [UAX 15], [UTS 10] and [UTS 35] – should be sufficient to implement a robust lexical sort.

Simple example: one of the most familiar examples is case-insensitive sorting on the ASCII subset of Unicode. In a binary ASCII sort, all lower case letters sort after the upper case "Z" because "Z" has a character code of 0x5A, and all lower case character codes are greater than or equal to 0x61.

More complex example: the “ö” character in German sorts between “n” and “p” characters whereas in Swedish, it sorts after “z”.

2.2.2.27 Lexical Matching

Lexical matching compares two (sub)strings for equality under certain lexical sorting conditions. *It is important to note* that equality in lexical matching is often less restrictive than equality in lexical sorting. In other words, two strings that are equal under a lexical sort will always be a lexical match. However, two strings that are a lexical match might not be equal under a lexical sort for the same language and region. In some cases, an implementation’s lexical sort might consider all alphabetic characters with diacritical marks (accents, umlauts, circumflexes, etc.) to be distinct, yet the same implementation might ignore diacritical marks in lexical matching. For example, the strings “resumé”, “resume” and “résumé” might sort as “resume” < “resumé” < “résumé”, but when a lexical match using the string “resume”, might find all three strings “resumé”, “resume” and “résumé”. For implementation techniques, see [UTS 10] Section 8, “Searching and Matching”.

2.2.2.28 Simple Non-case-sensitive Sort Order

A simple non-case-sensitive sort order applies only to Roman alphabetic characters. All lower case ASCII alphabetic characters MUST sort the same as their uppercase equivalent, except when compared directly with their upper case equivalent, in which case the upper case character SHOULD sort before its lower case equivalent. This means that of the following three ordering relations, #1 MUST be true, at least one of #2 and #3 MUST be true, and #2 SHOULD be true.

“A” ≤ “a” < “B” ≤ “b” < … < “Y” ≤ “y” < “Z” ≤ “z”

“A” < “a” < “B” < “b” < … < “Y” < “y” < “Z” < “z”

“A” = “a” < “B” = “b” < … < “Y” = “y” < “Z” = “z”

Additionally, the same upper and lower case relationships SHOULD hold for non-ASCII Roman alphabetic characters. That is, lower case alphabetic characters with diacritical marks SHOULD sort as their upper case equivalent, except when compared directly with their upper case equivalent, in which case the upper case character should sort before its lower case equivalent. The ordering relation between ASCII and non-ASCII alphabetic characters is left unspecified. Also, the ordering relation between non-ASCII alphabetic characters that are not upper or lower case equivalents of each other is left unspecified. This may be summarized in the following relations. In each, the letter “c” represents any non-ASCII Roman alphabetic character. #4 SHOULD be true for all “c”. #5 SHOULD be true for all “c”. If #5 is false for any “c”, it should be false for all “c” and #6 SHOULD be true for all “c”.

upper(c) ≤ lower(c)

upper(c) < lower(c)

upper(c) = lower(c)

2.2.2.29 Simple Non-case-sensitive Matching

In a simple non-case-sensitive match, relation #0 above MUST be true, and relation #0 above SHOULD be true.

2.2.2.30 Numeric Sort Order

A sort order in which values are compared numerically. If the type of an individual value is numeric, the numeric value is used. If the type of an individual value is string, the string is converted to a number and that numeric value is used.

2.2.2.31 Boolean Sort Order

Boolean values are sorted with “0” (false) being less than “1” (true).

2.2.2.32 Sequenced Sort

A sequenced sort is a sort applied to a set of values, each of which is composed of a sequence of subvalues. The sequence is often in a CSV list, but there are other kinds of sequences used for sorting in this specification. The sequenced sort starts by sorting based on the first subvalue in the sequence. If all values differ in the first subvalue, the sort is finished. Otherwise, each subset of equal subvalues is then sorted based on the next subvalue in the sequence. This process repeats iteratively until there are no more subsets of equal subvalues or the sequence is exhausted.

2.2.2.33 Sequenced Lexical Sort

A sequenced sort in which all subvalues are strings and the subvalues are compared lexically.

2.2.2.34 Sequenced Numeric Sort

A sequenced sort in which each subvalue is either a number or the number represented by a string.

2.2.2.35 Lexical Numeric Sort

A lexical numeric sort is one where one or more substrings are known to represent numbers. The strings are then sorted using a sequenced sort, where the sequence is composed of the sequence of non-numeric and numeric substrings from the larger string.

For example, assume a property has the form <letter>-<number>, where <number> ranges from 1 to 10. In a straight ascending lexical sort, the values “A-10”, “A-1”, “A-2” would sort as: “A-1”, “A-10”, “A-2”. “A-10” sorts before “A-2” because they are equal in the first two character positions, but in the third position, “1” < “2”. However, in a lexical numeric sort, each string is considered to be a sequence of a letter and number separated by a hyphen. These values then sort as “A-1”, “A-2”, “A-10” because all three are equal in the first subvalue, “A”, but the second subvalue sorts as 1, 2, 10 in numeric order.

2.2.2.36 *type* Relationship Sort

This is a sort defined exclusively for independent properties that have a dependent property relationship named “*type*”. These properties are sorted as a sequence of two subvalues: the first subvalue is the value of the property’s *xxx@type* dependent property; the second subvalue is the value of the independent property *xxx* itself. The *xxx@type* subvalues are sorted as specified for the dependent *xxx@type* property in its own section. The independent property subvalues are sorted according to the order specified in its section. Sorting of the independent property may vary with the value of the dependent property.

2.3 ScheduledRecording Service Architecture

2.3.1 *recordSchedule*

A ScheduledRecording service implementation has a single, flat list of *recordSchedule* instances. A *recordSchedule* represents the user-level recording instructions to the ScheduledRecording service. These user-level instructions have various levels of complexity. For example, a simple instruction may state: “record channel 15 at 4PM on March 19, 2004,” while a more complex instruction may state: “record all episodes of the *DIY Home Improvement Show* on any channel that has the show for the next month.” The behavior of a *recordSchedule* is described by one or more properties, and these properties can be manipulated through several actions.

As shown in Figure 1, when a control point requests a new scheduled recording to the ScheduledRecording service via the *CreateRecordSchedule()* action, the control point sets a number of properties and passes them to the ScheduledRecording service to express user-desired instructions to the scheduled recording. Then, as a response to the *CreateRecordSchedule()* action, the ScheduledRecording service creates a *recordSchedule*, assigns a unique ID to the *recordSchedule* and returns the *recordSchedule* with the complete set of initial property settings. The ScheduledRecording service MUST add OPTIONAL properties to the *recordSchedule* when a control point did not specify them. Additionally, the ScheduledRecording service MAY add some informative properties.

If a control point specifies unsupported or unknown properties as input to the *CreateRecordSchedule()* action, the ScheduledRecording service MUST gracefully ignore these. A control point can always parse the generated *recordSchedule* returned in the *Result* argument of the *CreateRecordSchedule* action to verify whether certain properties were rejected by the ScheduledRecording service. If unsupported values are set for supported properties, the ScheduledRecording service MUST return an error and the *recordSchedule* MUST NOT be created.

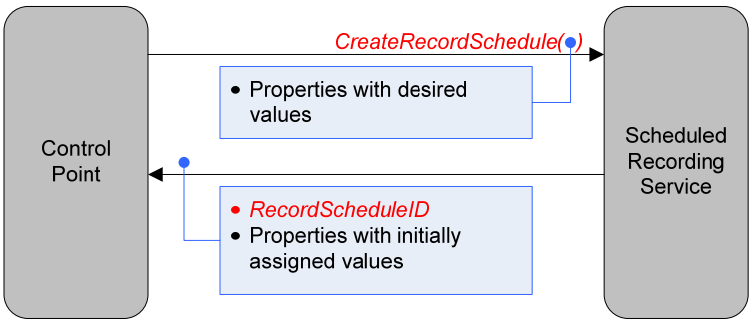

Figure 1: Creating a new *recordSchedule*

Some properties are defined as optional in the ScheduledRecording service. Therefore, a control point needs to determine which properties a ScheduledRecording service implementation actually supports. Since support levels and allowed values for properties can be different for *recordScheduleParts*, *recordSchedule* or *recordTask* usage, a pair of actions (*GetPropertyList()* and *GetAllowedValues()*) are provided to retrieve the relevant information. Figure 2 illustrates the concept.

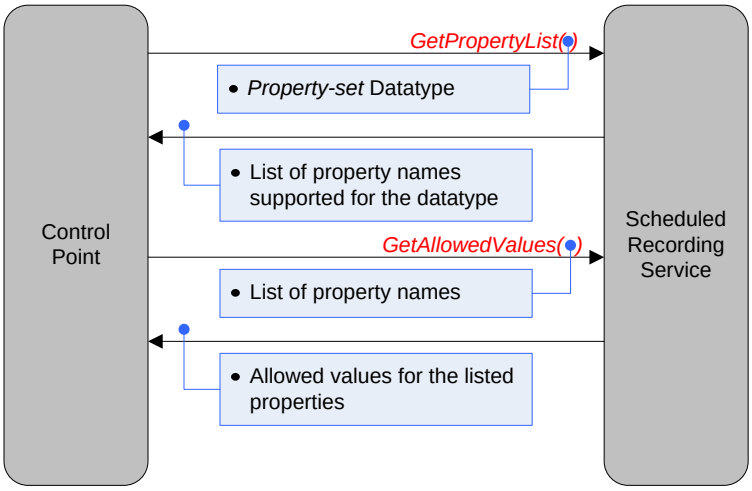

Figure 2: Capability check.

Figure 3 illustrates how *recordSchedule* instances can be browsed by the control point after they have been created, to retrieve the updated/current values of the properties.

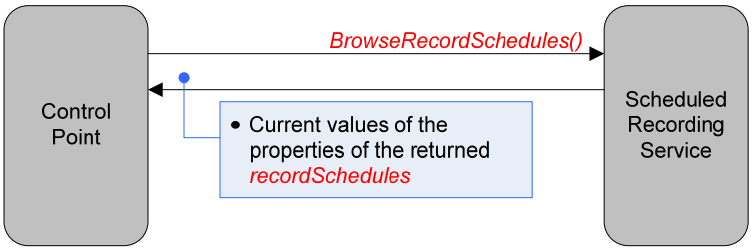

Figure 3: Browse *recordSchedule*.

Figure 4 illustrates how a control point can delete a *recordSchedule* from the ScheduledRecording service.

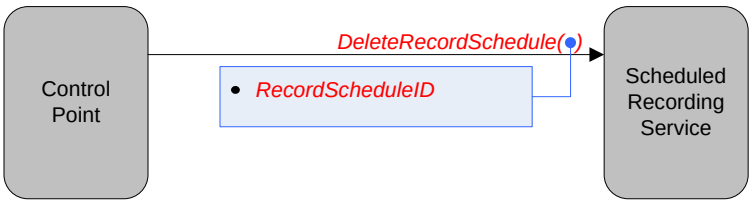

Figure 4: Delete a *recordSchedule*

2.3.2 *recordTask*

A *recordSchedule* will generate one *recordTask* for each recording operation that matches the criteria of the *recordSchedule*. A *recordTask* also has properties indicating its behavior. A *recordTask* is different from a *recordSchedule* in that it always represents a single recording operation whereas a *recordSchedule* may actually represent multiple recording operations. For example, a ScheduledRecording service that interprets a *recordSchedule* to lead to three different recording operations could generate three different *recordTask* instances over its lifetime. At a given time, a *recordSchedule* can have zero (no recording operations currently scheduled) or more *recordTask* instances associated with it. A ScheduledRecording service MUST report at least one *recordTask* when the underlying system is performing a recording operation for some *recordSchedule*.

When a *recordSchedule* is created, the ScheduledRecording service generates necessary *recordTask* instances associated with each scheduled recording occurrence. The ScheduledRecording service may also later add a new *recordTask* whenever a new scheduled recording occurrence arrives. Similarly, a ScheduledRecording service may delete *recordTask* instances when they are no longer needed. This MAY happen in a device dependent manner. For example, some ScheduledRecording service implementations delete a *recordTask* when the recording is finished while other ScheduledRecording service implementations keep maintaining finished *recordTask* instances. A *recordTask* can only be created by the ScheduledRecording service as a result of a *trigger* from a *recordSchedule*. A control point can never create a *recordTask* directly. Both a *recordTask* and a *recordSchedule* MAY be deleted by the ScheduledRecording service or a control point.

The lifetime of a *recordTask* is determined in a vendor dependent way. Some implementations maintain a *recordTask* even after it finishes its recording while others may delete the *recordTask* once the recording finishes. However, in any implementation, when a *recordSchedule* is deleted, the ScheduledRecording service MUST delete all of its associated *recordTask* instances.

2.4 State Variables

Like the ContentDirectory service, the ScheduledRecording service is primarily action-based. The service state variables exist primarily to support argument passing within service actions. Information is not exposed directly through explicit state variables. Instead, a client retrieves ScheduledRecording service

information via the return arguments of the actions defined below. The majority of state variables defined below exist simply to provide data type information for the arguments of the various actions of this service.

Reader Note: For a first-time reader, it may be more helpful to read the action definitions before reading the state variable definitions.

2.4.1 State Variable Overview

Table 2-2: State Variables

Variable Name	R/O[1]	Data Type	Allowed Value	Default Value	Eng. Units
SortCapabilities	*R*	**string**	CSV (**string**)		
SortLevelCapability	*R*	**ui4**			
StateUpdateID	*R*	**ui4**			
LastChange	*R*	**string**			
A_ARG_TYPE_PropertyList	*R*	**string**	CSV (**string**)		
A_ARG_TYPE_DataTypeID	*R*	**string**	See Table 2-3		
A_ARG_TYPE_ObjectID	*R*	**string**			
A_ARG_TYPE_ObjectIDList	*O*[2]	**string**	CSV (**string**)		
A_ARG_TYPE_PropertyInfo	*R*	**string**			
A_ARG_TYPE_Index	*R*	**ui4**			
A_ARG_TYPE_Count	*R*	**ui4**			
A_ARG_TYPE_SortCriteria	*R*	**string**	CSV (**string**)		
A_ARG_TYPE_RecordSchedule	*R*	**string**			
A_ARG_TYPE_RecordTask	*R*	**string**			
A_ARG_TYPE_RecordScheduleParts	*R*	**string**			

[1] *R* = Required, *O* = Optional, *X* = Non-standard.

[2] CONDITIONALLY REQUIRED. This argument type variable is REQUIRED when the *GetRecordScheduleConflicts()* or *GetRecordTaskConflicts()* actions are implemented. See Sections 2.6.17, "*GetRecordScheduleConflicts()*" and 2.6.18, "*GetRecordTaskConflicts()*" to determine when these actions MUST be implemented.

Table 2-3: allowedValueList for the *DataTypeID* argument

Value	R/O[1]
"*A_ARG_TYPE_RecordSchedule*"	*R*
"*A_ARG_TYPE_RecordTask*"	*R*
"*A_ARG_TYPE_RecordScheduleParts*"	*R*
Vendor-defined	*X*

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

2.4.2 *SortCapabilities*

This state variable contains a CSV list of property names that the ScheduledRecording service can use to sort the information returned in the *Result* argument of various actions, such as *BrowseRecordSchedules()* and *BrowseRecordTasks()*. An empty string indicates that the device does not support any kind of sorting. A wildcard "`srs:*`" indicates that any supported property within the srs namespace can be used for sorting.

2.4.3 *SortLevelCapability*

This state variable contains an integer that indicates the maximum number of property names that can be specified in the *SortCriteria* argument at the same time.

2.4.4 *StateUpdateID*

This state variable is a ScheduledRecording service system-wide numeric value. Its initial value is 0.

- *StateUpdateID* MUST be incremented by 1 whenever any of the following occurs:
  - A *recordSchedule* or *recordTask* is created or deleted.
  - A *recordSchedule* or *recordTask* is modified, which means that one or more properties are added, deleted or had their value changed.
  - Any other change to the state of the ScheduledRecording service that could be observed by a control point. This includes any vendor- or other future-defined behavior.
- When the value of *StateUpdateID* is equal to the **ui4** maximum value of 4294967295 (2^{32}-1), incrementing it causes it to roll over to the value 0.
- The increment and the operation that caused it must occur atomically relative to all information visible to any control point – including both action out arguments and evented variable values.

  For example, consider the case where a control point invokes *CreateRecordSchedule()* to create a new *recordSchedule* that also immediately spawns exactly one *recordTask*. Assume that *StateUpdateID* is 10 when the control point invokes the action and that for a short time period around this invocation, no other activity occurs that affects the value of *StateUpdateID*. During this time period, exactly one of the following MUST be true as seen by all external observations (including the returned values from this *CreateRecordSchedule()* invocation):
  - *StateUpdateID* is 10; *and* the new *recordSchedule* has not been created; *and* the new *recordTask* has not been created.
  - *StateUpdateID* is 11; *and* the *recordSchedule* has been created; *and* the new *recordTask* has not been created; *and* the *recordSchedule*'s value of *currentRecordTaskCount* is 0, indicating that no *recordTask* has been created.
  - *StateUpdateID* is 12; *and* the *recordSchedule* has been created; *and* the new *recordTask* has been created; *and* the *recordSchedule*'s value of *currentRecordTaskCount* is 1, indicating that the child *recordTask* has been created.

ScheduledRecording service implementations SHOULD maintain the same value for *StateUpdateID* through power cycles and any other disappearance/reappearance of the service on the network. Control points can use a change in the value of this variable to determine if there has been a change in the ScheduledRecording service.

The value of the *StateUpdateID* state variable, returned within events and returned as an output argument of certain actions should be monitored very closely by control points. Indeed, whenever an action returns with a *StateUpdateID* value in its *UpdateID* argument that is less than the *StateUpdateID* value received in the *updateID* attribute from the most recent *LastChange* event, the information returned by that action is potentially stale. A control point may want to refresh that information for instance by invoking the appropriate *Browsexxx()* or *Getxxx()* action. It is safe to use the information as long as the *StateUpdateID* value returned in the *UpdateID* argument of the action is greater than or equal to the *StateUpdateID* value received in the *updateID* attribute from the most recent *LastChange* event.

2.4.5 *LastChange*

Note: It is assumed that the default namespace for this sub-section (2.4.5, "*LastChange*") *of the specification is srs-lc.*

This state variable is used for eventing purposes to allow clients to receive meaningful event notifications whenever a *recordSchedule* or *recordTask* in the ScheduledRecording service changes. [SRS-EVENT-XSD] defines the schema for the *StateEvent XML Document* used in this state variable. The optional XML header `<?xml version="1.0" ?>` is allowed. One root element, `<StateEvent>` MAY have zero or more elements, each of which represent one update to a *recordSchedule* or *recordTask* instance. Six types of update elements are defined as shown in Table 2-4, "Allowed Elements in `<StateEvent>` Element". Future ScheduledRecording service specifications MAY add other types of update elements. A vendor MAY add vendor-defined elements. The ScheduledRecording:1 service does not define the value for these elements. Vendor-defined element names MUST follow the rules set forth in Section 1.5, "Vendor-defined Extensions". Note that future ScheduledRecording service specifications MAY define sub-elements for the elements. Also note that this state variable MUST be properly escaped as defined in [XML].

Table 2-4: Allowed Elements in `<StateEvent>` Element

Element Name	Description
`RecordScheduleCreated`	A new *recordSchedule* is created.
`RecordScheduleModified`	One or more properties of a *recordSchedule* are modified (added, deleted or values are changed).
`RecordScheduleDeleted`	A *recordSchedule* is deleted.
`RecordTaskCreated`	A new *recordTask* is created.
`RecordTaskModified`	One or more properties of a *recordTask* are modified (added, deleted or values are changed).
`RecordTaskDeleted`	A *recordTask* is deleted.
Vendor-defined	See Section 1.5, "Vendor-defined Extensions".

Each element MUST have one *updateID* attribute, which is set to the value of the *StateUpdateID* state variable at the time of the update and one *objectID* attribute, whose value is set to the value of the *@id* property of the updated *recordSchedule* or *recordTask* instance. Future ScheduledRecording service specifications MAY add other attributes to existing update elements. A vendor MAY add vendor-defined attributes for existing update elements.

Example (before XML escaping)

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

```
   urn:schemas-upnp-org:av:srs-event
    http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd">
   <RecordScheduleCreated updateID="213" objectID="s001"/>
   <RecordTaskCreated updateID="214" objectID="s001-001"/>
   <RecordTaskModified updateID="215" objectID="s001-001"/>
</StateEvent>
```

The *LastChange* state variable is evented and moderated. When multiple updates occurred within a *LastChange* moderation period, the new *LastChange* state variable reports more than one update at the same time. A series of updates and the resulting eventing activity are illustrated in their temporal order in the following example.

Example

0: ScheduledRecording service activity = Power-on.

StateUpdateID = 0

LastChange (before XML escaping):

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

GENA behavior: None

1: ScheduledRecording service activity = a *recordSchedule* with *@id* = "*s001*" is created.

StateUpdateID = 1

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:srs-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:srs-event
    http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd">
   <RecordScheduleCreated updateID="1" objectID="s001">
   </RecordScheduleCreated>
</StateEvent>
```

GENA behavior: Nothing is evented since there are no current subscribers.

2: ScheduledRecording service activity = new control point signs up for events.

StateUpdateID = 1

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:srs-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:srs-event
    http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd">
   <RecordScheduleCreated updateID="1" objectID="s001">
   </RecordScheduleCreated>
```

```
</StateEvent>
```

GENA behavior: Send initial Notify with the *LastChange* value above.

3: **ScheduledRecording service activity** = a *recordTask* with *@id* = "*t001-000*" is created. Its associated *recordSchedule* with *@id* = "*s001*" is modified by the ScheduledRecording service at the same time because its *currentReordTaskCount* property is updated to reflect the existence of the new *recordTask*.

StateUpdateID = 3

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:srs-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:srs-event
    http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd">
   <RecordTaskCreated updateID="2" objectID="t001-000">
   </RecordTaskCreated>
   <RecordScheduleModified updateID="3" objectID="s001">
   </RecordScheduleModified>
</StateEvent>
```

GENA behavior: Wait for the next moderation period to elapse and then send Notify with the *LastChange* value above.

4: **ScheduledRecording service activity** = a *recordTask* with *@id* = "*t001-001*" is created. Its associated *recordSchedule* with *@id* = "*s001*" is modified by the ScheduledRecording service at the same time because its *currentReordTaskCount* property is updated to reflect the existence of the new *recordTask*. Within the same moderation period, a *recordTask* with *@id* = "*t001-002*" is also created. Its associated *recordSchedule* with *@id* = "*s001*" is modified by the ScheduledRecording service at the same time because its *currentReordTaskCount* property is updated to reflect the existence of the new *recordTask*.

StateUpdateID = 7

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:srs-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:srs-event
    http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd">
   <RecordTaskCreated updateID="4" objectID="t001-001">
   </RecordTaskCreated>
   <RecordScheduleModified updateID="5" objectID="s001">
   </RecordScheduleModified>
   <RecordTaskCreated updateID="6" objectID="t001-002">
   </RecordTaskCreated>
   <RecordScheduleModified updateID="7" objectID="s001">
   </RecordScheduleModified>
</StateEvent>
```

GENA behavior: Wait for the next moderation period to elapse and then send Notify with the *LastChange* value above.

2.4.6 *A_ARG_TYPE_PropertyList*

This state variable is introduced to provide type information for various action arguments that contain a CSV list of property names. Namespace prefixes MUST be included with all property names (see Section

1.4, "Management of XML Namespaces"). The exact semantics of these property names depend on the associated action.

2.4.7 A_ARG_TYPE_DataTypeID

This state variable is introduced to provide type information for various action arguments that are used to identify a specific *property-set* data type (see Section 2.2.2.13, "Property-set Data Types"). An argument of type *A_ARG_TYPE_DataTypeID* can have the values listed in Table 2-3, "allowedValueList for the *DataTypeID* argument".

2.4.8 A_ARG_TYPE_ObjectID

This state variable is introduced to provide type information for various action arguments that uniquely identify an individual *recordSchedule* or a *recordTask* by their object ID.

2.4.9 A_ARG_TYPE_ObjectIDList

This state variable is introduced to provide type information for various action arguments that contain a CSV list of object IDs (*@id*) used to identify a collection of either *recordSchedule* or *recordTask* instances (the list MUST be homogeneous).

2.4.10 A_ARG_TYPE_PropertyInfo

This state variable is introduced to provide type information for various action arguments that contain detailed XML-based information on supported properties and their interdependencies for a particular ScheduledRecording service implementation. The format of these arguments is similar to the XML Service Description (SCPD), but instead of describing state variables and actions, they describe properties, their allowed values, and interdependencies.

Refer to [AVDT] for the definition of the AVDT Datastructure Template.

Note that since the format of these arguments is based on XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.11 A_ARG_TYPE_Index

This state variable is introduced to provide type information for various action arguments that specify an offset into an arbitrary set of objects. A value of 0 represents the first object in the set.

2.4.12 A_ARG_TYPE_Count

This state variable is introduced to provide type information for various action arguments that specify a number of arbitrary objects.

2.4.13 A_ARG_TYPE_SortCriteria

This state variable is introduced to provide type information for various action arguments that contain a CSV list of property names prefixed by one or more sort modifiers. Namespace prefixes MUST be included with all property names that do not belong to the srs namespace. Namespace prefixes MAY be included with property names that belong to the srs namespace (see Section 1.4, "Management of XML Namespaces"). The "+" and "-" sort modifier prefixes indicate that the sort is in ascending or descending order, respectively, with regard to the value of the prefixed property name.

2.4.14 A_ARG_TYPE_RecordSchedule

This state variable is introduced to provide type information for various action arguments that contain a list of zero or more *recordSchedule* objects. All instances of this data type MUST comply with the SRS schema. See Appendix A, "*srs XML Document*" for details.

The structure of an argument of data type *A_ARG_TYPE_RecordSchedule* is an *srs XML Document*:

- Optional XML declaration `<?xml version="1.0" ?>`
- `<srs>` is the root element.
- The `<srs>` element MUST have zero or more `<item>` elements, each representing a *recordSchedule* object.
- Each `<item>` element has a set of property values describing the *recordSchedule* object. Each property is expressed either as the content of an XML element or as the value of an XML attribute.
- See [SRS-XSD] for more details on the structure. The ScheduledRecording service-defined names for metadata are described in Appendix B, "AV Working Committee Extended Properties."

Note that since the SRS format of an argument of data type *A_ARG_TYPE_RecordSchedule* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.15 A_ARG_TYPE_RecordTask

This state variable is introduced to provide type information for various action arguments that contain a list of zero or more *recordTask* objects. All instances of this data type MUST comply with the SRS schema. See Appendix A, "*srs XML Document*" for details.

The structure of an argument of data type *A_ARG_TYPE_RecordTask* is an *srs XML Document*:

- Optional XML declaration `<?xml version="1.0" ?>`
- `<srs>` is the root element.
- The `<srs>` element MUST have zero or more `<item>` elements, each representing a *recordTask* object.
- Each `<item>` element has a set of property values describing the *recordTask* object. Each property is expressed either as the content of an XML element or as the value of an XML attribute.
- See [SRS-XSD] for more details on the structure. The ScheduledRecording service-defined names for metadata are described in Appendix B, "AV Working Committee Extended Properties."

Note that since the SRS format of an argument of data type *A_ARG_TYPE_RecordTask* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.16 A_ARG_TYPE_RecordScheduleParts

This state variable is introduced to provide type information for various action arguments that contain a single *recordScheduleParts* object. A *recordScheduleParts* object indicates the desired values for a subset of properties that provide a template for other *recordSchedule* objects. Typically, a *recordScheduleParts* is used to create new *recordSchedule* objects. All instances of this data type MUST comply with the SRS schema. See Appendix A, "*srs XML Document*" for details.

The structure of an argument of data type *A_ARG_TYPE_RecordScheduleParts* is an *srs XML Document*:

- Optional XML declaration `<?xml version="1.0" ?>`
- `<srs>` is the root element.

- The <srs> element MUST have a single <item> element, representing the *recordScheduleParts* object.
- The <item> element has a set of property values describing the *recordScheduleParts* object. Each property is expressed either as the content of an XML element or as the value of an XML attribute.
- See [SRS-XSD] for more details on the structure. The ScheduledRecording service-defined names for metadata are described in Appendix B, "AV Working Committee Extended Properties."

Note that since the SRS format of an argument of data type *A_ARG_TYPE_RecordScheduleParts* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.5 Eventing and Moderation

Table 2-5: Eventing and Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
SortCapabilities	*NO*	*NO*			
SortLevelCapability	*NO*	*NO*			
StateUpdateID	*NO*	*NO*			
LastChange	*YES*	*YES*	0.2 seconds		
A_ARG_TYPE_PropertyList	*NO*	*NO*			
A_ARG_TYPE_DataTypeID	*NO*	*NO*			
A_ARG_TYPE_ObjectID	*NO*	*NO*			
A_ARG_TYPE_ObjectIDList	*NO*	*NO*			
A_ARG_TYPE_PropertyInfo	*NO*	*NO*			
A_ARG_TYPE_Index	*NO*	*NO*			
A_ARG_TYPE_Count	*NO*	*NO*			
A_ARG_TYPE_SortCriteria	*NO*	*NO*			
A_ARG_TYPE_RecordSchedule	*NO*	*NO*			
A_ARG_TYPE_RecordTask	*NO*	*NO*			
A_ARG_TYPE_RecordScheduleParts	*NO*	*NO*			

[1] Determined by N, where Rate = (Event)/(N secs).

[2] (N) * (allowedValueRange Step).

2.6 Actions

Table 2-6: Actions

Name	R/O[1]
GetSortCapabilities()	*R*

Name	R/O[1]
GetPropertyList()	*R*
GetAllowedValues()	*R*
GetStateUpdateID()	*R*
BrowseRecordSchedules()	*R*
BrowseRecordTasks()	*R*
CreateRecordSchedule()	*R*
DeleteRecordSchedule()	*R*
GetRecordSchedule()	*R*
EnableRecordSchedule()	*O*[2]
DisableRecordSchedule()	*O*[2]
DeleteRecordTask()	*O*
GetRecordTask()	*R*
EnableRecordTask()	*O*[3]
DisableRecordTask()	*O*[3]
ResetRecordTask()	*O*[3]
GetRecordScheduleConflicts()	*O*[4]
GetRecordTaskConflicts()	*O*[5]

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

[2] CONDITIONALLY REQUIRED. The *EnableRecordSchedule()* and *DisableRecordSchedule()* actions MUST be implemented as a combination. If one action is implemented, then the other action MUST also be implemented.

[3] CONDITIONALLY REQUIRED. The *EnableRecordTask()*, *DisableRecordTask()*, and *ResetRecordTask()* actions MUST be implemented as a combination. If one action is implemented, then the other actions MUST also be implemented.

[4] CONDITIONALLY REQUIRED. See Section 2.6.17, "*GetRecordScheduleConflicts()*" to determine when this action MUST be implemented.

[5] CONDITIONALLY REQUIRED. See Section 2.6.18, "*GetRecordTaskConflicts()*" to determine when this action MUST be implemented.

2.6.1 *GetSortCapabilities()*

This action returns a CSV list of property names that can be used in the *SortCriteria* argument of various actions.

2.6.1.1 Arguments

Table 2-7: Arguments for *GetSortCapabilities()*

Argument	Direction	relatedStateVariable
SortCaps	*OUT*	*SortCapabilities*
SortLevelCap	*OUT*	*SortLevelCapability*

2.6.1.1.1 *SortCaps*

This argument contains a CSV list of property names that the ScheduledRecording service can use to sort the information returned in the *Result* argument of various actions, such as *BrowseRecordSchedules()* and *BrowseRecordTasks()*. The appropriate namespace prefixes (either "`srs:`" or "`<vendor-defined namespace prefix>:`") MUST be included with the returned property names (see Section 1.4, "Management of XML Namespaces"). An empty string indicates that the device does not support any kind of sorting. A wildcard "`srs:*`" indicates that any property within the srs namespace can be used for sorting. See also Section 2.4.2, "*SortCapabilities*"

2.6.1.1.2 *SortLevelCap*

This argument contains an integer that indicates the maximum number of property names that can be specified at the same time in the *SortCriteria* argument of various actions. See also Section 2.4.3, "*SortLevelCapability*."

2.6.1.2 Dependency on State

None.

2.6.1.3 Effect on State

None.

2.6.1.4 Errors

Table 2-8: Error Codes for *GetSortCapabilities()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.2 *GetPropertyList()*

The *GetPropertyList()* action provides a means to retrieve from a particular ScheduledRecording service implementation which properties are actually supported for a specific *property-set* data type. The *GetPropertyList()* action returns a CSV list of property names that may appear in action arguments of the *property-set* data type, specified in the *DataTypeID* input argument. This CSV list MUST include property names of imported properties from other namespaces as well as any vendor-defined property names. For example, the ContentDirectory service imported properties (such as *dc:title*) that are included as part of the value of the *cdsReference* property, MUST be returned.

The appropriate namespace prefixes MUST be included with *all* property names (see Section 1.4, "Management of XML Namespaces").

The set of allowed values for srs properties and vendor-defined properties (when used for the specified *property-set* data type) can be obtained via the *GetAllowedValues()* action. The set of allowed values for imported properties cannot be retrieved by the *GetAllowedValues()* action.

2.6.2.1 Arguments

Table 2-9: Arguments for *GetPropertyList()*

Argument	Direction	relatedStateVariable
DataTypeID	*IN*	*A_ARG_TYPE_DataTypeID*
PropertyList	*OUT*	*A_ARG_TYPE_PropertyList*

2.6.2.1.1 *DataTypeID*

The *DataTypeID* argument identifies the *property-set* data type for which the set of property names is to be returned. See Section 2.4.7, "*A_ARG_TYPE_DataTypeID*" for details regarding its format. The set of allowed values is listed in Table 2-3, "allowedValueList for the *DataTypeID* argument".

2.6.2.1.2 *PropertyList*

The *PropertyList* argument contains the set of property names (including their namespace prefixes) that may appear in action arguments of the *property-set* data type, specified by the *DataTypeID* input argument.

2.6.2.2 Dependency on State

None.

2.6.2.3 Effect on State

None.

2.6.2.4 Errors

Table 2-10: Error Codes for *GetPropertyList()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
711	Invalid DataTypeID	An invalid value has been specified in the *DataTypeID* input argument.

2.6.3 *GetAllowedValues()*

This action is used to determine the allowed values and dependencies for srs properties that can appear within action arguments of the specified *property-set* data type. The set of allowed values that are returned is static and does not depend on the current state of the ScheduledRecording service. The property information is returned in an *AVDT XML Document* as defined in [AVDT]. The set of properties for which information is returned is determined by the intersection of the property names in the *Filter* argument and the names of the properties supported by the implementation for the specified *property-set* data type in the *DataTypeID* argument. All property names MUST belong either to the srs namespace or a vendor-defined namespace.

The set of allowed values for imported properties cannot be retrieved by the *GetAllowedValues()* action.

2.6.3.1 Arguments

Table 2-11: Arguments for *GetAllowedValues()*

Argument	Direction	relatedStateVariable
DataTypeID	*IN*	*A_ARG_TYPE_DataTypeID*
Filter	*IN*	*A_ARG_TYPE_PropertyList*
PropertyInfo	*OUT*	*A_ARG_TYPE_PropertyInfo*

2.6.3.1.1 *DataTypeID*

See Section 2.6.2.1.1, "*DataTypeID*".

2.6.3.1.2 *Filter*

The *Filter* argument contains a CSV list of property names that indicates for which properties allowed value information is to be returned in the *AVDT XML Document*, contained in the *PropertyInfo* output argument. The *Filter* argument SHOULD only include property names that are returned in the *PropertyList* argument of the *GetPropertyList()* action when specifying the same value in the *DataTypeID* argument. ScheduledRecording service implementations MUST gracefully ignore other property names. The "`srs:`" namespace prefix MUST be included with srs property names in the *Filter* argument. Likewise, a namespace prefix MUST be included with all vendor-defined property names in the *Filter* argument (see Section 1.4, "Management of XML Namespaces").

If the *Filter* argument is set to "`*:*`", then allowed values for all supported properties (including srs properties and vendor-defined properties, but excluding imported properties) for the specified *property-set* data type MUST be returned. If the *Filter* argument is set to "`srs:*`", then allowed values for all supported properties in the srs namespace MUST be returned. If the *Filter* argument is set to "`<vendor-defined namespace prefix>:*`", then allowed values for all vendor-defined properties in that namespace MUST be returned. If the *Filter* argument is set to the empty string, no information is provided (an *AVDT XML Document* with an empty root element is returned).

Examples of valid *Filter* argument values include:

- "`srs:@id,srs:priority@orderedValue`"
- "`srs:title,srs:class`"
- "`*:*`"
- "`srs:*`"

2.6.3.1.3 *PropertyInfo*

The *PropertyInfo* argument MUST only include allowed value and dependency information on properties that are specified in the *Filter* argument. The *PropertyInfo* argument MUST be properly escaped as defined in [XML]. The particular *AVDT XML Document* that is returned in the *PropertyInfo* argument depends on the *property-set* data type, specified in the *DataTypeID* input argument. See Appendix A, "*srs XML Document*" for further details.

2.6.3.2 Dependency on State

None.

2.6.3.3 Effect on State

None.

2.6.3.4 Errors

Table 2-12: Error Codes for *GetAllowedValues()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
711	Invalid DataTypeID	An invalid value has been specified in the *DataTypeID* input argument.

2.6.4 *GetStateUpdateID()*

This action returns the current value of the *StateUpdateID* state variable in the *Id* output argument. This action can be used to poll the ScheduledRecording service for any change in the service that might have occurred since the last time this action was invoked. If the returned *Id* value is different from the value that was returned the last time this action was invoked, then there has been a change in one or more *recordSchedule* or *recordTask* objects in the ScheduledRecording service. See Section 2.4.4, "*StateUpdateID*" for more information.

2.6.4.1 Arguments

Table 2-13: Arguments for *GetStateUpdateID()*

Argument	Direction	Related State Variable
Id	*OUT*	*StateUpdateID*

2.6.4.1.1 *Id*

The *Id* argument contains the current value of the *StateUpdateID* state variable.

2.6.4.2 Dependency on State

None.

2.6.4.3 Effect on State

None.

2.6.4.4 Errors

Table 2-14: Error Codes for *GetStateUpdateID()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.5 *BrowseRecordSchedules()*

This action is used to browse the set of *recordSchedule* objects in the ScheduledRecording service.

2.6.5.1 Arguments

Table 2-15: Arguments for *BrowseRecordSchedules()*

Argument	Direction	relatedStateVariable
Filter	*IN*	*A_ARG_TYPE_PropertyList*
StartingIndex	*IN*	*A_ARG_TYPE_Index*
RequestedCount	*IN*	*A_ARG_TYPE_Count*
SortCriteria	*IN*	*A_ARG_TYPE_SortCriteria*
Result	*OUT*	*A_ARG_TYPE_RecordSchedule*
NumberReturned	*OUT*	*A_ARG_TYPE_Count*
TotalMatches	*OUT*	*A_ARG_TYPE_Count*
UpdateID	*OUT*	*StateUpdateID*

2.6.5.1.1 *Filter*

The *Filter* argument contains a CSV list of property names that indicates which properties are to be returned in the *srs XML Document*, contained in the *Result* output argument. Namespace prefixes MUST be included with all property names, specified in the *Filter* argument (see Section 1.4, "Management of XML Namespaces").

The *Filter* argument has no impact on the *number* of objects returned in the *Result* argument. Instead, the *Filter* argument allows control points to control the *complexity* of the object metadata that is returned in the *srs XML Document* for each object. It allows a control point to specify a subset of the supported properties for inclusion in the *srs XML Document*. Properties that are REQUIRED by the SRS Schema MUST always be returned. Compliant ScheduledRecording service implementations MUST NOT return optional properties unless they are explicitly requested in the *Filter* input argument or are needed to create a valid XML document. For example, specifying a dependent property in the *Filter* argument, such as *priority@orderedValue*, will cause its associated independent property, *priority*, to be included in the *srs XML Document*.

In all cases, a compliant ScheduledRecording service implementation MUST always respond to query requests with the smallest, valid *srs XML Document* in the *Result* argument that satisfies the *Filter* input argument. If the *Filter* argument is set to the empty string (""), then only the REQUIRED properties are returned.

If the *Filter* argument is equal to "`*:*`", then all supported properties for all supported namespaces MUST be returned. If the *Filter* argument is equal to "`<namespace prefix>:*`", then all of the REQUIRED srs properties and all of the supported properties within that single namespace MUST be returned. For example, "`srs:*`" is equivalent to listing all srs namespace properties supported by the device.

Properties defined in the ContentDirectory service MUST only be imported through the multi-valued *cdsReference* property. Therefore, if the *Filter* argument contains property names from namespaces defined in the ContentDirectory service specification, the appropriate *cdsReference* property values MUST be included in the *Result* output argument and those values MUST be filtered, according to what is specified in the *Filter* argument but also preserving the validity of the *DIDL-Lite XML Document*, returned in the *cdsReference* property.

Examples of valid *Filter* argument values include:

- "srs:@id,srs:priority@orderedValue"
- "srs:title,dc:title"
- "*:*"
- "upnp:*,dc:*,didl_lite:*"

A compliant ScheduledRecording service implementation MUST also ignore optional properties requested in the *Filter* input argument which are not actually present in the matching objects. For example, a *BrowseRecordSchedules()* *Filter* input argument of the form "srs:activePeriod" is successful and returns a *Result* value that complies with the other *BrowseRecordSchedules()* input arguments, even in the case where the objects represented in the *Result* argument do not have an *activePeriod* property defined.

2.6.5.1.2 *StartingIndex* and *RequestedCount*

This action returns a specified number of *recordSchedule* objects from the list as indicated by the *RequestedCount* argument and starting from a specified index in the list, as indicated by the *StartingIndex* argument. The first *recordSchedule* in the list MUST be indexed by an index value of 0. Specifying 0 in the *RequestedCount* argument is PROHIBITED. If the range indicated by the *StartingIndex* and *RequestedCount* arguments reaches beyond the end of the list, then the ScheduledRecording service MUST return all *recordSchedule* objects up to the end of the list and starting from the specified *StartingIndex*.

2.6.5.1.3 *SortCriteria*

The order of the *recordSchedule* objects in the *Result* argument is determined by the *SortCriteria* argument. When an empty string is specified in the *SortCriteria* argument, then the order is device dependent. Additionally, this device dependent ordering MUST remain constant unless the *UpdateID* argument value has changed since the last *BrowseRecordSchedules()* action. In other words, any two objects that appear in a *Result* argument MUST always appear in the same relative order as long as the *UpdateID* argument value (and therefore the *StateUpdateID* state variable) did not change.

The *SortCriteria* argument contains a CSV list of property names (namespace prefixes MUST always be included). Each property name MUST be prefixed by either a "+" or a "-" sort modifier. The "+" and "-" modifiers indicate that the sort is in ascending or descending order, respectively, with regard to the value of its associated property.

The ScheduledRecording service MUST NOT accept any property name in the *SortCriteria* argument that is not included in the *SortCapabilities* state variable.

The objects are first sorted on the value of the first property in the *SortCriteria* argument. If all values differ in the first property, the sort is finished. If any values of the first property are equal, each subset of equal values is then sorted based on the next property in the *SortCriteria* argument. This process repeats iteratively until there are no more subsets of equal values or the *SortCriteria* argument list is exhausted.

For example, a value for the *SortCriteria* argument of the *BrowseRecordSchedules()* action of:

"+srs:scheduledStartDateTime,-srs:scheduledChannelID,+srs:matchingName"

would sort the returned *recordSchedule* instances first by start date&time in ascending order, then for each date&time, the instances would be sorted by descending channel ID and finally, for each channel ID, the instances would be sorted by ascending program name.

Sorting rules for each property depend on that property's semantics. Sorts for individual properties can be any of: numeric sort, lexical sort, lexical numeric sort, Boolean sort, sequenced sort, *type* relationship sort, or *property specific*, according to an explicit ordering of values defined individually for that property. The definition of each kind of sort may be found in Section 2.2.2.26, "Lexical Sort Order". The specific sort order rules that MUST be used for each property are given in Appendix B, "AV Working Committee Extended Properties".

When a *SortCriteria* argument contains property names of optional and/or multi-valued properties, the following rules apply:

If the property is prefixed by "+" then:

- Objects that do not have a value for the property are returned first in their group.
- Objects that have at least one value for the property are returned next in their group. Objects that have multiple values for the property (either multi-valued or CSV list) are sorted based on the property value that would cause the object to appear earliest in the list.

If the property is prefixed by "-" then:

- Objects that have at least one value for the property are returned first in their group. Objects that have multiple values (either multi-valued or CSV list) for the property are sorted based on the property value that would cause the object to appear earliest in the list.
- Objects that do not have a value for the property are returned last in their group.

Example:

Assume a ScheduledRecording service contains the following items and the current date is Tuesday, June 21, 2005:

```
<item id="1">
   …
   <scheduledStartDateTime>2006-02-07T15:30:00</ScheduledStartDateTime>
   …
</item>
<item id="2">
   …
   <scheduledStartDateTime>MONT15:30:00</ScheduledStartDateTime>
   <scheduledStartDateTime>WEDT15:30:00</ScheduledStartDateTime>
   …
</item>
<item id="3">
   …
   <scheduledStartDateTime>MON-FRIT16:00:00</ScheduledStartDateTime>
   …
</item>
<item id="4">
   …
   No <scheduledStartDateTime> property
   …
</item>
```

A value for the *SortCriteria* argument of the *BrowseRecordSchedules()* action of:

"`+srs:scheduledStartDateTime`"

would sort the returned *recordSchedule* instances on Tuesday, June 21, 2005 as follows:

```
<item id="4"/>
<item id="2"/>
<item id="3"/>
<item id="1"/>
```

because:

- `<item id="4"/>` has no *srs:scheduledStartDateTime* property, it therefore appears first.

- `<item id="2"/>` *srs:scheduledStartDateTime* property resolves to Wednesday, 2005-06-22T15:30:00 since this is the earliest date&time in the list. It therefore appears second.
- `<item id="3"/>` *srs:scheduledStartDateTime* property resolves to Wednesday, 2005-06-22T16:00:00. It therefore appears third.
- `<item id="1"/>` *srs:scheduledStartDateTime* property resolves to Tuesday, 2006-02-07T15:30:00. It therefore appears last.

Sorting on ContentDirectory service imported properties is not supported.

2.6.5.1.4 *Result*

The *Result* output argument contains an XML escaped *srs XML Document* (see [SRS-XSD]). This document contains a set of zero or more *recordSchedule* objects as described in Appendix A, "*srs XML Document*". Each of the returned *recordSchedule* objects MUST NOT have properties other than those specified in the *Filter* argument unless they are needed to create a valid *srs XML Document*. The ScheduledRecording service implementation MUST ignore unknown properties specified in the *Filter* argument. If "*:*" is specified in the *Filter* argument, then all supported properties for which the ScheduledRecording service has meaningful values MUST be returned. The REQUIRED properties (for example, *@id*, *title*, *class*, …) MUST always be included even if not specified in the *Filter* argument (the *srs XML Document* MUST be valid). The ScheduledRecording service implementation MUST ensure that the information returned in this argument is always consistent. In other words, if during the information gathering process, certain updates occur, the ScheduledRecording service implementation MUST re-examine the already gathered information to verify that this information is still accurate before returning from the action invocation.

2.6.5.1.5 *NumberReturned*

The *NumberReturned* argument MUST indicate the actual number of returned objects.

2.6.5.1.6 *TotalMatches*

The *TotalMatches* argument MUST indicate the total number of *recordSchedule* objects that exist in the ScheduledRecording service.

2.6.5.1.7 *UpdateID*

The returned *UpdateID* argument MUST be the value of the *StateUpdateID* state variable at the time the returned data has been completely and consistently collected. In other words, if during the information gathering process, certain updates occur, the ScheduledRecording service implementation MUST re-examine the already gathered information to verify that this information is still accurate before returning from the action invocation. Refer to Section 2.4.4, "*StateUpdateID*" for additional information.

The *UpdateID* argument is used to verify whether the returned information in the *Result* argument has not become stale. After the action completes, if the value of the *StateUpdateID* state variable is different from the value returned in the *UpdateID* argument, then the information returned in the *Result* argument may be stale. In this case, the control point should invoke the appropriate action to refresh its copy of the desired information (for example, via the *BrowseRecordSchedules()* or *GetRecordSchedule()* action).

2.6.5.2 Dependency on State

None.

2.6.5.3 Effect on State

None.

2.6.5.4 Errors

Table 2-16: Error Codes for *BrowseRecordSchedules()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
709	Unsupported or invalid sort criteria	The sort criteria specified are not supported or are invalid.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.6 *BrowseRecordTasks()*

This action is used to browse the list of *recordTask* objects associated with a single *recordSchedule*. In addition, it can be used to browse the entire list of all *recordTask* objects available in the entire ScheduledRecording service, independent of their parent *recordSchedule*.

The *Result* argument contains an XML escaped *srs XML Document* that contains a set of *recordTask* objects. When the *RecordScheduleID* input argument contains the *@id* value of an existing *recordSchedule*, then the *Result* argument returns an XML escaped *srs XML Document* that contains the set of *recordTask* objects associated with that particular *recordSchedule*. When the *RecordScheduleID* input argument is set to the empty string (“”), then the *Result* argument returns an XML escaped *srs XML Document* that contains a list of all available *recordTask* objects in the entire ScheduledRecording service.

2.6.6.1 Arguments

Table 2-17: Arguments for *BrowseRecordTasks()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*
Filter	*IN*	*A_ARG_TYPE_PropertyList*
StartingIndex	*IN*	*A_ARG_TYPE_Index*
RequestedCount	*IN*	*A_ARG_TYPE_Count*
SortCriteria	*IN*	*A_ARG_TYPE_SortCriteria*
Result	*OUT*	*A_ARG_TYPE_RecordTask*
NumberReturned	*OUT*	*A_ARG_TYPE_Count*
TotalMatches	*OUT*	*A_ARG_TYPE_Count*
UpdateID	*OUT*	*StateUpdateID*

The syntax and semantics of the arguments (the *RecordScheduleID* argument not included) of the *BrowseRecordTasks()* action are identical to those of the *BrowseRecordSchedules()* action, except that the objects returned by this action are *recordTask* objects instead of *recordSchedule* objects.

2.6.6.1.1 *RecordScheduleID*

The *RecordScheduleID* input argument contains the object ID of the *recordSchedule* for which all associated *recordTask* instances are returned in the *Result* argument. If the *RecordScheduleID* input argument contains the empty string (“”), then all available *recordTask* instances in the entire ScheduledRecording service are returned.

2.6.6.1.2 *Filter*

See Section 2.6.5.1.1, “*Filter*”.

2.6.6.1.3 *StartingIndex* and *RequestedCount*

See Section 2.6.5.1.2, “*StartingIndex* and *RequestedCount*”.

2.6.6.1.4 *SortCriteria*

See Section 2.6.5.1.3, “*SortCriteria*”.

2.6.6.1.5 *Result*

See Section 2.6.5.1.4, “*Result*”. However, the returned objects are *recordTask* objects instead of *recordSchedule* objects.

2.6.6.1.6 *NumberReturned*

See Section 2.6.5.1.5, “*NumberReturned*”.

2.6.6.1.7 *TotalMatches*

When the *RecordScheduleID* input argument contains the *@id* value of an existing *recordSchedule*, then the *TotalMatches* argument MUST indicate the total number of *recordTask* objects that exist in the ScheduledRecording service for the indicated *recordSchedule*. When the *RecordScheduleID* input argument is set to the empty string (“”), then the *TotalMatches* argument MUST indicate the total number of *recordTask* objects that exist in the entire ScheduledRecording service, independent of their parent *recordSchedule*.

2.6.6.1.8 *UpdateID*

See Section 2.6.5.1.7, “*UpdateID*”.

2.6.6.2 Dependency on State

None.

2.6.6.3 Effect on State

None.

2.6.6.4 Errors

Table 2-18: Error Codes for *BrowseRecordTasks()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.

ErrorCode	errorDescription	Description
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
709	Unsupported or invalid sort criteria	The sort criteria specified is not supported or is invalid.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.7 *CreateRecordSchedule()*

This action creates a *recordSchedule* (that is: a scheduled recording list entry) for some media content (for example, broadcast content, analog input content, etc). This action creates a new object of a class, derived from the *recordSchedule* class. Control points that want to schedule a recording invoke the *CreateRecordSchedule()* action.

If the *CreateRecordSchedule()* action returns successfully, then a new *recordSchedule* object is added to the list of Record Schedules maintained by the ScheduledRecording service. This list can be consulted through the *BrowseRecordSchedules()* action. The ScheduledRecording service MAY also instantiate one or more *recordTask* objects to represent the discrete recording tasks that are associated with the high level schedule, defined by the *recordSchedule*. The instantiation of *recordTask* objects may happen after the *CreateRecordSchedule()* action returns successfully. However, if the created *recordSchedule* would lead to the instantiation of one or more *recordTask* objects, these *recordTask* objects MUST be created by the ScheduledRecording service as soon as possible and within a reasonable amount of time. If any of these spawned *recordTask* objects end up in a state that indicates that these *recordTask* objects should already be recording, then the ScheduledRecording service MUST ensure that these recordings start as soon as possible and within a reasonable amount of time (this will most likely result in a partial recording). If a ScheduledRecording service implementation can not ensure that these recordings start as soon as possible, then the *CreateRecordSchedule()* action MUST return with error code 720 without any change.

2.6.7.1 Arguments

Table 2-19: Arguments for *CreateRecordSchedule()*

Argument	Direction	relatedStateVariable
Elements	*IN*	*A_ARG_TYPE_RecordScheduleParts*
RecordScheduleID	*OUT*	*A_ARG_TYPE_ObjectID*
Result	*OUT*	*A_ARG_TYPE_RecordSchedule*
UpdateID	*OUT*	*StateUpdateID*

2.6.7.1.1 *Elements*

The *Elements* input argument contains an XML escaped *srs XML Document* (see [SRS-XSD]). This document contains a single *recordScheduleParts*. The *recordScheduleParts* object identifies the desired property values for the *recordSchedule* object to be created. The new *recordSchedule* will be an instance of a specific *recordSchedule* class. Each class defines its set of member properties, some of which are REQUIRED, and some of which are OPTIONAL. See Appendix C, "AV Working Committee Class Definitions" for details. All REQUIRED member properties MUST be specified. If a control point omits supported OPTIONAL member properties from the *Elements* argument, then the ScheduledRecording service MUST create the *recordSchedule* with the appropriate default value for those omitted member properties. If unsupported properties or unknown properties are specified in the *Elements* argument, the

ScheduledRecording service MUST gracefully accept these. If an unsupported value is specified for a supported member property, the ScheduledRecording service MUST detect this and return error code 703.

2.6.7.1.2 *RecordScheduleID*

If the ScheduledRecording service accepts the *recordSchedule* in the *Elements* input argument, then the ScheduledRecording service MUST provide a value in this output argument. The returned *RecordScheduleID* value MUST be a unique value within the ScheduledRecording service. *RecordScheduleID* values are assumed to be opaque values without special meaning. Although a ScheduledRecording service may choose to use a *RecordScheduleID* value that was previously assigned (and later removed from the active list of *recordSchedule* instances), this specification recommends that the *RecordScheduleID* value be unique in time as well.

2.6.7.1.3 *Result*

The *Result* output argument contains an XML escaped *srs XML Document* (see [SRS-XSD]). This document contains the newly created *recordSchedule* object as described in Appendix A, "*srs XML Document*". Any properties specified in the input *Elements* argument MUST have the same values in the output *recordSchedule*. The ScheduledRecording service MUST return *all* supported member properties for which it has meaningful values. This complete set allows a control point to see the default values of those properties that it did not specify in the input *Elements* argument. Note that some properties such as *scheduleState* are defined as REQUIRED for an output *recordSchedule* and MUST be included in the returned document. Refer to Appendix C.1.1, "Relationships between Classes and Properties" for the support level of each property.

The ScheduledRecording service implementation MUST ensure that the information returned in this argument is always consistent. In other words, if during the information gathering process, certain updates occur, the ScheduledRecording service implementation MUST re-examine the already gathered information to verify that this information is still accurate before returning from the action invocation.

2.6.7.1.4 *UpdateID*

See Section 2.6.5.1.7, "*UpdateID*".

2.6.7.2 Dependency on State

None.

2.6.7.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.7.4 Errors

Table 2-20: Error Codes for *CreateRecordSchedule()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Invalid Syntax	The *recordSchedule* in the *Elements* argument has invalid syntax. This includes malformed XML in the *Elements* input argument or a general schema violation.

ErrorCode	errorDescription	Description
703	Invalid Value	One or more properties in the input *recordSchedule* (in the *Elements* argument) have an invalid value.
707	Read only Property	Specifying a read only property is not allowed.
708	Required Property	Omitting a REQUIRED property is not allowed
720	Cannot Process the Request	Cannot process the request in a reasonable amount of time.
730	Conflict	The specified *recordSchedule* is conflicting with one or more existing *recordSchedule* objects. The ScheduledRecording service MAY reject a conflicting *recordSchedule* and return with this error code.
731	Protected Contents	The specified contents are copy protected. The ScheduledRecording service MAY reject a *recordSchedule* that specifies copy protected contents and return with this error code.
732	No Media	The specified removable media is not inserted.
733	Media Write Protect	The specified removable media is write-protected.
734	Media No Space	The specified media does not have sufficient capacity.
735	Media Error	Error related to the specified destination media.
736	Too Many recordSchedules	The maximum number of *recordSchedule* objects is reached.
737	Resource Error	Error related to an application resource.

2.6.8 *DeleteRecordSchedule()*

The *DeleteRecordSchedule()* action is used to delete a specific *recordSchedule*. When the *recordSchedule* is deleted, all of the associated *recordTask* objects MUST also be deleted. The list of Record Schedules and their associated *recordScheduleID* currently maintained by the ScheduledRecording service can be retrieved through the *BrowseRecordSchedules()* action.

A *recordSchedule* can only be deleted when all of its associated *recordTask* objects are in the "*IDLE*" or the "*DONE*" phase. If any of the associated *recordTask* objects are in the "*ACTIVE*" phase, then the ScheduledRecording service MUST return with error code 705 (active *recordTask*) without any change. A control point that wants to recover from this error scenario can first delete the associated active *recordTask* objects by invoking the *DeleteRecordTask()* action on these objects and then delete the *recordSchedule*. The active *recordTask* objects can be retrieved by properly invoking the *BrowseRecordTasks()* action.

It must be noted that a ScheduledRecording service can delete a *recordSchedule* without control point intervention. For example, a non-recurring *recordSchedule* that has completed its last *recordTask* MAY (OPTIONALLY) be automatically deleted along with its associated *recordTask* objects. However, it is RECOMMENDED that a ScheduledRecording service implementation retains completed *recordSchedule* instances and their associated *recordTask* instances for a reasonable amount of time so that the user can examine potential error information after recording is completed.

2.6.8.1 Arguments

Table 2-21: Arguments for *DeleteRecordSchedule()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.8.1.1 *RecordScheduleID*

The *RecordScheduleID* argument contains the object ID of the *recordSchedule* to be deleted.

2.6.8.2 Dependency on State

None.

2.6.8.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.8.4 Errors

Table 2-22: Error Codes for *DeleteRecordSchedule()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
705	Active *recordTask*	One or more *recordTask* instances are actively recording.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.9 *GetRecordSchedule()*

This action is used to retrieve a single *recordSchedule* from the ScheduledRecording service.

2.6.9.1 Arguments

Table 2-23: Arguments for *GetRecordSchedule()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*
Filter	*IN*	*A_ARG_TYPE_PropertyList*
Result	*OUT*	*A_ARG_TYPE_RecordSchedule*
UpdateID	*OUT*	*StateUpdateID*

2.6.9.1.1 *RecordScheduleID*

The *RecordScheduleID* contains the object ID of the *recordSchedule* for which information is to be returned.

2.6.9.1.2 *Filter*

See Section 2.6.5.1.1, "*Filter*".

2.6.9.1.3 *Result*

The *Result* output argument contains an XML escaped *srs XML Document* that contains a single *recordSchedule* identified by the *@id* value specified in the *RecordScheduleID* argument. For further details, see Section 2.6.5.1.4, "*Result*".

2.6.9.1.4 *UpdateID*

See Section 2.6.5.1.7, "*UpdateID*".

2.6.9.2 Dependency on State

None.

2.6.9.3 Effect on State

None.

2.6.9.4 Errors

Table 2-24: Error Codes for *GetRecordSchedule()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.10 *EnableRecordSchedule()*

This OPTIONAL action is used to enable a previously disabled *recordSchedule*. Enabling a *recordSchedule* is allowed in any state except for the "*COMPLETED*" state. In this case, the action MUST return with error code 740.

The invocation of the *EnableRecordSchedule()* action enables all the associated *recordTask* objects in the "*IDLE*" or "*ACTIVE*" phase (See Section 2.6.14, "*EnableRecordTask()*") except for those which were disabled individually at the *recordTask* level via the *DisableRecordTask()* action. Disabling at the *recordTask* level always takes precedence. If any of the associated *recordTask* objects end up in a state that indicates that these *recordTask* objects should already be recording, then the ScheduledRecording service MUST ensure that these recordings start as soon as possible and within a reasonable amount of time (this will most likely result in a partial recording). If a ScheduledRecording service implementation can not ensure that these recordings start as soon as possible, then the *EnableRecordSchedule()* action MUST

return with error code 720. If the ScheduledRecording service can not enable some of the *recordTask* objects in the "*IDLE*" or "*ACTIVE*" phase, it MUST return error code 740 without any change.

Enabling a *recordSchedule* MUST NOT affect its *recordTask* objects in the "*DONE*" phase. These *recordTask* objects MUST NOT cause error code 739 to be generated.

2.6.10.1 Arguments

Table 2-25: Arguments for *EnableRecordSchedule()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.10.1.1 *RecordScheduleID*

The *RecordScheduleID* argument contains the object ID of the *recordSchedule* to be enabled.

2.6.10.2 Dependency on State

None.

2.6.10.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated (the *scheduleState@currentErrors* property and some *taskState@xxx* error properties might be updated).

2.6.10.4 Errors

Table 2-26: Error Codes for *EnableRecordSchedule()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.
739	Cannot enable/disable recordSchedule	One or more of the associated *recordTask* objects could not be enabled or disabled.
740	recordSchedule "COMPLETED"	The *recordSchedule* has already completed and cannot be enabled or disabled.

2.6.11 *DisableRecordSchedule()*

This OPTIONAL action is used to disable a *recordSchedule*. Disabling a *recordSchedule* is allowed in any state except for the "*COMPLETED*" state. In this case, the action MUST return with error code 740.

The invocation of the *DisableRecordSchedule()* action disables all associated *recordTask* objects in the "*IDLE*" phase (See Section 2.6.15, "*DisableRecordTask()*") except for those which were enabled individually at the *recordTask* level via the *EnableRecordTask()* action. Enabling at the *recordTask* level

always takes precedence. If the ScheduledRecording service can not disable some of the *recordTask* objects in the "*IDLE*"phase, it MUST return error code 739 without any change.

The *DisableRecordSchedule()* action has no impact on *recordTask* objects already in the "*ACTIVE*" phase. These *recordTask* objects complete as planned.

Also, disabling a *recordSchedule* MUST NOT affect its *recordTask* objects in the "*DONE*" phase. These *recordTask* objects MUST NOT cause error code 739 to be generated. A disabled *recordSchedule* MUST continue to generate new *recordTask* objects but they MUST all be disabled. This allows control points to understand which *recordTask* objects will become active, once the *RecordSchedule* is re-enabled. This also provides the means for a control point to enable individual *recordTask* objects, even when the *recordSchedule* is disabled.

2.6.11.1 Arguments

Table 2-27: Arguments for *DisableRecordSchedule()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.11.1.1 *RecordScheduleID*

The *RecordScheduleID* argument contains the object ID of the *recordSchedule* to be disabled.

2.6.11.2 Dependency on State

None.

2.6.11.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated (the *scheduleState@currentErrors* property and some *taskState@xxx* error properties might be updated).

2.6.11.4 Errors

Table 2-28: Error Codes for *DisableRecordSchedule()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.
739	Cannot enable/disable recordSchedule	One or more of the associated *recordTask* objects could not be enabled ordisabled.
740	recordSchedule "COMPLETED"	The *recordSchedule* has already completed and cannot be enabled or disabled.

2.6.12 *DeleteRecordTask()*

This OPTIONAL action is used to delete a *recordTask*. For any existing *recordTask*, this action MUST always succeed. The *recordTask* object is removed from the list of *recordTask* objects that is maintained by the ScheduledRecording service for the (parent) *recordSchedule* and any ongoing recording for this *recordTask* MUST stop immediately. The associated recorded content for that *recordTask* MUST NOT be deleted as a result of this action.

2.6.12.1 Arguments

Table 2-29: Arguments for *DeleteRecordTask()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.12.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* to be deleted.

2.6.12.2 Dependency on State

None.

2.6.12.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.12.4 Errors

Table 2-30: Error Codes for *DeleteRecordTask()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such *recordTask* ID	The specified *recordTask* does not exist.

2.6.13 *GetRecordTask()*

This action is used to retrieve a single *recordTask* from the ScheduledRecording service.

2.6.13.1 Arguments

Table 2-31: Arguments for *GetRecordTask()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*
Filter	*IN*	*A_ARG_TYPE_PropertyList*
Result	*OUT*	*A_ARG_TYPE_RecordTask*

Argument	Direction	relatedStateVariable
UpdateID	*OUT*	*StateUpdateID*

2.6.13.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* for which information is to be returned.

2.6.13.1.2 *Filter*

See Section 2.6.5.1.1, "*Filter*".

2.6.13.1.3 *Result*

The *Result* output argument contains an XML escaped *srs XML Document* that contains a single *recordTask* instance, identified by the *@id* value specified in the *RecordTaskID* argument. The *Result* argument is identical to the *Result* argument of the *BrowseRecordTasks()* action. See Section 2.6.6.1.5, "*Result*" for further details.

2.6.13.1.4 *UpdateID*

See Section 2.6.5.1.7, "*UpdateID*".

2.6.13.2 Dependency on State

None.

2.6.13.3 Effect on State

None.

2.6.13.4 Errors

Table 2-32: Error Codes for *GetRecordTask()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such *recordTask* ID	The specified *recordTask* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.14 *EnableRecordTask()*

This OPTIONAL action is used to first de-synchronize the *recordTask* enable/disable behavior from the (parent) *recordSchedule* and then individually enable the *recordTask*, if not already enabled.

- A *recordTask* that is enabled in the "*IDLE*" phase will record content in the future unless the occurrence of an error prevents that.

- A *recordTask* that is enabled in the "*ACTIVE*" phase MUST start recording content as soon as possible and within a reasonable amount of time unless the occurrence of an error prevents that. In that case, it MUST return error code 720 without any change.

Invoking *EnableRecordTask()* on a *recordTask* in the "*DONE*" phase MUST NOT affect the state of the *recordTask* and MUST fail with error code 741.

Enabling a *recordTask* always takes persistent precedence over enabling/disabling activities performed at the (parent) *recordSchedule* level. A *recordTask* that is enabled by invoking *EnableRecordTask()* remains enabled until explicitly disabled by invoking *DisableRecordTask*() on that *recordTask*. Invoking *EnableRecordSchedule()* or *DisableRecordSchedule()* on the (parent) *recordSchedule* does not affect the *recordTask* anymore. A *recordTask* enable/disable behavior can be re-synchronised to the (parent) *recordSchedule* by invoking the *ResetRecordTask()* action. From that point onwards, a *recordTask* will follow any enabling/disabling activities performed at the (parent) *recordSchedule* level again.

2.6.14.1 Arguments

Table 2-33: Arguments for *EnableRecordTask()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.14.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* to be enabled.

2.6.14.2 Dependency on State

None.

2.6.14.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.14.4 Errors

Table 2-34: Error Codes for *EnableRecordTask()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such recordTask ID	The specified *recordTask* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.
741	recordTask in "DONE" phase	A *recordTask* in the "DONE" phase cannot be enabled or disabled.

2.6.15 *DisableRecordTask()*

This OPTIONAL action is used to first de-synchronize the *recordTask* enable/disable behavior from the (parent) *recordSchedule* and then individually disable the *recordTask*, if not already disabled. A disabled *recordTask* MUST behave identical to an enabled *recordTask*, except for the following:

- A disabled *recordTask* in the "*IDLE*" phase MUST report error code 101 (Disabled) in the *taskState@pendingErrors* property.
- A disabled *recordTask* in the "*ACTIVE*" phase MUST NOT record content and it MUST report error code 101 (Disabled) in the *taskState@currentErrors* and *taskState@errorHistory* properties.

When a *recordTask* in the "*ACTIVE*" phase is disabled, it MUST stop recording immediately. If that is not possible, it MUST return error code 720 without any change. Invoking *DisableRecordTask()* on a *recordTask* in the "*DONE*" phase MUST NOT affect the state of the *recordTask* and MUST fail with error code 741.

Disabling a *recordTask* always takes persistent precedence over enabling/disabling activities performed at the (parent) *recordSchedule* level. A *recordTask* that is disabled by invoking *DisableRecordTask()* remains disabled until explicitly re-enabled by invoking *EnableRecordTask()* on that *recordTask*. Invoking *EnableRecordSchedule()* or *DisableRecordSchedule()* on the (parent) *recordSchedule* does not affect the *recordTask* anymore. A *recordTask* enable/disable behavior can be re-synchronised to the (parent) *recordSchedule* by invoking the *ResetRecordTask()* action. From that point onwards, a *recordTask* will follow any enabling/disabling activities performed at the (parent) *recordSchedule* level again.

2.6.15.1 Arguments

Table 2-35: Arguments for *DisableRecordTask()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.15.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* to be disabled.

2.6.15.2 Dependency on State

None.

2.6.15.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.15.4 Errors

Table 2-36: Error Codes for *DisableRecordTask()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such recordTask ID	The specified *recordTask* does not exist.

ErrorCode	errorDescription	Description
720	Cannot process the request	Cannot process the request in a reasonable amount of time.
741	recordTask in "DONE" phase	A *recordTask* in the "DONE" phase cannot be enabled or disabled.

2.6.16 *ResetRecordTask()*

This OPTIONAL action is used to force a previously enabled or disabled *recordTask* to follow any enabling/disabling activities performed at the (parent) *recordSchedule* level again.

If the (parent) *recordSchedule* is in the "*ENABLED*" state, then the effect of invoking the *ResetRecordTask()* action on an associated *recordTask* is identical to invoking the *EnableRecordTask()* action on that *recordTask* and from that point onwards, following any enabling/disabling activities performed at the (parent) *recordSchedule* level again for that *recordTask*.

If the (parent) *recordSchedule* is in the "*DISABLED*" state, then the effect of invoking the *ResetRecordTask()* action on an associated *recordTask* is identical to invoking the *DisableRecordTask()* action on that *recordTask* and from that point onwards, following any enabling/disabling activities performed at the (parent) *recordSchedule* level again for that *recordTask*.

2.6.16.1 Arguments

Table 2-37: Arguments for *ResetRecordTask()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*

2.6.16.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* to be reset.

2.6.16.2 Dependency on State

None.

2.6.16.3 Effect on State

The value of the *StateUpdateID* state variable is changed and the *LastChange* state variable is updated.

2.6.16.4 Errors

Table 2-38: Error Codes for *ResetRecordTask()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such recordTask ID	The specified *recordTask* does not exist.

ErrorCode	errorDescription	Description
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.17 *GetRecordScheduleConflicts()*

This action returns a CSV list of *recordSchedule* objects that conflict with the *recordSchedule* indicated by the *RecordScheduleID* argument.

Support of this action is REQUIRED if the ScheduledRecording service implementation allows conflicting *recordSchedule* instances to be created.

2.6.17.1 Arguments

Table 2-39: Arguments for *GetRecordScheduleConflicts()*

Argument	Direction	relatedStateVariable
RecordScheduleID	*IN*	*A_ARG_TYPE_ObjectID*
RecordScheduleConflictIDList	*OUT*	*A_ARG_TYPE_ObjectIDList*
UpdateID	*OUT*	*StateUpdateID*

2.6.17.1.1 *RecordScheduleID*

The *RecordScheduleID* argument contains the object ID of the *recordSchedule* for which all conflicting *recordSchedule* object ID values are to be returned in the *RecordScheduleConflictIDList* output argument.

2.6.17.1.2 *RecordScheduleConflictIDList*

This output argument contains the CSV list of *recordSchedule* object IDs that conflict with the *recordSchedule*, indicated by the *RecordScheduleID* argument.

2.6.17.1.3 *UpdateID*

The returned *UpdateID* argument MUST contain the most recent value of the *StateUpdateID* state variable *before* the action began collecting information to create the value returned in the *RecordScheduleConflictIDList* argument. This ensures that any changes that occur during the gathering of information can be detected by comparing the value of the *UpdateID* argument to the *updateID* attribute value in the most recent *LastChange* event. Refer to Section 2.4.4, "*StateUpdateID*" for more detailed information on the use of this argument.

2.6.17.2 Dependency on State

None.

2.6.17.3 Effect on State

None.

2.6.17.4 Errors

Table 2-40: Error Codes for *GetRecordScheduleConflicts()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.18 *GetRecordTaskConflicts()*

This action returns a CSV list of *@id* values of all the *recordTask* instances that conflict with the *recordTask* indicated by the *RecordTaskID* argument.

Support of this action is REQUIRED if the ScheduledRecording service implementation allows conflicting *recordTask* instances to be created.

2.6.18.1 Arguments

Table 2-41: Arguments for *GetRecordTaskConflicts()*

Argument	Direction	relatedStateVariable
RecordTaskID	*IN*	*A_ARG_TYPE_ObjectID*
RecordTaskConflictIDList	*OUT*	*A_ARG_TYPE_ObjectIDList*
UpdateID	*OUT*	*StateUpdateID*

2.6.18.1.1 *RecordTaskID*

The *RecordTaskID* argument contains the object ID of the *recordTask* for which all conflicting *recordTask* object ID values are to be returned in the *RecordTaskConflictIDList* output argument.

2.6.18.1.2 *RecordTaskConflictIDList*

This output argument contains the CSV list of *recordTask* object IDs that conflict with the *recordTask*, indicated by the *RecordTaskID* argument.

2.6.18.1.3 *UpdateID*

The returned *UpdateID* argument MUST contain the most recent value of the *StateUpdateID* state variable *before* the action began collecting information to create the value returned in the *RecordTaskConflictIDList* argument. This ensures that any changes that occur during the gathering of information can be detected by comparing the value of the *UpdateID* argument to the *updateID* attribute value in the most recent *LastChange* event. Refer to Section 2.4.4, "*StateUpdateID*" for more detailed information on the use of this argument.

2.6.18.2 Dependency on State

None.

2.6.18.3 Effect on State

None.

2.6.18.4 Errors

Table 2-42: Error Codes for *GetRecordTaskConflicts()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
713	No such recordTask ID	The specified *recordTask* does not exist.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.

2.6.19 Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 2-43: Common Error Codes

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.
701	Invalid Syntax	The *recordSchedule* in the *Elements* argument has invalid syntax. This includes malformed XML in the *Elements* input argument or a general schema violation.
702		Reserved for future extensions.
703	Invalid Value	One or more properties in the input *recordSchedule* (in the *Elements* argument) have an invalid value.
704	No such recordSchedule ID	The specified *recordSchedule* does not exist.
705	Active recordTask	One or more *recordTask* instances are actively recording.
706		Reserved for future extensions.
707	Read-only property	Unable to specify read-only property.
708	Required property	Omitting a REQUIRED property is not allowed
709	Unsupported or invalid sort criteria	The sort criteria specified are not supported or are invalid.
710		Reserved for future extensions.

ErrorCode	errorDescription	Description
711	Invalid DataTypeID	An invalid value has been specified in the *DataTypeID* input argument.
712		Reserved for future extensions.
713	No such recordTask ID	The specified *recordTask* does not exist.
714-719		Reserved for future extensions.
720	Cannot process the request	Cannot process the request in a reasonable amount of time.
721-729		Reserved for future extensions.
730	Conflict	The specified *recordSchedule* is conflicting with one or more existing *recordSchedule* objects. The ScheduledRecording service MAY reject a conflicting *recordSchedule* and return with this error code.
731	Protected Contents	The specified contents are copy protected. The ScheduledRecording service MAY reject a *recordSchedule* that specifies copy protected contents and return with this error code.
732	No Media	The specified removable media is not inserted.
733	Media Write Protect	The specified removable media is write-protected.
734	Media No Space	The specified media does not have sufficient capacity.
735	Media Error	Error related to the specified destination media.
736	Too many record schedules	The maximum number of *recordSchedule* objects is reached.
737	Resource Error	Error related to an application resource.
738		Reserved for future extensions.
739	Cannot enable/disable recordSchedule	One or more of the associated *recordTask* objects could not be enabled or disabled.
740	recordSchedule "COMPLETED"	The *recordSchedule* has already completed and cannot be enabled or disabled.
741	recordTask in "DONE" phase	A *recordTask* in the "DONE" phase cannot be enabled or disabled.

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

2.7 State Diagram of *recordTask*

In the ScheduledRecording service, the state of each *recordTask* is represented by its state properties (that is: *taskState* and its associated properties *taskState@xxx*). The definitions are described in Appendix B.16, "Task State Properties". Additionally, the state behavior of a *recordTask* is illustrated by a state diagram to give a visual description of each state and the state transitions. State diagrams are provided for

informational purposes. Whenever there is a discrepancy between the state diagram and the textual description of state and state transition, the normative textual description takes precedence.

2.7.1 A Full-Featured State Diagram

As described above, the *taskState* property reflects the current state of the *recordTask*. Its value changes over time as the *recordTask* progresses through its life-cycle. The following state transition diagram shows the possible states and state transitions that a given *recordTask* may take throughout its life time. It is assumed that all (REQUIRED and OPTIONAL) normative states and attributes of a *recordTask* are supported by the device. Further, it is assumed that a device is able to resume recording in the middle of the "*ACTIVE*" phase., The *GetAllowedValues()* action can be used to determine if a device supports all states and attributes.

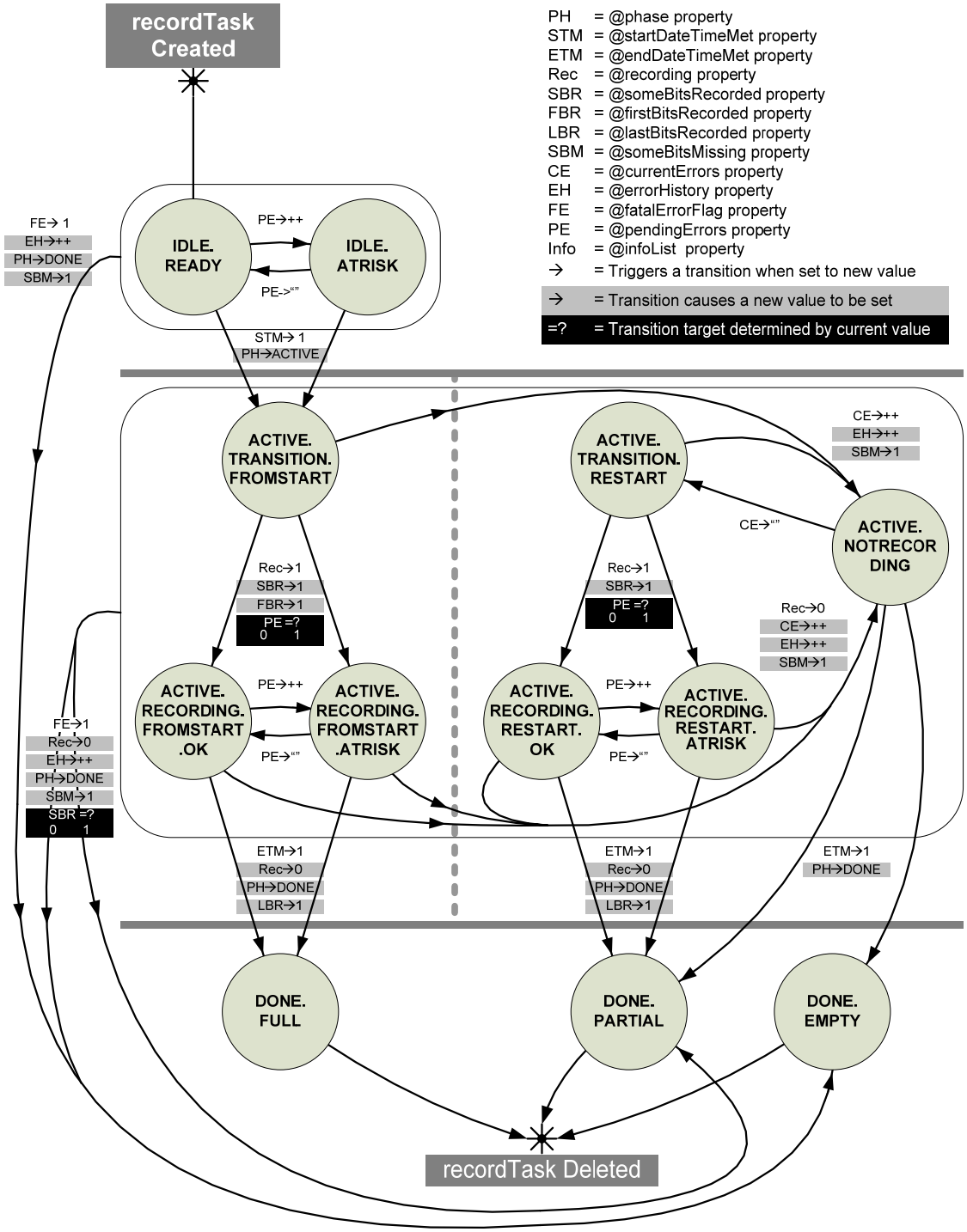

Figure 5: A Full-Featured State Diagram

2.7.1.1 "*IDLE*" Phase

The states in this phase indicate that the *recordTask*'s start time has not yet been reached and that the target content is not yet available for recording. The *recordTask* will remain in the "*IDLE*" phase (that is: in any of the IDLE states), until either the *recordTask*'s start time is reached or a fatal error is detected. If/when the start time is reached, the *recordTask* will transition to one of the states in the "*ACTIVE*" phase. If a fatal error is detected, the *recordTask* will transition directly to the "*DONE.EMPTY*" state within the "*DONE*" phase.

2.7.1.1.1 "*IDLE.READY*" State

This state indicates that the *recordTask* is waiting for the start time to be reached and that no error conditions (either fatal or non-fatal) have been detected. If/when the start time is reached, the *recordTask* will transition to one of the states in the "*ACTIVE*" phase. If, while waiting for the start time, a non-fatal error is detected, the *recordTask* will transition to the "*IDLE.ATRISK*" state indicating that the *recordTask* is at risk of not completing successfully due to some non-fatal error condition.

2.7.1.1.2 "*IDLE.ATRISK*" State

This state indicates that the *recordTask* is waiting for the start time to be reached, but that at least one non-fatal error condition has been detected. If/when the start time is reached, the *recordTask* will transition to one of the states in the "*ACTIVE*" phase. If, while waiting for the start time, the non-fatal error is resolved, the *recordTask* will transition back to the "*IDLE.READY*" state.

2.7.1.2 "*ACTIVE*" Phase

The states in this phase indicate that the *recordTask*'s start time has been reached and that the target content is available for recording. While in this phase (that is: in one of these states), the device will attempt to record the content. The *recordTask* will remain in this phase until either the *recordTask*'s end time is reached (that is: the content is no longer available) or until a fatal error is detected. If/when the end time is reached, the *recordTask* will transition to the appropriate "*DONE*" state based on how much of the content was recorded (that is: all – "*DONE.FULL*", part – "*DONE.PARTIAL*", or none – "*DONE.EMPTY*"). If a fatal error is detected, the *recordTask* will transition to either the "*DONE.PARTIAL*" or the "*DONE.EMPTY*" state, depending on how much of the content was recorded (that is: part or none).

2.7.1.2.1 "*ACTIVE.TRANSITION.FROMSTART*" State

This state indicates that the *recordTask* is attempting to begin recording the *recordTask*'s content from the beginning of the designated start time. The *recordTask* remains in this state until either the device actually begins recording data to the media or until a non-fatal or fatal error occurs. If the device actually starts to record data to the media, the *recordTask* will transition to "*ACTIVE.RECORDING.FROMSTART*" states where the content continues to be recorded. If the initial recording attempt fails due to a non-fatal error, the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts is made to resolve the problem and re-start the recording. If a fatal error is detected, the *recordTask* will transition to either the "*DONE.PARTIAL*" or the "*DONE.EMPTY*" state, depending on how much of the content was actually recorded (that is: part or none).

Although the *recordTask* remains in this state for a relatively short period of time, this state bridges an inherent discontinuity between the "*IDLE*" states and the "*ACTIVE*" states. Specifically, at the instant when the *recordTask*'s start time is reached, the *recordTask* (by definition) must transition out of the "*IDLE*" phase and into the "*ACTIVE*" phase, However, since the device has not yet attempted to record any content data on to the media, it is unknown which "*ACTIVE*" state the *recordTask* should transition to. Firstly, it is not appropriate to transition to any of the "*ACTIVE.RECORDING.xxx*" states because the device has not yet actually recorded any content data. Secondly, it is not appropriate to transition to the "*ACTIVE.NOTRECORDING*" state because this state (by definition) means that a non-fatal error has occurred resulting in the loss of content. Since no other "*ACTIVE*" states are appropriate at this instant in

time, the "*ACTIVE.TRANSITION.xxx*" states exist as a brief transition point while the true disposition of the *recordTask* is determined.

2.7.1.2.2 "*ACTIVE.TRANSITION.RESTART*" State

This state indicates that the *recordTask* is attempting to re-start the recording of the *recordTask*'s content some time after the beginning of the designated start time. This implies that either the initial recording attempt failed or that the initial recording attempt succeeded, but was later disrupted due to a non-fatal error. The *recordTask* remains in this state until either the device actually begins recording data to the media or until a non-fatal or fatal error occurs. If the device actually starts to record data to the media, the *recordTask* will transition to "*ACTIVE.RECORDING.RESTART*" states where the content continues to be recorded. If the initial recording attempt fails due to a non-fatal error, the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts is made to resolve the problem and re-attempt to start the recording. If a fatal error is detected, the *recordTask* will transition to either the "*DONE.PARTIAL*" or the "*DONE.EMPTY*" state, depending on how much of the content was recorded (that is: part of none).

Although the *recordTask* remains in this state for a relatively short period of time, this state bridges an inherent discontinuity between the "*ACTIVE.NOTRECORDING*" state and the "*ACTIVE.RECORDING.xxx*" states. Specifically, at the instant when a current non-fatal error has been resolved, the *recordTask* (by definition) must transition out of the "*ACTIVE.NOTRECORDING*" state and into one of the other "*ACTIVE*" states. However, since the device has not yet attempted to restart the recording of content data on to the media, it is unknown which "*ACTIVE*" state the *recordTask* should transition to. Firstly, it is not appropriate to transition to any of the "*ACTIVE.RECORDING.xxx*" states because the device has not yet actually (re)started to record any content data. Secondly, it is not appropriate to transition back to the "*ACTIVE.NOTRECORDING*" state because there are no unresolved non-fatal errors. Since no other "*ACTIVE*" states are appropriate at this instant in time, the "*ACTIVE.TRANSITION.xxx*" states exists as a brief transition point while the true disposition of the *recordTask* is determined.

2.7.1.2.3 "*ACTIVE.RECORDING.FROMSTART.OK*" State

This state indicates that the *recordTask* has reached its start time and that all of the target content has been recorded continuously from the beginning. Additionally, no non-fatal or fatal errors have occurred or have been detected which would otherwise threaten the future continuity of the recording. The *recordTask* remains in this state until either the *recordTask*'s end time is reached or until a non-fatal or fatal error occurs or a pending non-fatal or fatal error is detected.

If the *recordTask* reaches its end time, the *recordTask* halts the recording and transitions to the "*DONE.FULL*" state indicating that the entire target content was recorded uninterrupted. If a non-fatal error actually occurs, the recording has already halted and the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts are made to resolve the problem and restart the recording. If a fatal error actually occurs, the recording has already halted and the *recordTask* transitions directly to the "*DONE.PARTIAL*" state indicating that part of the target content was recorded. If a pending non-fatal or fatal error is detected (but has not yet occurred), the *recordTask* transitions to the "*ACTIVE.RECORDING.FROMSTART.ATRISK*" state indicating that the target content has been recorded continuously from the beginning, but a pending error has been detected that threatens the remainder of the recording.

2.7.1.2.4 "*ACTIVE.RECORDING.FROMSTART.ATRISK*" State

This state indicates that the *recordTask* has reached its start time and that all of the target content has been recorded continuously from the beginning. Although no non-fatal or fatal errors have occurred, one or more pending non-fatal or fatal errors have been detected that threaten the future continuity of the recording. The *recordTask* remains in this state until either the *recordTask*'s end time is reached or until all

of the pending non-fatal and fatal errors have been resolved or until a non-fatal or fatal error actually occurs.

If the *recordTask* reaches its end time, the *recordTask* halts the recording and transitions to the "*DONE.FULL*" state indicating that the entire target content was recorded uninterrupted. If all of the pending errors have been resolved, the *recordTask* transitions to the "*ACTIVE.RECORDING.FROMSTART.OK*" state indicating that the target content has been recorded continuously from the beginning and that no pending non-fatal or fatal errors have been detected. If a non-fatal error actually occurs, the recording has already halted and the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts are made to resolve the problem and restart the recording. If a fatal error actually occurs, the recording has already halted and the *recordTask* transitions directly to the "*DONE.PARTIAL*" state indicating that part of the target content was recorded.

2.7.1.2.5 "*ACTIVE.RECORDING.RESTART.OK*" State

This state indicates that the *recordTask* has reached its start time and that the target content data is being recorded onto the media. However, at some point in the past, the recording was disrupted either at the beginning or somewhere in the middle so that part of the content was not recorded. Fortunately, no pending non-fatal or fatal errors have been detected which would otherwise threaten the future continuity of the recording. The *recordTask* remains in this state until either the *recordTask*'s end time is reached or until a non-fatal or fatal actually occurs or a pending non-fatal or fatal error is detected.

If the *recordTask* reaches its end time, the *recordTask* halts the recording and transitions to the "*DONE.PARTIAL*" state indicating that part, but not all, of the target content was recorded. If a non-fatal error actually occurs, the recording has already halted and the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts are made to resolve the problem and again restart the recording. If a fatal error actually occurs, the recording has already halted and the *recordTask* transitions directly to the "*DONE.PARTIAL*" state indicating that part of the target content was recorded. If a pending non-fatal or fatal error is detected (but has not yet occurred), the *recordTask* transitions to the "*ACTIVE.RECORDING.RESTART.ATRISK*" state indicating that part of the target content has been recorded and that additional non-fatal or fatal errors are pending which threaten the remainder of the recording.

2.7.1.2.6 "*ACTIVE.RECORDING.RESTART.ATRISK*" State

This state indicates that the *recordTask* has reached its start time and that the target content data is being recorded onto the media. However, at some point in the past, the recording was disrupted either at the beginning or somewhere in the middle so that part of the content was not recorded. Additionally, one or more pending non-fatal or fatal errors have been detected that threaten the future continuity of the recording. The *recordTask* remains in this state until either the *recordTask*'s end time is reached or until all of the pending non-fatal and fatal errors have been resolved or until a non-fatal or fatal actually occurs.

If the *recordTask* reaches its end time, the *recordTask* halts the recording and transitions to the "*DONE.PARTIAL*" state indicating that part, but not all, of the target content was recorded. If all of the pending errors have been resolved, the *recordTask* transitions to the "*ACTIVE.RECORDING.RESTART.OK*" state indicating that the target content continues to be recorded, but with some content missing, and that no pending non-fatal or fatal errors have been detected. If a non-fatal error actually occurs, the recording has already halted and the *recordTask* transitions to the "*ACTIVE.NOTRECORDING*" state where one or more attempts are made to resolve the problem and again restart the recording. If a fatal error actually occurs, the recording has already halted and the *recordTask* transitions directly to the "*DONE.PARTIAL*" state indicating that part of the target content was recorded.

2.7.1.2.7 "*ACTIVE.NOTRECORDING*" State

This state indicates that a non-fatal error has occurred while the device was recording the target content or while the device was attempting to start recording the target content. The *recordTask* remains in this state

until either the *recordTask*'s end time is reached or until all of the current non-fatal errors are resolved, or until a fatal error actually occurs.

If the *recordTask* reaches its end time, the *recordTask* transitions to either the "*DONE.PARTIAL*" or "*DONE.EMPTY*" depending on how much of the content was actually recorded (that is: part or none). If all of the current non-fatal errors have been resolved, the *recordTask* transitions to the "*ACTIVE.TRANSITION.RESTART*" state where one or more attempts are made to restart the recording. If a fatal error is detected, the *recordTask* transitions to either the "*DONE.PARTIAL*" or the "*DONE.EMPTY*" state depending on how much of the content was actually recorded (that is: part or none).

2.7.1.3 "*DONE*" Phase

The states in this phase indicate that the device is finished with this *recordTask*. Each "*DONE*" state indicates the success or failure of the *recordTask* based on how much of the target content was actually recorded. Once the *recordTask* reaches one of the "*DONE*" states, it remains in that state until the *recordTask* is deleted and none of the *recordTask*'s property values change.

2.7.1.3.1 "*DONE.FULL*" State

This state indicates that all of the *recordTask*'s target content was recorded in its entirety without any interruptions. No error occurred while recording the target content. The *recordTask* remains in this state until the *recordTask* is deleted.

2.7.1.3.2 "*DONE.PARTIAL*" State

This state indicates that part of the *recordTask*'s target content was recorded, but not all of it. One or more errors occurred while recording the target content that prevented part of that content from being recorded. The *recordTask* remains in this state until the *recordTask* is deleted.

2.7.1.3.3 "*DONE.EMPTY*" State

This state indicates that none of the *recordTask*'s target content was recorded. One or more errors occurred that prevented the recording from even getting started. The *recordTask* remains in this state until the *recordTask* is deleted.

2.7.2 A Minimal-Implementation State Diagram

The simplest state diagram based on the minimum required state related properties is illustrated below to show the behavior of such a device and the progression of its state. The support level of these state related properties is defined in Appendix C.3.2, "*object.recordTask* Class". This example only uses the set of REQUIRED allowed values for the *taskState* property. In the example below, it is assumed that the device is UNABLE to resume recording once the "*ACTIVE*" phase is entered. By definition, any device MUST support at least the following 5 illustrated states.

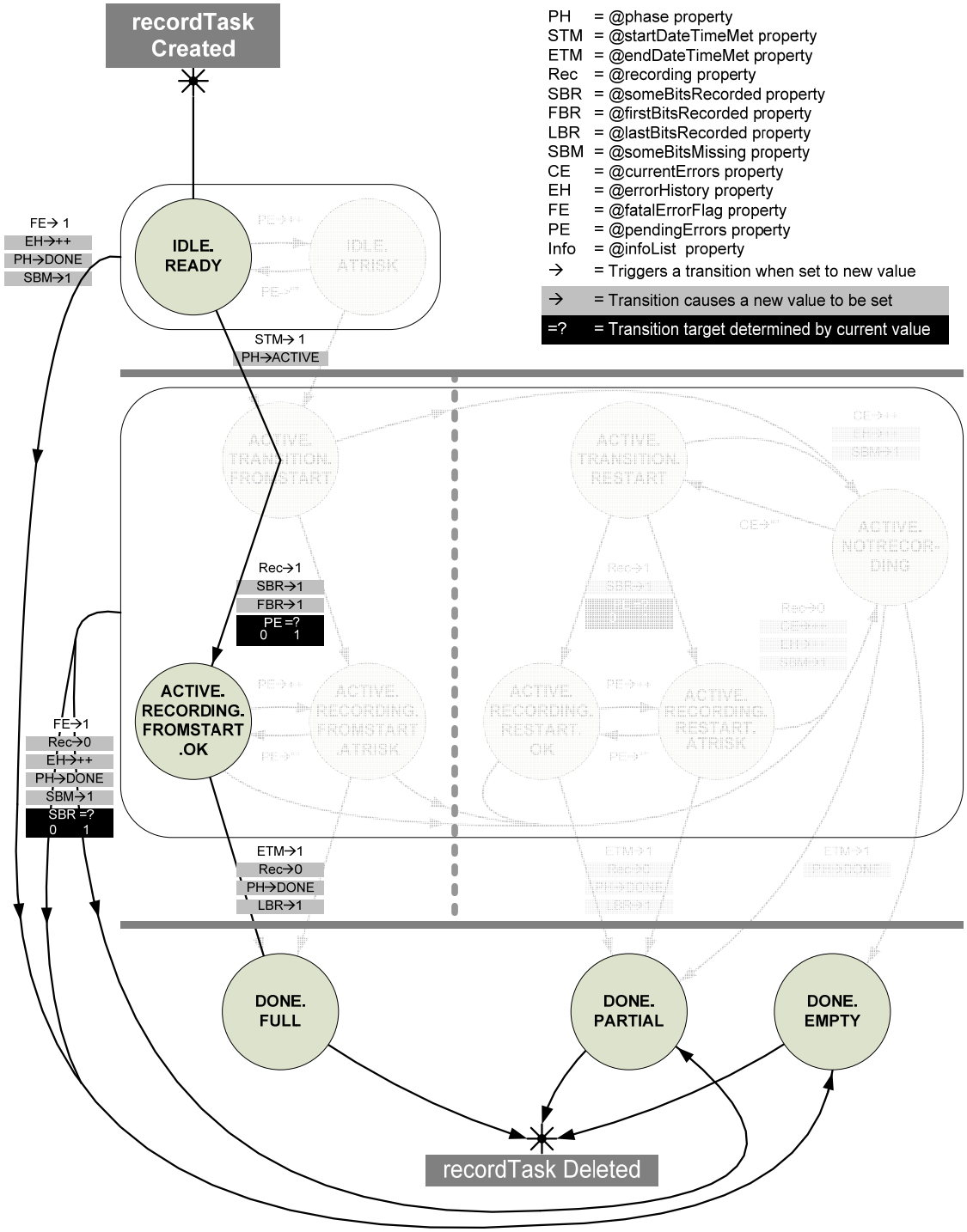

Figure 6: A Minimal-Implementation State Diagram

2.7.2.1 "*IDLE*" Phase

In this phase, the device is not able to detect pending errors (that is: *taskState@pendingErrors* MUST be empty); therefore, once the start time is reached, the device will go to the "*ACTIVE.RECORDING.FROMSTART.OK*" state and start recording. If the device can not start recording, it is treated as a fatal error, and the *recordTask* will transition directly to the "*DONE.EMPTY*" state. Also, anytime during the "*IDLE*" phase, a fatal error can occur, and the *recordTask* will transition directly to the "*DONE.EMPTY*" state.

2.7.2.1.1 "*IDLE.READY*" State

Because there is no error detecting mechanism (that is: *taskState@pendingErrors*) supported, this state indicates that the *recordTask* is waiting for the start time to be reached and that no errors conditions (either fatal or non-fatal) have been detected. If/when the start time is reached, the *recordTask* will attempt to record immediately.

2.7.2.1.2 "*IDLE.ATRISK*" State

Because there is no error detecting mechanism (that is: *taskState@pendingErrors*) supported, this state is not supported.

2.7.2.2 "*ACTIVE*" Phase

Because there is no pending error (that is: *taskState@pendingErrors*) detection mechanism supported, nor is an interrupted "*ACTIVE*" recording or late recording (that is: the start time is missed) able to resume recording (due to device limitations), only one state MUST be supported in the "*ACTIVE*" phase, that is: "*ACTIVE.RECORDING FROMSTART.OK*". It indicates a perfect recording condition.

2.7.2.2.1 "*ACTIVE.TRANSITION.FROMSTART*" State

This state is not supported.

2.7.2.2.2 "*ACTIVE.TRANSITION.RESTART*" State

This state is not supported.

2.7.2.2.3 "*ACTIVE.RECORDING.FROMSTART.OK*" State

This is the only state that MUST be supported in the "*ACTIVE*" phase. It indicates the perfect recording condition. The *recordTask* has reached its start time and all of the target content has been recorded continuously from the beginning. The *recordTask* remains in this state until either the *recordTask*'s end time is reached or until a fatal error is detected.

If the *recordTask* reaches its end time, the *recordTask* halts the recording and transitions to the "*DONE.FULL*" state indicating that the entire target content was recorded uninterrupted. If the recording is interrupted for any reason, it is treated as a fatal error, and the *recordTask* immediately transitions to either the "*DONE.PARTIAL*" or the "*DONE.EMPTY*" state.

2.7.2.2.4 "*ACTIVE.RECORDING.FROMSTART.ATRISK*" State

This state is not supported since the device does not support pending errors.

2.7.2.2.5 "*ACTIVE.RECORDING.RESTART.OK*" State

This state is not supported since the device can not resume an interrupted recording or catch a late recording that misses the beginning.

2.7.2.2.6 "*ACTIVE.RECORDING.RESTART.ATRISK*" State

This state is not supported since the device does not support pending errors.

2.7.2.2.7 "*ACTIVE.NOTRECORDING*" State

This state is not supported since the device can not resume interrupted recording. Any interruptions during the middle of recording will cause a transition to the "*DONE*" phase.

2.7.2.3 "*DONE*" Phase

The states in this phase indicate that the device is finished with this *recordTask*. Each "*DONE*" state indicates the success or failure of the *recordTask* based on how much of the target content was actually recorded.

2.7.2.3.1 "*DONE.FULL*" State

This state indicates a perfect recording. The *recordTask*'s target content was recorded in its entirety without any interruptions.

2.7.2.3.2 "*DONE.PARTIAL*" State

This state indicates that part of the *recordTask*'s target content was recorded, but not all of it. This state is reached from an "*ACTIVE*" *recordTask* due to a fatal error.

2.7.2.3.3 "*DONE.EMPTY*" State

This state indicates that none of the *recordTask*'s target content was recorded. It is a result of a recording that has never been started due to a fatal error.

2.7.3 *recordTask* State Example

The following example illustrates the use of state attributes. In this example, it is assumed that a device is able to resume a recording after it is interrupted.

The events occurs at:

- T0: Sytem is idle.
- T1: Error 1 (for example, DRM protected is being broadcast) and Error 3 (for example, conflicted-loser) are predicted.
- T2: The *recordSchedule* reaches the scheduled start time, but Error 1 prevents the recording from starting.
- T3: Suddenly, a new Error 2 occurs (for example, disabled)
- T4: Error 1 is fixed (for example, the protected part ends.), but Error 3 is still predicited.
- T5: Error 2 is fixed (for example, enabled by user), but Error 3 is still predicited.
- T6: Error 3 occurs (for example, other prioritized program starts)
- T7: Error 3 is fixed (for example, the prioritized program ends)
- T8: The *recordSchedule* reached the scheduled end time

Table 2-44: *recordTask* State Timeline

Time Line	Recording Schedule	Error 1	Error 2	Error 3	*taskState*	*@phase*	*@recording*	*@someBitsRecorded*	*@someBitsMissing*	*@startDateTimeMet*	*@endDateTimeMet*	*@firstBitsRecorded*	*@lastBitsRecorded*	*@fatalError*	*@currentErrors*	*@pendingErrors*	*@errorHistory*
T0					"*IDLE.READY*"	"*IDLE*"	0	0	0	0	0	0	0	0	Ø	Ø	Ø
T1					"*IDLE.ATRISK*"	"*IDLE*"	0	0	0	0	0	0	0	0	Ø	1,3	Ø
T2					"*ACTIVE.NOTRECORDING*"	"*ACTIVE*"	0	0	1	1	0	0	0	0	1	3	1
T3					"*ACTIVE.NOTRECORDING*"	"*ACTIVE*"	0	0	1	1	0	0	0	0	1,2	3	1,2
T4					"*ACTIVE.NOTRECORDING*"	"*ACTIVE*"	0	0	1	1	0	0	0	0	2	3	1,2
T5					"*ACT.RECORDING.RESTART.ATRISK*"	"*ACTIVE*"	1	1	1	1	0	0	0	0	Ø	3	1,2
T6					"*ACTIVE.NOTRECORDING*"	"*ACTIVE*"	0	1	1	1	0	0	0	0	3	Ø	1,2,3
T7					"*ACT.RECORDING.RESTART.OK*"	"*ACTIVE*"	1	1	1	1	0	0	0	0	Ø	Ø	1,2,3
T8					"*DONE.PARTIAL*"	"*DONE*"	0	1	1	1	1	0	1	0	Ø	Ø	1,2,3

2.8 ScheduledRecording Service Priority Model

2.8.1 Introduction of the ScheduledRecording Service Priority Model

The ScheduledRecording service priority model allows control points to provide desired priority information in order to help the ScheduledRecording service prioritize conflicting *recordTask* instances that were generated by different *recordSchedule* instances. The ScheduledRecording service priority model does not remove these conflicts from the system, but it does help the ScheduledRecording service make scheduling decisions that more closely match the desires of the end-user.

The ScheduledRecording service priority model is based on a "priority level" system in which each *recordSchedule* is assigned a specific priority level. The *recordTask* inherits the priority of its parent *recordSchedule*. In other words, the *recordTask* instances generated by a *recordSchedule* of a higher priority level are given higher priority than those *recordTask* instances generated by a *recordSchedule* of a lower priority level. Except for those ScheduledRecording service implementations that support "ordered priority" (described below), all of the *recordTask* instances generated by any of the *recordSchedule* instances assigned to the same priority level will have the same priority. If conflicts arise between any of these (same priority) *recordTask* instances, the ScheduledRecording service MAY give preference to any of these *recordTask* instances in a device-dependant manner.

The number of distinct priority levels supported by a ScheduledRecording service is vendor-dependent. Each priority level is identified by its name which MUST have the form "*L<x>*" where "L" is an abbreviation for "Level" and <x> is a number ranging from 1 to some device-specific maximum value n where n is the total number of distinct priority levels supported by the ScheduledRecording service.

For example, a ScheduledRecording service that supports 5 distinct priority levels will have the following priority levels named as follows:

- "*L1*" (Highest priority level)
- "*L2*"

- "*L3*"
- "*L4*"
- "*L5*" (Lowest priority level)

The list of priority levels supported by a ScheduledRecording service is obtainable via the *GetAllowedValues()* action by examining the allowed value list of the *priority* property. Each existing *recordSchedule* (on a given ScheduledRecording service) MUST be assigned one of these supported priority levels. The *priority* property of each *recordSchedule* indicates the current priority level assigned to that *recordSchedule* which can be retrieved via the *BrowseRecordSchedules()* action.

2.8.2 Ordered Priority within Each Priority Level

In addition to supporting one or more priority levels, some ScheduledRecording service implementations are able to prioritize the *recordSchedule* instances within each priority level. When ordered priority is supported, each *recordSchedule* (in addition to its assigned priority level) is also assigned a unique "ordered priority slot" ranging from 1 to <n> where <n> is the total number of *recordSchedule* instances within the ScheduledRecording service. A value of 1 represents the highest priority *recordSchedule* within the ScheduledRecording service; that is: the highest priority *recordSchedule* within the highest priority level "*L1*". The value <n> represents the lowest priority *recordSchedule* within the ScheduledRecording service; that is: the lowest *recordSchedule* within the lowest priority level. The ordered priority slot assigned to each *recordSchedule* can be obtained via the *recordSchedule*'s *priority@orderedValue* property. A ScheduledRecording service that support ordered priority MUST expose this property for each of their *recordSchedule* instances. Conversely, a ScheduledRecording service that does not support this capability MUST NOT expose the *priority@orderedValue* property. Within a given ScheduledRecording service, each ordered priority slot is assigned to exactly one *recordSchedule*.

As a natural consequence, the *recordSchedule* instances assigned to a higher priority level will always have a higher ordered priority than the *recordSchedule* instances assigned to a lower priority levels.

The following examples shows a ScheduledRecording service that supports ordered priority values within each of its 5 priority levels. The first example shows a ScheduledRecording service with fewer *recordSchedule* instances than the number of priority levels supported by that ScheduledRecording service. The second example shows a ScheduledRecording service with more *recordSchedule* instances than the number of priority levels supported by the ScheduledRecording service.

Of particular note, *recordSchedule* instances do not need to be evenly distributed between the different priority levels. Ordered priority slots are contiguously assigned starting with the highest priority *recordSchedule* down to the lowest priority *recordSchedule*.

Table 2-45: Example 1: Fewer *recordSchedule* instances than the Number of Supported Priority Levels.

Priority Level	*RecordScheduleID*	Ordered Priority Slot
"*L1*" (highest priority level)	*RS-A*	1
"*L2*"		
"*L3*"	*RS-C* *RS-B*	2 3
"*L4*"		
"*L5*" (lowest priority level)		

Table 2-46: Example 2: More *recordSchedule* instances than the Number of Supported Priority Levels.

Priority Level	*RecordScheduleID*	Ordered Priority Slot
"*L1*" (highest priority level)	*RS-A*	1
"*L2*"	*RS-F*	2
"*L3*"	*RS-C* *RS-B*	3 4
"*L4*"	*RS-E*	5
"*L5*" (lowest priority level)	*RS-G* *RS-D*	6 7

2.8.3 Setting the Initial Priority Level of a *recordSchedule*

The initial priority level of a *recordSchedule* is determined by the ScheduledRecording service when the *recordSchedule* is created. When determining the initial priority level, the ScheduledRecording service MUST examine the *recordSchedule*'s incoming *desiredPriority* property, and if provided, set the *recordSchedule*'s initial priority level as indicated. If the *desiredPriority* property is not set, then the ScheduledRecording service MUST assign the *recordSchedule* to one of the supported priority levels based on some device-dependent assignment algorithm. As described below, the *desiredPriority* property can be set to one of many different values which allow control points to express the desired priority in a number of different ways. The *GetAllowedValues()* action can be used to determine which values a ScheduledRecording service allows for its *desiredPriority* property.

The *desiredPriority* property has an associated *desiredPriority@type* property that MUST be set to "*PREDEF*" except when an object ID is specified in the *desiredPriority* property. In this case the *desiredPriority@type* property MUST be set to "*OBJECTID*" (see below for details).

In the simplest case, the incoming *desiredPriority* property is set to the name of one of the supported priority levels. This value indicates that the *recordSchedule* MUST be assigned to the specified priority level. If the ScheduledRecording service is not able to complete the assignment, then it MUST fail the creation request.

If a control point does not have a desired priority for a *recordSchedule* that it is about to create, the control point may set the incoming *desiredPriority* property to the value "*DEFAULT*". This value indicates that the control point is willing to accept the ScheduledRecording service's default priority level assignment.

If the ScheduledRecording service supports ordered priority (that is: the ScheduledRecording service supports the *priority@orderedValue* property), the ScheduledRecording service MUST also support some additional values for its *desiredPriority* property. Firstly, the ScheduledRecording service MUST support a value with the following format (without the double-quotes): "*<@id>*" where <@id> is the *@id* property value of an already existing *recordSchedule*. (The associated *desiredPriority@type* property MUST be set to "*OBJECTID*" in this case). This value indicates that the new *recordSchedule* MUST be assigned to the same priority level as the existing *recordSchedule* identified by <@id>. Furthermore, the new *recordSchedule* MUST be assigned the ordered priority slot of the existing *recordSchedule* with the existing *recordSchedule* and all other lower priority *recordSchedule* instances shifted to the next lower ordered priority slot. (See examples below.)

Additionally, when ordered priority is supported, the ScheduledRecording service MUST also support a number of convenience values corresponding to the highest and lowest ordered priority slots within each of its supported priority level. These convenience values MUST have the form "*L<x>_HI*" or "*L<x>_LOW*" where "L" is an abbreviation for "Level", <x> is a number ranging from 1 to some device-specific maximum value n where n is the total number of distinct priority levels supported by the ScheduledRecording service. For example, a ScheduledRecording service that supports 5 priority levels and also ordered priority MUST support the values "*L1_HI*", "*L1_LOW*", "*L2_HI*", "*L2_LOW*", "*L3_HI*",

"*L3_LOW*", "*L4_HI*", "*L4_LOW*", "*L5_HI*", "*L5_LOW*" for the *desiredPriority* property. Furthermore, the ScheduledRecording service MUST also support two additional convenience values corresponding to the highest and lowest priority within the ScheduledRecording service. These two additional convenience values are "*HIGHEST*" (which is equivalent to the highest ordered priority slot in the highest priority level "*L1_HI*"), and "*LOWEST*" (which is equivalent to the lowest priority slot within the lowest priority level "*L<n>_LOW*" when n is the total number of priority slots supported by the ScheduledRecording service).

All of these additional convenience values behave just like a "*<@id>*" value. The primary benefit of the convenience values is that they can be used to specify a specific ordered priority slot without having to determine the *@id* of the existing *recordSchedule* currently assigned to that slot. Additionally, as with a "*<@id>*" value, the existing *recordSchedule* already assigned to that desired ordered priority slot and those *recordSchedule* instances assigned to lower priority slots, are shifted to the next lower slot. However, all *recordSchedule* instances remain within their same priority level.

In the following examples, the ScheduledRecording service supports 3 priority levels and also supports ordered priority. The examples begin with the following *recordSchedule* priorities already assigned.

Table 2-47: Existing *recordSchedule* Priorities

Priority Level	*RecordScheduleID*	Ordered Priority Value
"*L1*" (highest priority level)	*RS-A*	1
"*L2*"	*RS-C*	2
"*L3*" (lowest priority level)	*RS-B*	3

Then the *CreateRecordSchedule()* action is invoked with the *desiredPriority* property set to "*RS-C*". After the action completes, a new *recordSchedule* is created with the *@id* property set to "*RS-D*". The set of *recordSchedule* instances is now prioritized as follows:

Table 2-48: *desiredPriority* Property Set to "*RS-C*"

Priority Level	*RecordScheduleID*	Ordered Priority Value
"*L1*" (highest priority level)	*RS-A*	1
"*L2*"	*RS-D* *RS-C*	2 3
"*L3*" (lowest priority level)	*RS-B*	4

Next the *CreateRecordSchedule()* action is invoked with the *desiredPriority* property set to "*HIGHEST*", "*L1_HI*", or "*RS-A*" (all values have the same effect). After the action completes, a new *recordSchedule* is created with the *@id* property set to "*RS-E*". The set of *recordSchedule* instances is now prioritized as follows:

Table 2-49: *desiredPriority* Property Set to "*HIGHEST*", "*L1_HI*", or "*RS-A*"

Priority Level	*RecordScheduleID*	Ordered Priority Value
"*L1*" (highest priority level)	*RS-E* *RS-A*	1 2
"*L2*"	*RS-D* *RS-C*	3 4
"*L3*" (lowest priority level)	*RS-B*	5

Now the *CreateRecordSchedule()* action is invoked with the *desiredPriority* property set to "*LOWEST*", "*L3_LOW*", or "*RS-B*" (all values have the same effect). After the action completes, a new *recordSchedule*

is created with the *@id* property set to "*RS-F*". The set of *recordSchedule* instances is now prioritized as follows:

Table 2-50: *desiredPriority* Property Set to "*LOWEST*", "*L3_LOW*", or "*RS-B*"

Priority Level	*RecordScheduleID*	Ordered Priority Value
"*L1*" (highest priority level)	*RS-E* *RS-A*	1 2
"*L2*"	*RS-D* *RS-C*	3 4
"*L3*" (lowest priority level)	*RS-B* *RS-F*	5 6

Finally, the *CreateRecordSchedule()* action is invoked with the *desiredPriority* property set to "*RS-C*". After the action completes, a new *recordSchedule* is created with the *@id* property set to "*RS-G*". The set of *recordSchedule* instances is now prioritized as follows:

Table 2-51: *desiredPriority* Property Set to "*RS-C*"

Priority Level	RecordScheduleID	Ordered Priority Value
"*L1*" (highest priority level)	*RS-E* *RS-A*	1 2
"*L2*"	*RS-D* *RS-G* *RS-C*	3 4 5
"*L3*" (lowest priority level)	*RS-B* *RS-F*	6 7

2.8.4 Sorting *recordSchedule* Instances Based on their Current Priority Settings

Control points can obtain the list of *recordSchedule* instances sorted either by their current priority level or by their ordered priority slot. In order to sort the list of *recordSchedule* instances by their current priority level (in descending order; that is: highest priority level *recordSchedule* instances listed first), control points can invoke the *BrowseRecordSchedules()* action with the *SortCriteria* argument set to "`+srs:priority`". In order to sort the list of *recordSchedule* instances sorted by their current ordered priority slot number (in descending order with the lowest ordered priority slot; that is: the highest slot number listed first), the control point can invoke the *BrowseRecordSchedules()* action with the *SortCriteria* argument set to "`-srs:priority@OrderedValue`".

2.9 Theory of Operation

2.9.1 Introduction

The following sections walk through several scenarios to illustrate the various actions supported by the ScheduledRecording service. It should be noted that these scenarios are for example purposes only and do not have any normative value. Vendors may combine the described components in a variety of ways.

NOTE: For easy readability, The *srs XML Documents* of the examples presented below are shown before XML-escaping to improve readability. However, they need to be escaped before embedding in a SOAP message. Also, a shorthand notation method is used to describe the actions. The SOAP envelope is omitted in the examples and replaced by a shorthand notation.

2.9.2 Checking the Capabilities of a ScheduledRecording Service

The following examples illustrate how to check the capabilities of the ScheduledRecording service by using the *GetSortCapabilities()*, *GetPropertyList()*, and *GetAllowedValues()* actions.

2.9.2.1 Checking the Sort Capabilities

Assume that the ScheduledRecording service supports sorting on *title*, *scheduledStartDateTime*, and *priority* only. Then the request:

Request:
```
GetSortCapabilities()
```

will result in the following response:

Response:
```
GetSortCapabilities("srs:title,srs:scheduledStartDateTime,srs:priority")
```

2.9.2.2 Checking Supported Properties and their Allowed Values

A number of properties are OPTIONAL and therefore, vendors are free to decide whether or not to support those properties for their particular ScheduledRecording implementations. The *GetPropertyList()* and *GetAllowedValues()* actions provide the means for a control point to determine which properties a particular ScheduledRecording service supports (*GetPropertyList()* action) and also what the allowed values are for these properties (*GetAllowedValues()* action). Since the set of supported properties and their allowed values may depend on the context within which these properties are used, the *GetPropertyList()* and *GetAllowedValues()* actions allow the control point to specify the *property-set* data type for which the control point wants to retrieve support level information.

2.9.2.2.1 Minimal Implementation Example

As a first example, assume that this particular ScheduledRecording service is a truly minimal implementation (only the *object.recordSchedule.direct.cdsNonEPG* class is supported and only required properties are supported).

Assume further that the control point wants to determine which properties it can specify in the *Elements* input argument of the *CreateRecordSchedule()* action of this minimal ScheduledRecording implementation. It first issues the following request (The *Elements* input argument of the *CreateRecordSchedule()* action is of data type *A_ARG_TYPE_RecordScheduleParts*):

Note: This *A_ARG_TYPE_RecordScheduleParts* example is marked by a white background for better reader orientation.

Request:
```
GetPropertyList("A_ARG_TYPE_RecordScheduleParts")
```

Then the following response will be generated:

Response:
```
GetPropertyList(
"srs:@id,srs:title,srs:class,srs:scheduledCDSObjectID,
 srs:scheduledStartDateTime,srs:scheduledDuration")
```

If the control point then wants to investigate further what values it may use for those properties when building a *recordSchedule*, it can retrieve that information using the following request:

Note: specifying "`*:*`" in the *Filter* argument is equivalent to specifying the complete list of property names that was returned in the *PropertyList* argument of the *GetPropertyList()* action with the *DataTypeID* argument set to "*A_ARG_TYPE_RecordScheduleParts*".

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordScheduleParts", "*:*")
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordScheduleParts</dataStructType>

   <fieldTable>
      <field>
         <name>srs:@id</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:title</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:class</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG
               </allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledCDSObjectID</name>
         <dataType>xsd:string</dataType>
```

```
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledStartDateTime</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledDuration</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>")
```

Assume further that the control point wants to determine which properties it can expect to get returned in the *Result* output argument of the *CreateRecordSchedule()* action of that same minimal ScheduledRecording implementation. It issues the following request (The *Result* argument of the *CreateRecordSchedule()* action is of data type *A_ARG_TYPE_RecordSchedule*):

Note: This *A_ARG_TYPE_RecordSchedule* example is marked by a grey background for better reader orientation.

Request:

```
GetPropertyList("A_ARG_TYPE_RecordSchedule")
```

The following response will be generated:

Response:

```
GetPropertyList(
"srs:@id,srs:title,srs:class,srs:priority,
 srs:recordDestination,srs:recordDestination@mediaType,
 srs:recordDestination@preference,
 srs:scheduledCDSObjectID,
 srs:scheduledStartDateTime,srs:scheduledDuration
 srs:scheduleState,srs:scheduleState@currentErrors,
 srs:abnormalTasksExist,srs:currentRecordTaskCount")
```

If the control point then wants to investigate further what values it may expect for some of those properties when browsing a *recordSchedule*, it can retrieve that information using the following request (the *Filter* argument contains only a subset of the possible properties in this example):

Note: specifying "*:*" in the *Filter* argument is again equivalent to specifying the complete list of properties returned in the *PropertyList* argument of the *GetPropertyList()* action with the *DataTypeID* argument set to "*A_ARG_TYPE_RecordSchedule*".

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordSchedule",
```

```
"srs:recordDestination,srs:recordDestination@mediaType,
 srs:scheduleState,srs:scheduleState@currentErrors,
 srs:abnormalTasksExist,srs:currentRecordTaskCount")
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordSchedule</dataStructType>

   <fieldTable>
      <field>
         <name>srs:recordDestination</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>Hard Disk</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination@mediaType</name>
         <dataType>xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>HDD</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduleState</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>OPERATIONAL</allowedValue>
```

```
               <allowedValue>ERROR</allowedValue>
               <allowedValue>COMPLETED</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduleState@currentErrors</name>
         <dataType>xsd:int</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduleState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue></allowedValue>
               <allowedValue>100</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:abnormalTasksExist</name>
         <dataType>xsd:boolean</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:currentRecordTaskCount</name>
         <dataType>xsd:unsignedInt</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>")
```

Assume further that the control point wants to determine which properties it can expect to get returned in the *Result* output argument of the *GetRecordTask()* action of that same minimal ScheduledRecording implementation. It issues the following request (The *Result* argument of the *GetRecordTask()* action is of data type *A_ARG_TYPE_RecordTask*):

Note: This *A_ARG_TYPE_RecordTask* example is marked by a light turquoise background for better reader orientation.

Request:

```
GetPropertyList("A_ARG_TYPE_RecordTask")
```

The following response will be generated:

Response:

```
GetPropertyList(
"srs:@id,srs:title,srs:class,srs:priority,
 srs:recordDestination,srs:recordDestination@mediaType,
```

```
 srs:recordDestination@preference,
 srs:recordScheduleID,
 srs:taskChannelID,srs:taskChannelID@type,srs:taskStartDateTime,
 srs:taskDuration,srs:recordQuality,srs:recordQuality@type,
 srs:taskState,srs:taskState@phase,
 srs:taskState@recording,srs:taskState@someBitsRecorded,
 srs:taskState@someBitsMissing,srs:taskState@fatalError,
 srs:taskState@currentErrors,srs:taskState@errorHistory,
 srs:taskState@pendingErrors,srs:taskState@infoList")
```

If the control point then wants to investigate further what values it may expect for some of those properties when browsing a *recordTask*, it can retrieve that information using the following request (the *Filter* argument contains only a subset of the possible properties in this example):

Note: specifying "*:*" in the *Filter* argument is again equivalent to specifying the complete list of properties returned in the *PropertyList* argument of the *GetPropertyList()* action with the *DataTypeID* argument set to "*A_ARG_TYPE_RecordTask*".

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordTask",
"srs:recordDestination,srs:recordDestination@mediaType,
 srs:taskState,srs:taskState@currentErrors")
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordTask</dataStructType>

   <fieldTable>
      <field>
         <name>srs:recordDestination</name>
         <dataType>xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>Hard Disk</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination@mediaType</name>
         <dataType>xsd:string</dataType>
```

```
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>HDD</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>IDLE.READY</allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.FROMSTART.OK
               </allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.FROMSTART.ATRISK
               </allowedValue>
               <allowedValue>DONE.FULL</allowedValue>
               <allowedValue>DONE.PARTIAL</allowedValue>
               <allowedValue>DONE.EMPTY</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@currentErrors</name>
         <dataType>xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue></allowedValue>
               <allowedValue>100</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>")
```

2.9.2.2.2 Full-fledged Implementation Example

In this example, it is assumed that this particular ScheduledRecording service supports all optional functionality, offered by the ScheduledRecording service specification.

Assume that the control point wants to determine which properties it can specify in the *Elements* input argument of the *CreateRecordSchedule()* action of this full-fledged ScheduledRecording implementation.

It issues the following request (The *Elements* input argument of the *CreateRecordSchedule()* action is of data type *A_ARG_TYPE_RecordScheduleParts*):

Note: This *A_ARG_TYPE_RecordScheduleParts* example is marked by a white background for better reader orientation.

Request:

```
GetPropertyList("A_ARG_TYPE_RecordScheduleParts")
```

Then the following response will be generated:

Response:

```
GetPropertyList(
"srs:@id,
 srs:title,
 srs:class,
 srs:desiredPriority,
 srs:desiredPriority@type,
 srs:recordDestination,
 srs:recordDestination@mediaType,
 srs:recordDestination@targetURL,
 srs:recordDestination@preference,
 srs:desiredRecordQuality,
 srs:desiredRecordQuality@type,
 srs:scheduledCDSObjectID,
 srs:scheduledChannelID,
 srs:scheduledChannelID@type,
 srs:scheduledStartDateTime,
 srs:scheduledDuration,
 srs:scheduledProgramCode,
 srs:scheduledProgramCode@type,
 srs:matchingName,
 srs:matchingName@type,
 srs:matchingName@subStringMatch,
 srs:matchingID,
 srs:matchingID@type,
 srs:matchingChannelID,
 srs:matchingChannelID@type,
 srs:matchingStartDateTimeRange,
 srs:matchingDurationRange,
 srs:matchingRatingLimit,
 srs:matchingRatingLimit@type,
 srs:matchingEpisodeType,
 srs:totalDesiredRecordTasks,
 srs:scheduledStartDateTimeAdjust,
 srs:scheduledDurationAdjust,
 srs:activePeriod,
 srs:durationLimit,
 srs:durationLimit@effect,
 srs:channelMigration,
 srs:timeMigration,
 srs:allowDuplicates,
 srs:persistedRecordings,
 srs:persistedRecordings@latest,
 srs:persistedRecordings@preAllocation,
 srs:persistedRecordings@storedLifetime")
```

If the control point then wants to investigate further what values it may use when building a *recordSchedule*, it can retrieve that information using the following request:

Request:
```
GetAllowedValues("A_ARG_TYPE_RecordScheduleParts", "*:*")
```

The following response will be generated:

Response:
See Appendix G.3, "*A_ARG_TYPE_RecordScheduleParts* AVDT Example" for a complete response message.

Assume further that the control point wants to determine which properties it can expect to get returned in the *Result* output argument of the *CreateRecordSchedule()* action of that same full-fledged ScheduledRecording implementation. It issues the following request (The *Result* output argument of the *CreateRecordSchedule()* action is of data type *A_ARG_TYPE_RecordSchedule*):

Note: This *A_ARG_TYPE_RecordSchedule* example is marked by a grey background for better reader orientation.

Request:
```
GetPropertyList("A_ARG_TYPE_RecordSchedule")
```

The following response will be generated:

Response:
```
GetPropertyList(
"srs:@id,
 srs:title,
 srs:class,
 srs:additionalStatusInfo,
 srs:cdsReference,
 srs:cdsReference@link,
 srs:priority,
 srs:priority@orderedValue,
 srs:desiredPriority,
 srs:desiredPriority@type,
 srs:recordDestination,
 srs:recordDestination@mediaType,
 srs:recordDestination@targetURL,
 srs:recordDestination@preference,
 srs:desiredRecordQuality,
 srs:desiredRecordQuality@type,
 srs:scheduledCDSObjectID,
 srs:scheduledCDSObjectID@link,
 srs:scheduledChannelID,
 srs:scheduledChannelID@type,
 srs:scheduledStartDateTime,
 srs:scheduledDuration,
 srs:scheduledProgramCode,
 srs:scheduledProgramCode@type,
 srs:matchingName,
 srs:matchingName@type,
 srs:matchingName@subStringMatch,
 srs:matchingID,
 srs:matchingID@type,
 srs:matchingChannelID,
 srs:matchingChannelID@type,
 srs:matchingStartDateTimeRange,
 srs:matchingDurationRange,
 srs:matchingRatingLimit,
 srs:matchingRatingLimit@type,
 srs:matchingEpisodeType,
 srs:totalDesiredRecordTasks,
```

```
 srs:scheduledStartDateTimeAdjust,
 srs:scheduledDurationAdjust,
 srs:activePeriod,
 srs:durationLimit,
 srs:durationLimit@effect,
 srs:channelMigration,
 srs:timeMigration,
 srs:allowDuplicates,
 srs:persistedRecordings,
 srs:persistedRecordings@latest,
 srs:persistedRecordings@preAllocation,
 srs:persistedRecordings@storedLifetime,
 srs:scheduleState,
 srs:scheduleState@currentErrors,
 srs:abnormalTasksExist,
 srs:currentRecordTaskCount,
 srs:totalCreatedRecordTasks,
 srs:totalCompletedRecordTasks")
```

If the control point then wants to investigate further what values it may expect for all of those properties when browsing a *recordSchedule*, it can retrieve that information using the following (The *Result* output argument of the *CreateRecordSchedule()* action is of data type *A_ARG_TYPE_RecordSchedule*):

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordSchedule", "*:*")
```

The following response will be generated:

Response:

See Appendix G.1, "*A_ARG_TYPE_RecordSchedule* AVDT Example" for a complete response message.

Assume further that the control point wants to determine which properties it can expect to get returned in the *Result* output argument of the *BrowseRecordTasks()* action of that same full-fledged ScheduledRecording implementation. It issues the following request (The *Result* output argument of the *BrowseRecordTasks()* action is of data type *A_ARG_TYPE_RecordTask*):

Note: This *A_ARG_TYPE_RecordTask* example is marked by a light turquoise background for better reader orientation.

Request:

```
GetPropertyList("A_ARG_TYPE_RecordTask")
```

The following response will be generated:

Response:

```
GetPropertyList(
"srs:@id,
 srs:title,
 srs:class,
 srs:additionalStatusInfo,
 srs:cdsReference,
 srs:cdsReference@link,
 srs:priority,
 srs:priority@orderedValue,
 srs:desiredPriority,
 srs:desiredPriority@type,
 srs:recordDestination,
 srs:recordDestination@mediaType,
 srs:recordDestination@targetURL,
 srs:recordDestination@preference,
srs:desiredRecordQuality,
```

```
srs:desiredRecordQuality@type,
srs:recordScheduleID,
srs:recordedCDSObjectID,
srs:recordedCDSObjectID@link
srs:taskCDSObjectID,
srs:taskCDSObjectID@link,
srs:taskChannelID,
srs:taskChannelID@type,
srs:taskStartDateTime,
srs:taskDuration,
srs:taskProgramCode,
srs:taskProgramCode@type,
srs:recordQuality,
srs:recordQuality@type,
srs:matchedName,
srs:matchedName@type,
srs:matchedID,
srs:matchedID@type,
srs:matchedRating,
srs:matchedRating@type,
srs:matchedEpisodeType,
srs:taskStartDateTimeAdjust,
srs:taskDurationAdjust,
srs:taskDurationLimit,
srs:taskDurationLimit@effect,
srs:taskChannelMigration,
srs:taskTimeMigration,
srs:taskState,
srs:taskState@phase,
srs:taskState@startDateTimeMet,
srs:taskState@endDateTimeMet,
srs:taskState@recording,
srs:taskState@someBitsRecorded,
srs:taskState@someBitsMissing,
srs:taskState@firstBitsRecorded,
srs:taskState@lastBitsRecorded,
srs:taskState@fatalError,
srs:taskState@currentErrors,
srs:taskState@errorHistory,
srs:taskState@pendingErrors,
srs:taskState@infoList")
```

If the control point then wants to investigate further what values it may expect for all of those properties when browsing a *recordTask*, it can retrieve that information using the following (The *Result* output argument of the *BrowseRecordTasks()* action is of data type *A_ARG_TYPE_RecordTask*):

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordTask", "*:*")
```

The following response will be generated:

Response:

See Appendix G.2, "*A_ARG_TYPE_RecordTask* AVDT Example" for a complete response message.

2.9.3 Adding a Scheduled Recording Entry to the List

The following examples illustrate how to create a *recordSchedule* entry in the list of *recordSchedule* instances by invoking the *CreateRecordSchedule()* action, using the different available *recordSchedule* classes. It is assumed that the implementation used in the examples that follow supports the allowed values for the *desiredRecordQuality* and *desiredRecordQuality@type* properties as indicated in Table B-9,

"*desiredRecordQuality* Example" and for the *recordQuality* and *recordQuality@type* properties as indicated in Table B-33, "*recordQuality* Example".

2.9.3.1 *object.recordSchedule.direct* classes

The *object.recordSchedule.direct* classes are used when the control point has all the necessary information available to uniquely identify the content to be recorded. The ScheduledRecording service does not have to perform searches or matching to determine what content is eligible for recording. Note that the control point might need to interact with external databases (like EPG information) to allow the user to make a selection of the content that he wants to record. Once the content is selected however, all information is available to set up the *recordSchedule* unambiguously.

2.9.3.1.1 Creating a *object.recordSchedule.direct.manual* Class *recordSchedule*

The *object.recordSchedule.direct.manual* class is used when the control point has access to the three basic components of information that are needed to uniquely identify the content to record:

- The scheduled channel that is used for broadcast of the content (*where*)
- The scheduled start date and time of the recording (*when*)
- The scheduled duration of the recording (*how long*)

It is assumed that the control point has some out-of-band means to retrieve this information. It passes this information into the *recordSchedule* using the REQUIRED properties *scheduledChannelID* and *scheduledChannelID@type*, *scheduledStartDateTime*, and *scheduledDuration*.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.direct.manual* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object.recordSchedule.direct.manual* class.

As an example, the control point wants to create a recurring *recordSchedule* to record the BBC news that is broadcast for one hour every evening at 7 pm on channel 47. Assume that the current date&time is Tuesday, June 28, 2005, 9:15 pm. If possible, the control point would like this recording to be stored on the internal hard disk, but if, for some reason, the hard disk is not available at the time of recording, the DVD+R drive may also be used as a secondary destination. The control point further specifies that this recording should be encoded using a low record quality setting of standard definition ("*SD*"). If that is not possible, any other record quality may be used ("*AUTO*"). A pre-roll time of two and a half minutes and a post-roll time of five minutes are also specified. The control point further instructs the ScheduledRecording service to keep at least the latest three recordings around. Older recordings may be discarded and no preallocation is desired.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>BBC News at 7pm</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</class>
      <desiredPriority type="PREDEF">L2</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
```

```
      </recordDestination>
      <recordDestination mediaType="DVD+R" preference="2">
         DVD Recorder
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         SD,AUTO
      </desiredRecordQuality>
      <scheduledChannelID type="ANALOG">47</scheduledChannelID>
      <scheduledStartDateTime>T19:00:00</scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
      <totalDesiredRecordTasks>0</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         -P00:02:30
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:05:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         3
      </persistedRecordings>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One or more *recordTask* instances may be created as a result of the *recordSchedule* creation. In this example, it is assumed that 2 *recordTask* instances are spawned immediately and it is also assumed that 2 new items are created in the associated ContentDirectory service that will hold the recorded content once the recordings are made (object IDs “rec00001” and “rec00002” are assigned).

Response:

```
CreateRecordSchedule("s101","
<?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">
   <item id="s101">
      <title>BBC News at 7pm</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</class>
      <priority>L2</priority>
      <desiredPriority type="PREDEF">L2</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="DVD+R" preference="2">
         DVD Recorder
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         SD,AUTO
      </desiredRecordQuality>
```

```
      <scheduledChannelID type="ANALOG">47</scheduledChannelID>
      <scheduledStartDateTime>T19:00:00</scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
      <totalDesiredRecordTasks>0</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         -P00:02:30
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:05:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         3
      </persistedRecordings>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>2</currentRecordTaskCount>
      <totalCreatedRecordTasks>2</totalCreatedRecordTasks>
      <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
   </item>
</srs>")
```

2.9.3.1.2 Creating a *object.recordSchedule.direct.cdsEPG* Class *recordSchedule*

The *object.recordSchedule.direct.cdsEPG* class is used when the control point has access to a local ContentDirectory service EPG database. The content to be recorded is uniquely identified by an EPG item in the associated ContentDirectory service. The association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device.

In this case, the basic component of information that is needed to uniquely identify the content to record is the object ID of the EPG item (contains the *where*, *when* and *how long* information) that represents that content. The control point passes this information into the *recordSchedule* using the REQUIRED *scheduledCDSObjectID* property.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.direct.cdsEPG* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object.recordSchedule.direct.cdsEPG* class.

As an example, the control point wants to create a *recordSchedule* to record the "UPnP Awards Ceremony" that is broadcast for a marathon fifteen hours on April 1st, at 9 am on channel 215. It finds this program in the EPG database of the associated ContentDirectory service and retrieves the object ID (value of the *didl-lite:@id* property of the EPG item). Due to the length of the program, the recording must be stored on the internal hard disk. If, for some reason, the hard disk is not available at the time of recording, the recording must be canceled. Further, if the recording would last longer than the anticipated 15 hours, the recording must be limited to 15 hours and the first part of the program discarded. The control point also specifies that this recording should be encoded using a low record quality setting of "*Q3*". If that is not possible, the recording will not be made. A pre-roll time of two minutes and a post-roll time of 15 minutes are also specified. The control point further instruct the ScheduledRecording service to keep track of this item in case the broadcaster decides to move it to a different channel and/or time.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>UPnP Awards Ceremony</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="QLEVEL">
         Q3
      </desiredRecordQuality>
      <scheduledCDSObjectID>
         epg_2005-04-01T09:00:00_P15:00:00
      </scheduledCDSObjectID>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         -P00:02:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:15:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <durationLimit effect="LAST">P15:00:00</durationLimit>
      <channelMigration>1</channelMigration>
      <timeMigration>1</timeMigration>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         1
      </persistedRecordings>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One *recordTask* instance may be created as a result of the *recordSchedule* creation. In this example, it is assumed that the *recordTask* instance is spawned immediately.

Response:

```
CreateRecordSchedule("s102","
<?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">
```

```
   <item id="s102">
      <title>UPnP Awards Ceremony</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
      <priority>L1</priority>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="QLEVEL">
         Q3
      </desiredRecordQuality>
      <scheduledCDSObjectID>
         epg_2005-04-01T09:00:00_P15:00:00
      </scheduledCDSObjectID>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         -P00:02:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:15:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <durationLimit effect="LAST">P15:00:00</durationLimit>
      <channelMigration>1</channelMigration>
      <timeMigration>1</timeMigration>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         1
      </persistedRecordings>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
      <totalCreatedRecordTasks>1</totalCreatedRecordTasks>
      <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
   </item>
</srs>")
```

2.9.3.1.3 Creating a *object.recordSchedule.direct.cdsNonEPG* Class *recordSchedule*

The *object.recordSchedule.direct.cdsNonEPG* class is used when the control point has access to a local ContentDirectory service database that contains items that identify content that will be available for recording at the time the recording is scheduled to start.

A typical example of this is TV tuner that is represented as a *channelGroup* container, containing items of class *object.item.videoItem.videoBroadcast*, each representing a channel to which the tuner can be tuned (User Channel). The association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device.

In this case, the basic components of information that are needed to uniquely identify the content to record are:

- The object ID of the ContentDirectory service item that represents the User Channel that is used for broadcast of the content (*where*)
- The scheduled start date and time of the recording (*when*)

- The scheduled duration of the recording (*how long*)

It is assumed that the control point has some out-of-band means to retrieve this information. It passes this information into the *recordSchedule* using the REQUIRED properties *scheduledCDSObjectID*, *scheduledStartDateTime*, and *scheduledDuration*.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.direct.cdsNonEPG* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object.recordSchedule.direct.cdsNonEPG* class.

As an example, assume that today's date is Tuesday, June 28, 2005 and the control point wants to create a *recordSchedule* to record the show "Life of a Software Developer" that is broadcast on channel 5 every Monday evening at 7 pm, starting on July 4th. The show lasts for an hour and runs for 13 episodes (until the end of September). The first fifteen minutes of each show are dedicated to a reading of the "Most Popular Software Code Quote of the Week". The user found all this information in a printed TV Guide. The ContentDirectory service has no EPG data.

The control point finds the User Channel that represents channel 5 in the associated ContentDirectory service and retrieves its object ID (value of the *didl-lite:@id* property of the User Channel item). The recording should be stored on the internal hard disk. If, for some reason, the hard disk is not available at the time of recording, the recording might also be recorded on an external network storage device. All episodes (13) of the show should be recorded. The control point also specifies that this recording should be encoded using a high record quality setting of High Definition ("*HD*"). The "Most Popular Software Code Quote of the Week" part of the show must be skipped but a pre-roll time of two minutes and a post-roll time of three minutes are also specified. All episodes must be preserved until deleted by the user.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>Life of a Software Developer</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2"
       targetURL="http://192.168.0.12/MyNAS/RecordedTV">
         Shared Content
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         HD
      </desiredRecordQuality>
      <scheduledCDSObjectID>
         User_Channel_5
      </scheduledCDSObjectID>
      <scheduledStartDateTime>
         MONT19:00:00
      </scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
```

```
      <totalDesiredRecordTasks>13</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         +P00:13:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:03:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/09-30T23:59:59</activePeriod>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="INFINITY">
         13
      </persistedRecordings>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One or more *recordTask* instances may be created as a result of the *recordSchedule* creation. In this example, it is assumed that 2 *recordTask* instances are spawned immediately.

Response:

```
CreateRecordSchedule("s103","
<?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">
   <item id="s103">
      <title>Life of a Software Developer</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
      <priority>L1</priority>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2"
       targetURL="http://192.168.0.12/MyNAS/RecordedTV">
         Shared Content
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         HD
      </desiredRecordQuality>
      <scheduledCDSObjectID>
         User_Channel_5
      </scheduledCDSObjectID>
      <scheduledStartDateTime>
         MONT19:00:00
      </scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
      <totalDesiredRecordTasks>13</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         +P00:13:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
```

```
          +P00:03:00
        </scheduledDurationAdjust>
        <activePeriod>NOW/09-30T23:59:59</activePeriod>
        <persistedRecordings
         latest="1"
         preAllocation="0"
         storedLifetime="INFINITY">
          13
        </persistedRecordings>
        <scheduleState
         currentErrors="">
          OPERATIONAL
        </scheduleState>
        <abnormalTasksExist>0</abnormalTasksExist>
        <currentRecordTaskCount>2</currentRecordTaskCount>
        <totalCreatedRecordTasks>2</totalCreatedRecordTasks>
        <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
    </item>
</srs>")
```

2.9.3.1.4 Creating a *object.recordSchedule.direct.programCode* Class *recordSchedule*

The *object.recordSchedule.direct.programCode* class is used when the control point has access (via the user, most likely) to a program code. The content to be recorded is uniquely identified by this program code in the sense that the program code contains in encoded form all necessary information for recording the program item (*where*, *when* and *how long*). If the ScheduledRecording service supports a particular program code type, that implies that the ScheduledRecording service must understand how to interpret and decode the program code into its *where*, *when* and *how long* components.

In this case, the basic component of information that is needed to uniquely identify the content to record is the program code of the program item that represents that content. The control point passes this information into the *recordSchedule* using the REQUIRED properties *scheduledProgramCode* and *scheduledProgramCode@type*.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.direct.programCode* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object. recordSchedule.direct.programCode* class.

As a hypothetical example, the control point wants to create a *recordSchedule* to record a program item, identified by a program code of type “upnpexample.com_upnpProgramCode” time. The upnpProgramCode type specifies the encoding to be simply:

program code = <channel number>_<StartDateTime>_<Duration>

The user retrieved the program code from some external source (a printed program guide) and the advertised title of the program is “Everything you ever wanted to know about SRS”.

It is assumed that the ScheduledRecording service supports the “upnpexample.com_upnpProgramCode” program code type and therefore knows how to decode the program code into its basic *where*, *when* and *how long* components. The recording must be stored on the internal DVD+RW drive. If, for some reason, the DVD+RW drive is not available at the time of recording, the recording must be stored on the internal hard disk. The control point also specifies that this recording should be recorded using any available record quality setting. No pre-roll or post-roll times are specified.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>About SRS</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</class>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="DVD+RW" preference="1">
         DVD Drive
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         AUTO
      </desiredRecordQuality>
      <scheduledProgramCode type="upnpexample.org_upnpProgramCode">
         2005-07-01_09:00:00_00:30:00
      </scheduledProgramCode>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One *recordTask* instance may be created as a result of the *recordSchedule* creation. In this example, it is assumed that the *recordTask* instance is spawned immediately.

Response:

```
CreateRecordSchedule("s104","
<?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">
   <item id="s104">
      <title>About SRS</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</class>
      <priority>L1</priority>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="DVD+RW" preference="1">
         DVD Drive
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         AUTO
      </desiredRecordQuality>
      <scheduledProgramCode type="upnpexample.org_upnpProgramCode">
         2005-07-01_09:00:00_00:30:00
      </scheduledProgramCode>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
```

```
            +P00:00:00
         </scheduledStartDateTimeAdjust>
         <scheduledDurationAdjust>
            +P00:00:00
         </scheduledDurationAdjust>
         <activePeriod>NOW/INFINITY</activePeriod>
         <durationLimit effect="LAST">INFINITY</durationLimit>
         <persistedRecordings
          latest="1"
          preAllocation="0"
          storedLifetime="ANY">
            0
         </persistedRecordings>
         <scheduleState
          currentErrors="">
            OPERATIONAL
         </scheduleState>
         <abnormalTasksExist>0</abnormalTasksExist>
         <currentRecordTaskCount>1</currentRecordTaskCount>
         <totalCreatedRecordTasks>1</totalCreatedRecordTasks>
         <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
      </item>
   </srs>")
```

2.9.3.2 *object.recordSchedule.query* classes

The *object.recordSchedule.query* classes are used when the control point only has partial information to identify possible candidates for recording. The ScheduledRecording service must perform further (continuous) searching or matching to determine what content is eligible for recording. The ScheduledRecording service must consult with external databases (like EPG information or over-the-wire Service Information) to find content that matches all the criteria, specified in the *recordSchedule*. Every time a match is found, a new *recordTask* is created.

2.9.3.2.1 Creating a *object.recordSchedule.query.contentName* Class *recordSchedule*

The *object.recordSchedule.query.contentName* class is used when the control point has knowledge about the (partial) name of the content to be recorded. This could either be a series name or a program name. Other properties, specified in the *recordSchedule* are also used to further narrow down what will be recorded (*activePeriod*, *totalDesiredRecordTasks*, etc.). It is the responsibility of the ScheduledRecording service to continuously search available external databases (like EPG or Service Information) and create a *recordTask* instance for every complete match (all specified matching criteria are satisfied) it finds within those external databases.

In this case, the basic piece of information that is needed to identify the content to record is the (partial) program or series name of the program item or series. The control point passes this information into the *recordSchedule* using the REQUIRED properties *matchingName* and *matchingName@type*.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.query.contentName* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object.recordSchedule.query.contentName* class.

As an example, the control point wants to create a *recordSchedule* to record the series entitled “Meet the UPnP Guys” (exact title). The control point has no further information, except that the series is broadcast during summer season and the series finale is planned somewhere during the month of September.

The recordings must be stored on the internal Hard Disk. If, for some reason, the Hard Disk is not available at the time of recording, the recording must be canceled. The control point also specifies that these recordings should be encoded using a low record quality (“*SD*”). If that is not possible, medium record quality (“*ED*”) may also be used. If that is not possible, no recording will be made. No pre-roll or post-roll

times are specified. If the broadcaster decides to change broadcast channel or date&time, the ScheduledRecording service is supposed to track.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>Meet the UPnP Guys series</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         SD,ED
      </desiredRecordQuality>
      <matchingName type="SERIES" subStringMatch="0">
         Meet the UPnP Guys
      </matchingName>
      <matchingStartDateTimeRange>
         NOW/09-30T23:59:59
      </matchingStartDateTimeRange>
      <totalDesiredRecordTasks>0</totalDesiredRecordTasks>
      <channelMigration>1</channelMigration>
      <timeMigration>1</timeMigration>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One or more *recordTask* instances may be created as a result of the *recordSchedule* creation. In this example, it is assumed that one *recordTask* instance is spawned immediately (12 remaining matches need to be found in the future, when new EPG data is available, for instance).

Response:

```
CreateRecordSchedule("s201","
<?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">
   <item id="s201">
      <title>Meet the UPnP Guys series</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
      <priority>L1</priority>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
```

```
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         SD,ED
      </desiredRecordQuality>
      <matchingName type="SERIES" subStringMatch="0">
         Meet the UPnP Guys
      </matchingName>
      <matchingStartDateTimeRange>
         NOW/09-30T23:59:59
      </matchingStartDateTimeRange>
      <totalDesiredRecordTasks>0</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         +P00:00:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:00:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <durationLimit effect="LAST">INFINITY</durationLimit>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         0
      </persistedRecordings>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
      <totalCreatedRecordTasks>1</totalCreatedRecordTasks>
      <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
   </item>
</srs>")
```

2.9.3.2.2 Creating a *object.recordSchedule.query.contentID* Class *recordSchedule*

The *object.recordSchedule.query.contentID* class is used when the control point has knowledge about the ID of the content to be recorded. This could either be a series ID or a program ID. Other properties, specified in the *recordSchedule* are also used to further narrow down what will be recorded (*activePeriod*, *totalDesiredRecordTasks*, etc.). It is the responsibility of the ScheduledRecording service to continuously search available external databases (like EPG or Service Information) and create a *recordTask* instance for every complete match (all specified matching criteria are satisfied) it finds within those external databases.

In this case, the basic piece of information that is needed to identify the content to record is the program ID or series ID of the program item or series. The control point passes this information into the *recordSchedule* using the REQUIRED properties *matchingID* and *matchingID@type*.

The control point creates a properly escaped *srs XML Document* that MUST contain all the REQUIRED properties necessary to create the *object.recordSchedule.query.contentID* class *recordSchedule*. The control point can add any OPTIONAL property that is applicable to the *object.recordSchedule.query.contentID* class.

As an example, the control point wants to create a *recordSchedule* to record the program with program ID "123456" from service provider "MyLocalProvider.net". It has obtained this ID through means outside the scope of this specification. The control point has no further information.

The recordings must be stored on the internal Hard Disk. If, for some reason, the Hard Disk is not available at the time of recording, the recording must be canceled. The control point also specifies that the recording

should be encoded using a high record quality setting of "*720p60*". If that is not possible, no recording will be made. No pre-roll or post-roll times are specified. If the broadcaster decides to change broadcast channel or date&time, the ScheduledRecording service is supposed to track.

To achieve the behavior specified above, the control point needs to provide the following *srs XML Document* in the *Elements* input argument of the *CreateRecordSchedule()* action:

Request:

```
CreateRecordSchedule("
<?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">
   <item id="">
      <title>My Program</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</class>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="ATSC">
         720p60
      </desiredRecordQuality>
      <matchingID type="MyLocalProvider.net">
         123456
      </matchingID>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
      <channelMigration>1</channelMigration>
      <timeMigration>1</timeMigration>
   </item>
</srs>")
```

If the creation of the *recordSchedule* is successful, the *CreateRecordSchedule()* action returns the following *srs XML Document* in the *Result* output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this *srs XML Document* MUST be properly escaped). One *recordTask* instance may be created as a result of the *recordSchedule* creation. In this example, it is assumed that the *recordTask* instance is spawned immediately.

Response:

```
CreateRecordSchedule("s202","
<?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">
   <item id="s202">
      <title>My Program</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</class>
      <priority>L1</priority>
      <desiredPriority type="PREDEF">L1</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <desiredRecordQuality type="ATSC">
         720p60
```

```
      </desiredRecordQuality>
      <matchingID type="MyLocalProvider.net">
         123456
      </matchingID>
      <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
      <scheduledStartDateTimeAdjust>
         +P00:00:00
      </scheduledStartDateTimeAdjust>
      <scheduledDurationAdjust>
         +P00:00:00
      </scheduledDurationAdjust>
      <activePeriod>NOW/INFINITY</activePeriod>
      <durationLimit effect="LAST">INFINITY</durationLimit>
      <persistedRecordings
       latest="1"
       preAllocation="0"
       storedLifetime="ANY">
         0
      </persistedRecordings>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
      <totalCreatedRecordTasks>1</totalCreatedRecordTasks>
      <totalCompletedRecordTasks>0</totalCompletedRecordTasks>
   </item>
</srs>")
```

2.9.4 Deleting a *recordSchedule*

A control point can delete a particular *recordSchedule* by invoking the *DeleteRecordSchedule()* action and specifying its object ID in the *RecordScheduleID* argument.

Assume that the *recordSchedule* to be deleted has its *@id* property set to “s301” .

To delete this *recordSchedule*, the control point generates the following request:

Request:

```
DeleteRecordSchedule("s301")
```

Response:

```
---
```

2.9.5 Browsing *recordSchedule* and *recordTask* instances

A control point can investigate which recordSchedule and/or recordTask instances are currently present within a ScheduledRecording service implementation by invoking the *BrowseRecordSchedules()* and *BrowseRecordTasks()* actions.

For example purposes, it is assumed that the control point has invoked the *CreateRecordSchedule()* action once for each of the cases described in Sections 2.9.3.1.1 through 2.9.3.1.4 and Sections 2.9.3.2.1 and 2.9.3.2.2. As a result, six *recordSchedule* instances as specified in the sections above have been created. In addition, eight *recordTask* instances have been created so that the available *recordSchedule* and *recordTask* instances in this particular ScheduledRecording service implementation are as follows:

recordSchedule (*@id* = “s101”, *class* = “*OBJECT.RECORDSCHEDULE.DIRECT.MANUAL*”)

recordTask (*@id* = “t101-001”, *class* = “*OBJECT.RECORDTASK*”)

recordTask (*@id* = “t101-002”, *class* = “*OBJECT.RECORDTASK*”)

recordSchedule (*@id* = "s102", *class* = "*OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG*")

recordTask (*@id* = "t102-001", *class* = "*OBJECT.RECORDTASK*")

recordSchedule (*@id* = "s103", *class* = "*OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG*")

recordTask (*@id* = "t103-001", *class* = "*OBJECT.RECORDTASK*")

recordTask (*@id* = "t103-002", *class* = "*OBJECT.RECORDTASK*")

recordSchedule (*@id* = "s104", *class* = "*OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE*")

recordTask (*@id* = "t104-001", *class* = "*OBJECT.RECORDTASK*")

recordSchedule (*@id* = "s201", *class* = "*OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME*")

recordTask (*@id* = "t201-001", *class* = "*OBJECT.RECORDTASK*")

recordSchedule (*@id* = "s202", *class* = "*OBJECT.RECORDSCHEDULE.QUERY.CONTENTID*")

recordTask (*@id* = "t202-001", *class* = "*OBJECT.RECORDTASK*")

2.9.5.1 Browsing *recordSchedule* instances

When a control point wants to gather detailed information on currently existing *recordSchedule* instances, it can do this by invoking the *BrowseRecordSchedules()* action. The following request:

Request:

```
BrowseRecordSchedules("", 0, 10, "+srs:title")
```

returns the following response (the result only returns the REQUIRED properties (*Filter* argument is set to "") and is sorted according to the value of the *title* property):

Response:

```
BrowseRecordSchedules("
<?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">
   <item id="s104">
      <title>About SRS</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</class>
      <priority>L1</priority>
      <recordDestination mediaType="DVD+RW" preference="1">
         DVD Drive
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2">
         Hard Disk
      </recordDestination>
      <scheduledProgramCode type="upnpexample.org_upnpProgramCode">
         2005-07-01_09:00:00_00:30:00
      </scheduledProgramCode>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
   </item>
   <item id="s101">
      <title>BBC News at 7pm</title>
```

```
      <class>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</class>
      <priority>L2</priority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="DVD+R" preference="2">
         DVD Recorder
      </recordDestination>
      <scheduledChannelID type="ANALOG">47</scheduledChannelID>
      <scheduledStartDateTime>T19:00:00</scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>2</currentRecordTaskCount>
   </item>
   <item id="s103">
      <title>Life of a Software Developer</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
      <priority>L1</priority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="HDD" preference="2"
       targetURL="http://192.168.0.12/MyNAS/RecordedTV">
         Shared Content
      </recordDestination>
      <scheduledCDSObjectID>
         User_Channel_5
      </scheduledCDSObjectID>
      <scheduledStartDateTime>
         MONT19:00:00
      </scheduledStartDateTime>
      <scheduledDuration>P01:00:00</scheduledDuration>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>2</currentRecordTaskCount>
   </item>
   <item id="s201">
      <title>Meet the UPnP Guys series</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
      <priority>L1</priority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <matchingName type="SERIES" subStringMatch="0">
         Meet the UPnP Guys
      </matchingName>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
   </item>
```

```
   <item id="s202">
      <title>My Program</title>
      <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</class>
      <priority>L1</priority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <matchingID type="MyLocalProvider.net">
         123456
      </matchingID>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
   </item>
   <item id="s102">
      <title>UPnP Awards Ceremony</title>
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
      <priority>L1</priority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <scheduledCDSObjectID>
         epg_2005-04-01T09:00:00_P15:00:00
      </scheduledCDSObjectID>
      <scheduleState
       currentErrors="">
         OPERATIONAL
      </scheduleState>
      <abnormalTasksExist>0</abnormalTasksExist>
      <currentRecordTaskCount>1</currentRecordTaskCount>
   </item>
</srs>",
6, 6, 123456)
```

2.9.5.2 Browsing *recordTask* instances associated with a single *recordSchedule*

When a control point wants to gather detailed information on currently existing *recordTask* instances that are associated with a particular *recordSchedule*, it can do this by invoking the *BrowseRecordTasks()* action.

As an example, assume that the control point wants to browse all *recordTask* instances, associated with the *recordSchedule* with its *@id* property set to "s101". It wants to retrieve all supported properties (*Filter* argument set to "*:*") and sorting is not important (*SortCriteria* argument set to "").

The following request:

Request:

```
BrowseRecordTasks("s101", "*:*", 0, 10, "")
```

returns the following response:

Response:

```
BrowseRecordTasks("
<?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">
   <item id="t101-001">
      <title>BBC News at 7pm</title>
      <class>OBJECT.RECORDTASK</class>
      <cdsReference link="LINK1">

<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple
string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;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-v2-20060531.xsd"&gt;
            &lt;item id="18" parentID="13" restricted="0"&gt;
               &lt;dc:title&gt;BBC News at 7pm&lt;/dc:title&gt;
               &lt;upnp:class&gt;
                  object.item.videoItem.videoBroadcast
               &lt;/upnp:class&gt;
               &lt;dc:creator&gt;BBC&lt;/dc:creator&gt;
            &lt;/item&gt;
         &lt;/DIDL-Lite&gt;

<!-- End of DIDL-Lite XML Document -->

      </cdsReference>
      <priority>L2</priority>
      <desiredPriority type="PREDEF">L2</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="DVD+R" preference="2">
         DVD Recorder
      </recordDestination>
      <desiredRecordQuality type="DEFAULT">
         SD,AUTO
      </desiredRecordQuality>
      <recordQuality type="DEFAULT">
         SD
      </recordQuality>
      <recordQuality type="ATSC">
         480i60
      </recordQuality>
      <recordQuality type="QLEVEL">
         Q3
      </recordQuality>
      <recordScheduleID>s101</recordScheduleID>
      <recordedCDSObjectID link="LINK1">
         rec00001
      </recordedCDSObjectID>
      <taskChannelID type="ANALOG">47</taskChannelID>
```

```
      <taskStartDateTime>
         2005-06-29T19:00:00
      </taskStartDateTime>
      <taskDuration>P01:00:00</taskDuration>
      <taskStartDateTimeAdjust>
         -P00:02:30
      </taskStartDateTimeAdjust>
      <taskDurationAdjust>
         +P00:05:00
      </taskDurationAdjust>
      <taskState
       phase="IDLE"
       startDateTimeMet="0"
       endDateTimeMet="0"
       recording="0"
       someBitsRecorded="0"
       someBitsMissing="0"
       firstBitsRecorded="0"
       lastBitsRecorded="0"
       fatalError="0"
       currentErrors=""
       errorHistory=""
       pendingErrors=""
       infoList="">
         IDLE.READY
      </taskState>
   </item>
   <item id="t101-002">
      <title>BBC News at 7pm</title>
      <class>OBJECT.RECORDTASK</class>
      <cdsReference link="LINK1">

<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple
string and therefore needs to be properly escaped
-->

        &lt;?xml version="1.0" encoding="UTF-8"?&gt;
        &lt;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-v2-20060531.xsd"&gt;
           &lt;item id="18" parentID="13" restricted="0"&gt;
              &lt;dc:title&gt;BBC News at 7pm&lt;/dc:title&gt;
              &lt;upnp:class&gt;
                 object.item.videoItem.videoBroadcast
              &lt;/upnp:class&gt;
              &lt;dc:creator&gt;BBC&lt;/dc:creator&gt;
           &lt;/item&gt;
        &lt;/DIDL-Lite&gt;

<!-- End of DIDL-Lite XML Document -->

      </cdsReference>
```

```
      <priority>L2</priority>
      <desiredPriority type="PREDEF">L2</desiredPriority>
      <recordDestination mediaType="HDD" preference="1">
         Hard Disk
      </recordDestination>
      <recordDestination mediaType="DVD+R" preference="2">
         DVD Recorder
      </recordDestination>
      <desiredRecordQuality type="LABEL">
         SD,AUTO
      </desiredRecordQuality>
      <recordQuality type="DEFAULT">
         HD
      </recordQuality>
      <recordQuality type="ATSC">
         1080i60
      </recordQuality>
      <recordQuality type="QLEVEL">
         Q1
      </recordQuality>
      <recordScheduleID>s101</recordScheduleID>
      <recordedCDSObjectID link="LINK1">
         rec00002
      </recordedCDSObjectID>
      <taskChannelID type="ANALOG">47</taskChannelID>
      <taskStartDateTime>
         2005-06-30T19:00:00
      </taskStartDateTime>
      <taskDuration>P01:00:00</taskDuration>
      <taskStartDateTimeAdjust>
         -P00:02:30
      </taskStartDateTimeAdjust>
      <taskDurationAdjust>
         +P00:05:00
      </taskDurationAdjust>
      <taskState
       phase="IDLE"
       startDateTimeMet="0"
       endDateTimeMet="0"
       recording="0"
       someBitsRecorded="0"
       someBitsMissing="0"
       firstBitsRecorded="0"
       lastBitsRecorded="0"
       fatalError="0"
       currentErrors=""
       errorHistory=""
       pendingErrors=""
       infoList="">
         IDLE.READY
      </taskState>
   </item>
</srs>",
2, 2, 123456)
```

2.9.6 Rating System

A ScheduledRecording service offers the OPTIONAL ability to impose rating limits on recordable content.

A ScheduledRecording service implementation may provide a list of supported ratings. The supported ratings can be retrieved by invoking the *GetAllowedValues()* action and specifying the *matchingRatingLimit* property in the *Filter* argument.

In the United States, TV manufacturers are REQUIRED to provide built-in support for the TV Parental Guidelines Monitoring Board rating system. (See http://www.tvguidelines.org.)

Motion picture content is rated on a voluntary basis by the Motion Picture Association of America. (See http://www.mpaa.org.)

Since it is not a simple matter to determine the rating system applicable to recordable content, the control point should provide values for all applicable rating systems when specifying a rating limit.

For example if the control point was configured to limit content for children, it may provide the following rating limit properties.

```
<matchingRatingLimit type="TVGUIDELINES.ORG">
   TV-G
</matchingRatingLimit>
<matchingRatingLimit type="MPAA.ORG">
   G
</matchingRatingLimit>
```

Since the intent of the rating limit is a limiting value, the ScheduledRecording service MUST exclude unrated content or content whose rating system does not match any of the rating types in the *matchingRatingLimit* properties provided by the control point.

Since rating limits are intended to preclude some (subset of) users from accessing content, it is up to the control point to identify users and apply the appropriate rating profile to individual users.

2.9.7 Conflict Detection and Resolution

Conflicts between *recordTask* instances arise when the recording events, associated with those recordTask instances, overlap in time and there are not enough resources available to record all of the requested recording events.

Conflict detection always happens at the *recordTask* level. It is possible that, at *recordSchedule* creation time, the ScheduledRecording service is not able to accurately indicate whether scheduling conflicts may arise in the future. Indeed, a ScheduledRecording service is not required or even capable (for a query-type *recordSchedule*) of generating all the *recordTask* instances that will ever be associated with the *recordSchedule*. Furthermore, a ScheduledRecording service implementation is allowed to either reject the creation of a *recordSchedule* that creates a scheduling conflict (the *CreateRecordSchedule()* action returns with error code 730, “Conflict”) or accept such a *recordSchedule*. A control point can therefore only rely on the occurrence of error code 401, “Conflicting Program Loser” or error code 402, “Conflicting Program Winner” in the *taskstate@currentErrors* property of all the *recordTask* instances to accurately determine whether scheduling conflicts exist. Note that the ScheduledRecording service always picks a Conflicting Program Winner, based upon priority settings and/or other vendor-defined criteria.

At this time, conflict resolution is not adequately supported by this specification. When one or more *recordTask* instances are conflicting, there is currently no straightforward way for a control point to change the Conflicting Program Winner. Instead, a control point may disable specific *recordTask* instances so that the intended *recordTask* becomes the Conflicting Program Winner. The drawback of this approach is that if the newly appointed Conflicting Program Winner changes over time (due to channel- or time migration, for instance), the disabled *recordTask* instances remain disabled and will not record, even if that would have become possible.

Alternatively, a control point may use the *DeleteRecordSchedule()* and *CreateRecordSchedule()* actions to reschedule the *recordSchedule* with a different priority level. The drawback of this approach is that all recordTask instances associated with the deleted *recordSchedule* are also deleted and any customization by the user that happened at the *recordTask* level will get lost as well.

A future version of this specification will address the conflict resolution issue in detail.

3 XML Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>

      <action>
         <name>GetSortCapabilities</name>
         <argumentList>
            <argument>
               <name>SortCaps</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SortCapabilities
               </relatedStateVariable>
            </argument>
            <argument>
               <name>SortLevelCap</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SortLevelCapability
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetPropertyList</name>
         <argumentList>
            <argument>
               <name>DataTypeID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_DataTypeID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>PropertyList</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_PropertyList
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetAllowedValues</name>
         <argumentList>
            <argument>
               <name>DataTypeID</name>
```

```
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_DataTypeID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Filter</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_PropertyList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>PropertyInfo</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_PropertyInfo
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>GetStateUpdateID</name>
      <argumentList>
         <argument>
            <name>Id</name>
            <direction>out</direction>
            <relatedStateVariable>
               StateUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>BrowseRecordSchedules</name>
      <argumentList>
         <argument>
            <name>Filter</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_PropertyList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>StartingIndex</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Index
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RequestedCount</name>
            <direction>in</direction>
            <relatedStateVariable>
```

```
            A_ARG_TYPE_Count
         </relatedStateVariable>
      </argument>
      <argument>
         <name>SortCriteria</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_SortCriteria
         </relatedStateVariable>
      </argument>
      <argument>
         <name>Result</name>
         <direction>out</direction>
         <relatedStateVariable>
            A_ARG_TYPE_RecordSchedule
         </relatedStateVariable>
      </argument>
      <argument>
         <name>NumberReturned</name>
         <direction>out</direction>
         <relatedStateVariable>
            A_ARG_TYPE_Count
         </relatedStateVariable>
      </argument>
      <argument>
         <name>TotalMatches</name>
         <direction>out</direction>
         <relatedStateVariable>
            A_ARG_TYPE_Count
         </relatedStateVariable>
      </argument>
      <argument>
         <name>UpdateID</name>
         <direction>out</direction>
         <relatedStateVariable>
            StateUpdateID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>BrowseRecordTasks</name>
   <argumentList>
      <argument>
         <name>RecordScheduleID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_ObjectID
         </relatedStateVariable>
      </argument>
      <argument>
         <name>Filter</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_PropertyList
```

```
            </relatedStateVariable>
         </argument>
         <argument>
            <name>StartingIndex</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Index
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RequestedCount</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Count
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SortCriteria</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SortCriteria
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Result</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecordTask
            </relatedStateVariable>
         </argument>
         <argument>
            <name>NumberReturned</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Count
            </relatedStateVariable>
         </argument>
         <argument>
            <name>TotalMatches</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Count
            </relatedStateVariable>
         </argument>
         <argument>
            <name>UpdateID</name>
            <direction>out</direction>
            <relatedStateVariable>
               StateUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>CreateRecordSchedule</name>
```

```
      <argumentList>
         <argument>
            <name>Elements</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecordScheduleParts
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RecordScheduleID</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Result</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecordSchedule
            </relatedStateVariable>
         </argument>
         <argument>
            <name>UpdateID</name>
            <direction>out</direction>
            <relatedStateVariable>
               StateUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>DeleteRecordSchedule</name>
      <argumentList>
         <argument>
            <name>RecordScheduleID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>GetRecordSchedule</name>
      <argumentList>
         <argument>
            <name>RecordScheduleID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
         <argument>
```

```
            <name>Filter</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_PropertyList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Result</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecordSchedule
            </relatedStateVariable>
         </argument>
         <argument>
            <name>UpdateID</name>
            <direction>out</direction>
            <relatedStateVariable>
               StateUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>EnableRecordSchedule</name>
      <argumentList>
         <argument>
            <name>RecordScheduleID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>DisableRecordSchedule</name>
      <argumentList>
         <argument>
            <name>RecordScheduleID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>DeleteRecordTask</name>
      <argumentList>
         <argument>
            <name>RecordTaskID</name>
            <direction>in</direction>
            <relatedStateVariable>
```

```
            A_ARG_TYPE_ObjectID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>GetRecordTask</name>
   <argumentList>
      <argument>
         <name>RecordTaskID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_ObjectID
         </relatedStateVariable>
      </argument>
      <argument>
         <name>Filter</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_PropertyList
         </relatedStateVariable>
      </argument>
      <argument>
         <name>Result</name>
         <direction>out</direction>
         <relatedStateVariable>
            A_ARG_TYPE_RecordTask
         </relatedStateVariable>
      </argument>
      <argument>
         <name>UpdateID</name>
         <direction>out</direction>
         <relatedStateVariable>
            StateUpdateID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>EnableRecordTask</name>
   <argumentList>
      <argument>
         <name>RecordTaskID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_ObjectID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>DisableRecordTask</name>
   <argumentList>
```

```
            <argument>
               <name>RecordTaskID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ObjectID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>ResetRecordTask</name>
         <argumentList>
            <argument>
               <name>RecordTaskID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ObjectID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetRecordScheduleConflicts</name>
         <argumentList>
            <argument>
               <name>RecordScheduleID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ObjectID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>RecordScheduleConflictIDList</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ObjectIDList
               </relatedStateVariable>
            </argument>
            <argument>
               <name>UpdateID</name>
               <direction>out</direction>
               <relatedStateVariable>
                  StateUpdateID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetRecordTaskConflicts</name>
         <argumentList>
            <argument>
               <name>RecordTaskID</name>
               <direction>in</direction>
```

```
            <relatedStateVariable>
               A_ARG_TYPE_ObjectID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>RecordTaskConflictIDList</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ObjectIDList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>UpdateID</name>
            <direction>out</direction>
            <relatedStateVariable>
               StateUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

</actionList>

<serviceStateTable>
   <stateVariable sendEvents="no">
      <name>SortCapabilities</name>
      <dataType>string</dataType>
   </stateVariable>

   <stateVariable sendEvents="no">
      <name>SortLevelCapability</name>
      <dataType>ui4</dataType>
   </stateVariable>

   <stateVariable sendEvents="no">
      <name>StateUpdateID</name>
      <dataType>ui4</dataType>
   </stateVariable>

   <stateVariable sendEvents="yes">
      <name>LastChange</name>
      <dataType>string</dataType>
   </stateVariable>

   <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_PropertyList</name>
      <dataType>string</dataType>
   </stateVariable>

   <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_DataTypeID</name>
      <dataType>string</dataType>
      <allowedValueList>
         <allowedValue>A_ARG_TYPE_RecordSchedule</allowedValue>
         <allowedValue>A_ARG_TYPE_RecordTask</allowedValue>
         <allowedValue>A_ARG_TYPE_RecordScheduleParts</allowedValue>
```

```
            </allowedValueList>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ObjectID</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ObjectIDList</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_PropertyInfo</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Index</name>
            <dataType>ui4</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Count</name>
            <dataType>ui4</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_SortCriteria</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_RecordSchedule</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_RecordTask</name>
            <dataType>string</dataType>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_RecordScheduleParts</name>
            <dataType>string</dataType>
         </stateVariable>

      </serviceStateTable>
   </scpd>
```

4 Test

No semantic tests have been specified for this service.

Appendix A. *srs XML Document* (Normative)

This section describes the *srs XML Document* that is used in action arguments of the *property-set* data type. Any *srs XML Document* MUST conform to the SRS schema as defined in [SRS-XSD]. Each *srs XML Document* contains one of the following data types: *A_ARG_TYPE_RecordSchedule*, *A_ARG_TYPE_RecordTask* or *A_ARG_TYPE_RecordScheduleParts*. All *property-set* data types are based on properties in the srs namespace and are therefore based on the SRS schema.

Due to limitations of the XML Schema syntax, the SRS schema in itself is often not adequate to accurately describe the limitations and restrictions imposed by a particular ScheduledRecording service implementation. For example, the set of supported properties and their allowed values may vary among implementations.

To allow ScheduledRecording service implementations to indicate which properties and their allowed values they support, the concept of the AV Datastructure Template (AVDT) is introduced. A ScheduledRecording service implementation can provide very detailed information about supported properties and their allowed values by means of an *AVDT XML Document*. The *AVDT XML Document* MUST conform to the AVDT schema as defined in [AVDT].

An *AVDT XML Document* can be retrieved by invoking the *GetAllowedValues()* action. The *DataTypeID* input argument identifies the data structure to be described by the *AVDT XML Document*. Indeed, depending on the particular ScheduledRecording service implementation, the set of supported properties and their allowed values of a given *property-set* data type may vary. For example, the set of properties that can be specified in the *Elements* input argument (of data type *A_ARG_TYPE_RecordScheduleParts*) of the *CreateRecordSchedule()* action may differ substantially between implementations. Additionally, the set of properties supported by different data types will obviously vary as well.

At this time, this specification identifies three different *AVDT XML Document* manifestations, depending on the data type of the objects described in the *AVDT XML Document*:

- The *A_ARG_TYPE_RecordSchedule AVDT XML Document*
- The *A_ARG_TYPE_RecordTask AVDT XML Document*
- The *A_ARG_TYPE_RecordScheduleParts AVDT XML Document*

A.1 *A_ARG_TYPE_RecordSchedule AVDT XML Document*

This type of *AVDT XML Document* is used to describe the data structure of a *recordSchedule* object for a particular ScheduledRecording service implementation. Examples of action arguments that use this data type include the *Result* output argument of the *BrowseRecordSchedules()* and *GetRecordSchedule()* actions.

When using the *AVDT XML Document* in this context, the following rules apply:

- The `<contextID>` field MUST be set to "uuid:*device-UUID*::urn:schemas-upnp-org:service:ScheduledRecording:1".
- The `<dataStructType>` field MUST be set to "*A_ARG_TYPE_RecordSchedule*".
- The `<fieldTable>` field MUST contain field elements for all the REQUIRED properties of all the *object.recordSchedule.xxx* classes supported by the service. Refer to Table C-2, "Class Properties Overview for *recordSchedule*", *recordSchedule*-related columns.
- The `<fieldTable>` field MUST also contain field elements for all the supported OPTIONAL properties of all the *object.recordSchedule.xxx* classes implemented by the service. Refer to Table C-2, "Class Properties Overview for *recordSchedule*", *recordSchedule*-related columns.

Field specific rules:

- There must be one and only one field with the subelement <name> set to "class"
- The allowed values for this field MUST only be derived from the *object.recordSchedule* virtual class.
- The <name> subelement of all <field> elements MUST only contain names of *recordSchedule* properties.

For a full-fledged example of a *A_ARG_TYPE_RecordSchedule* *AVDT XML Document*, see Appendix G.1, "*A_ARG_TYPE_RecordSchedule* AVDT Example

A.2 *A_ARG_TYPE_RecordTask* AVDT XML Document

This type of *AVDT XML Document* is used to describe the data structure of a *recordTask* object for a particular ScheduledRecording service implementation. Examples of action arguments that use this data type include the *Result* output argument of the *BrowseRecordTasks()* and *GetRecordTask()* actions.

When using the *AVDT XML Document* in this context, the following rules apply:

- The <contextID> field MUST be set to "uuid:*device-UUID*::urn:schemas-upnp-org:service:ScheduledRecording:1".
- The <dataStructType> field MUST be set to "*A_ARG_TYPE_RecordTask*".
- The <fieldTable> field MUST contain field elements for all the REQUIRED properties of the *object.recordTask* class. Refer to Table C-2, "Class Properties Overview for *recordSchedule*", *recordTask*-related column.
- The <fieldTable> field MUST also contain field elements for all the supported OPTIONAL properties of the *object.recordTask* class. Refer to Table C-2, "Class Properties Overview for *recordSchedule*", *recordTask*-related column.

Field specific rules:

- There must be one and only one field with the subelement <name> set to "class"
- The allowed values for this field MUST only be derived from the *object.recordTask* class.
- The <name> subelement of all <field> elements MUST only contain names of *recordTask* properties.

For a full-fledged example of a *A_ARG_TYPE_RecordTask* *AVDT XML Document*, see Appendix G.2, "*A_ARG_TYPE_RecordTask* AVDT Example".

A.3 *A_ARG_TYPE_RecordScheduleParts* AVDT XML Document

This This type of *AVDT XML Document* is used to describe the data structure of a *recordScheduleParts* object for a particular ScheduledRecording service implementation. Examples of action arguments that use this data type include the *Elements* input argument of the *CreateRecordSchedule()* action.

When using the *AVDT XML Document* in this context, the following rules apply:

- The <contextID> field MUST be set to "uuid:*device-UUID*::urn:schemas-upnp-org:service:ScheduledRecording:1".
- The <dataStructType> field MUST be set to "*A_ARG_TYPE_RecordScheduleParts*".
- The <fieldTable> field MUST contain field elements for all the REQUIRED properties of all the *object.recordSchedule.xxx* classes supported by the service. Refer to Table C-1, "Class Properties Overview ", *recordSchedule*-related columns.

- The `<fieldTable>` field MUST also contain field elements for all the supported OPTIONAL properties of all the *object.recordSchedule.xxx* classes implemented by the service. Refer to Table C-1, "Class Properties Overview ", *recordSchedule*-related columns.

Field specific rules:

- There must be one and only one field with the subelement `<name>` set to "class"
- The allowed values for this field MUST only be derived from the *object.recordSchedule* virtual class.
- The `<name>` subelement of all `<field>` elements MUST only contain names of *recordSchedule* properties.

For a full-fledged example of a *A_ARG_TYPE_RecordScheduleParts* *AVDT XML Document*, see Appendix G.3, "*A_ARG_TYPE_RecordScheduleParts* AVDT Example".

Appendix B. AV Working Committee Extended Properties (Normative)

The tables and sections below list all properties defined by the AV Working Committee. A property is expressed in XML as either an XML element or an XML attribute.

In the following sections, the definition of each property and its default value, if applicable, is described, followed by the specifics pertaining to INPUT and OUTPUT usage for this property. The INPUT usage indicates how the property is used in a *recordScheduleParts* object. The OUTPUT usage indicates how the property is used in a *recordSchedule* and/or *reccordTask* object.

Note: The NS column in the tables contains the namespace prefix of the namespace to which the property name belongs. The M-Val column indicates whether the property is multi-valued (M-Val = *YES*) or single-valued (M-Val = *NO*). See Section 2.2.2.17, "Multi-valued property" and Section 2.2.2.18, "Single-valued property".

B.1 Base Properties

Table B-1: Base Properties Overview

Property Name	NS	Data Type	M-Val	Reference
@id	srs	xsd:string	*NO*	Appendix B.1.1
title	srs	xsd:string	*NO*	Appendix B.1.2
class	srs	xsd:string	*NO*	Appendix B.1.3
additionalStatusInfo	srs	xsd:string	*NO*	Appendix B.1.4
cdsReference	srs	xsd:string	*YES*	Appendix B.1.5
cdsReference@link	srs	xsd:string	*NO*	Appendix B.1.5.1

B.1.1 *@id*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *@id* property identifies a *recordSchedule* or *recordTask* object. The value MUST be unique in the ScheduledRecording service. The value MUST be set by the ScheduledRecording service.

Default Value: N/A – Required on input.

Sort Order: Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating *@id* values. If *@id* values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical.

Input: When creating a new *recordSchedule* object, the *@id* property MUST be specified to satisfy the SRS XML Schema and MUST be set to the empty string.

Output: The unique object ID set by the ScheduledRecording service.

B.1.2 *title*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: Primary title of the object. The *title* property contains a friendly name to identify the object. This property can be either user-supplied or derived from the content name the object represents. This property is not to be confused with the *matchingName* or *matchedName* property. See also http://dublincore.org/documents/dces.

Default Value: N/A – Required on input.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.1.3 *class*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *class* property identifies the class of the object. A ScheduledRecording service implementation MUST list all classes it supports. If some (vendor-defined) classes are derived from other classes, then both the derived classes and the parent classes MUST be listed. See Appendix C, "AV Working Committee Class Definitions" for details.

Default Value: N/A – Required on input.

Sort Order: Sequenced Lexical. Sequence subvalues are substrings separated by periods.

Input: The desired setting.

Output: The current setting.

B.1.3.1 allowedValueList for the *class* Property

Table B-2: allowedValueList for the *class* Property

Value	R/O	Description
"*OBJECT.RECORDSCHEDULE.DIRECT.MANUAL*"	*O*	Control points should support *all* predefined values in these rows.
"*OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG*"	*O*	
"*OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG*"	*R*	
"*OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE*"	*O*	
"*OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME*"	*O*	
"*OBJECT.RECORDSCHEDULE.QUERY.CONTENTID*"	*O*	
"*OBJECT.RECORDTASK*"	*O*	
vendor-defined. Vendor-defined class names MUST obey the rules set forth in Appendix D.3, "Class Name Syntax".	*X*	See Appendix C.1, "Class Hierarchy" for rules on vendor-defined class extensions.

B.1.4 *additionalStatusInfo*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *additionalStatusInfo* property is a general-purpose property that can hold text-based additional status information.

Default Value: N/A – Output only.

Sort Order: Lexical.

Input: N/A.

Output: The current setting.

B.1.5 *cdsReference*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *cdsReference* property MUST only contain metadata of a ContentDirectory service object that is referenced (directly or indirectly) by a *recordSchedule* or *recordTask* object.

Note that this is a multi-valued property so that metadata of multiple referenced ContentDirectory service objects can be stored. A *recordSchedule* or *recordTask* object references ContentDirectory service objects through properties, such as the *scheduledCDSObjectID* property, *recordedCDSObjectID* property, etc. (collectively indicated by the notation: *xxxCDSObjectID* property). To indicate which *cdsReference* property is associated with which *xxxCDSObjectID* property, both properties have a dependent property, *cdsReference@link* and *xxxCDSObjectID@link* respectively, that MUST contain the same unique, vendor-defined link identifier.

The *cdsReference* property MUST contain a *valid* and properly escaped *DIDL-Lite XML Document*. The *DIDL-Lite XML Document* describes a device-dependent (sub)set of imported properties (metadata) of the ContentDirectory service object that is referenced by the linked *xxxCDSObjectID* property. See Appendix B.17, "ContentDirectory Service Imported Properties" for details.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.1.5.1 *cdsReference@link*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *cdsReference@link* contains a unique, vendor-defined link identifier that unambiguously links its *cdsReference* property to a particular *xxxCDSObjectID* property within the same *recordSchedule* or *recordTask* object. See Appendix B.17, "ContentDirectory Service Imported Properties" for details.

Default Value: N/A – Output only.

Sort Order: Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating *cdsReference@link* values. If *cdsReference@link* values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical

Input: N/A.

Output: The current setting.

B.2 Priority Properties

Table B-3: Priority Properties

Property Name	NS	Data Type	M-Val	Reference
priority	srs	xsd:string	*NO*	Appendix B.2.1

Property Name	NS	Data Type	M-Val	Reference
priority@orderedValue	srs	xsd:unsignedInt	*NO*	Appendix B.2.1.2
desiredPriority	srs	xsd:string	*NO*	Appendix B.2.2

B.2.1 *priority*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *priority* property indicates the priority level of the associated object (a *recordSchedule* or a *recordTask*). The `priority-value` format syntax of the *priority* property is described in Appendix D, "EBNF Syntax Definitions".

Example values for this property include: "*L1*", "*L2*", "*L3*", … where "*L1*" represents the highest priority level with subsequent values representing progressively lower priority levels.

Note: Desired priority settings are specified via the *desiredPriority* property passed into the *CreateRecordSchedule()* action. See Section 2.8**, "**ScheduledRecording Service Priority Model**"** for details.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on priority order. Ascending: lowest priority first.

Input: N/A.

Output: The current setting.

B.2.1.1 allowedValueList for the *priority* Property

Table B-4: allowedValueList for the *priority* Property

Value	R/O	Description
"*L1*"	*R*	The highest priority level supported by the device.
"*L2*"	*O*	The next progressively lower priority level supported by the device.
…	*O*	Progressively lower priority level supported by the device.
"*L<x>*"	*O*	The lowest priority level supported by the device where <x> is the total number of distinct priority levels supported by the device.

Notes:
All devices MUST support 1 or more priority levels.

If "*L<x>*" is supported, then all values between "*L1*" and "*L<x>*" MUST be supported.

B.2.1.2 *priority@orderedValue*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *priority@orderedValue* property indicates the relative numerical priority value of the associated object (a *recordSchedule* or a *recordTask*). A value of 1 indicates that this object is the highest priority object of that object type (that is: of all *recordSchedule* instances or all *recordTask* instances). Other ascending values indicate that the object has a progressively lower priority relative to the other objects of that type. A value of N (where N is the total number of objects of that type) indicates that the

object is the lowest priority object of that type. No two objects of the same type will have the same value for this property.

Note: This property is not evented when the priority of the object changes (for example due to the creation of a new object with a higher priority).

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

B.2.2 *desiredPriority*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *desiredPriority* property specifies the desired priority level of the associated object (a *recordSchedule* or a *recordTask*). The `priority-value` format syntax of the *desiredPriority* property is defined in Appendix D, "EBNF Syntax Definitions".

Except as noted below, the value for this property MUST match one of the allowed values returned by the *GetInputPropertyInfo()* action for this property. The allowed values MUST comply with the table in Appendix B.2.2.1, "allowedValueLists for the *desiredPriority* Property" below. Additionally, if the *priority@orderedValue* property is supported, the *desiredPriority* property can also be set to one of the allowed values listed in Table B-6, "Additional allowedValueList for the *desiredPriority* Property".

Default Value: "*DEFAULT*".

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.2.2.1 allowedValueLists for the *desiredPriority* Property

Table B-5: Primary allowedValueList for the *desiredPriority* Property

Value	R/O	Description
"*DEFAULT*"	*R*	No priority preference. The device itself will determine the object's priority.
"*L1*"	*O*	The highest priority level supported by the device.
"*L2*"	*O*	The next to highest priority level supported by the device.
…	*O*	Progressively lower priority levels between 1 and <x>, "*L3*", "*L4*", etc.
"*L<x>*"	*O*	The lowest priority level supported by the device where <x> is the total number of distinct priority levels supported by the device.

Notes:
All devices MUST support 1 or more priority levels.

If "*L<x>*" is supported, then all values between "*L1*" and "*L<x>*" MUST be continuously supported; that is: a device MUST not support only "*L1*", "*L3*", and "*L5*".

Additionally, if the device supports the *priority@orderedValue* property, then the device MUST also support the following allowed values. Conversely, if any of these allowed values are supported, then the device MUST support the *priority@orderedValue* property. These allowed values provide a mechanism for more precise prioritization control with those devices that support it.

Table B-6: Additional allowedValueList for the *desiredPriority* Property

Value	R/O	Description
"*HIGHEST*"	*R*	The highest level possible. – Same as "*L1_HI*" defined below.
"*LOWEST*"	*R*	The lowest level possible. – Same as "*L<x>_LOW*" defined below.
"*L1_HI*"	*R*	The highest priority possible within the highest priority level.
"*L1_LOW*"	*R*	The lowest priority possible within the highest priority level.
"*L2_HI*"	*R*	The highest priority possible within the next to highest priority level.
"*L2_LOW*"	*R*	The lowest priority possible within the next to highest priority level.
…	*R*	Progressively lower priority levels.
"*L<x>_HI*"	*R*	The highest priority possible within the lowest priority level where <x> is the total number of distinct priority levels supported by the device.
"*L<x>_LOW*"	*R*	The lowest priority possible, but within the lowest priority level where <x> is the total number of distinct priority levels supported by the device.
<@id>	*R*	The next highest priority "slot" immediately higher than (but within the same priority level of) the existing object whose *@id* is specified by <@id>.

Notes:

1. If a device supports the *priority@orderedValue* property, then the device MUST also support these CONDITIONALLY REQUIRED allowed values. Conversely, if any of these allowed values are supported, then the device MUST support the *priority@orderedValue* property.
2. These allowed values provide a mechanism for more precise prioritization control with those devices that support it. If "*L<x>_LOW*" is supported, then all values between "*L1_HI*" and "*L<x>_LOW*" MUST be continuously supported; that is: a device MUST not support only "*L1_HI*", "*L1_LOW*", "*L3_HI*", "*L3_LOW*", "*L5_HI*" and "*L5_LOW*" or only "*L1_HI*", "*L2_HI*", and "*L3_HI*".

B.2.3 *desiredPriority@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: When the *desiredPriority@type* property is set to "*PREDEF*", it indicates that the *desiredPriority* property contains one of the predefined priority labels ("*L1*", "*L2_LOW*", etc.). When set to "*OBJECTID*", it indicates that the *desiredPriority* property contains an object ID (*@id* value).

Default Value: "*PREDEF*".

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.2.3.1 allowedValueLists for the *desiredPriority@type* Property

Table B-7: allowedValueList for the *desiredPriority@type* Property

Value	R/O	Description
"*PREDEF*"	*R*	
"*OBJECTID*"	*R*	

B.3 Output Control Properties

Table B-8: Output Control Properties

Property Name	NS	Data Type	M-Val	Reference
recordDestination	srs	xsd:string	*YES*	Appendix B.3.1
recordDestination@mediaType	srs	CSV (xsd:string)	*NO*	Appendix B.3.1.1
recordDestination@targetURL	srs	xsd:anyURI	*NO*	Appendix B.3.1.2
recordDestination@preference	srs	xsd:unsignedInt	*NO*	Appendix B.3.1.3
desiredRecordQuality	srs	xsd:string	*NO*	Appendix B.3.2
desiredrecordQuality@type	srs	xsd:string	*NO*	Appendix B.3.2.2

B.3.1 *recordDestination*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *recordDestination* property identifies the storage unit to be used for the recording. This identifier, which is to be generated by the ScheduledRecording service, SHOULD be a user-friendly name for the storage unit so that its value is meaningful to a user when displayed.

This is a multi-valued property so that more than one record destination can be specified for a recording. The *recordDestination@preference* property allows the order of preference among multiple record destinations to be specified. If none of the specified record destinations is available at the time of recording, then the recording MUST NOT take place.

Examples: "*Hard Disk Drive*", "*DVD-1*", "*LINE1*", "*AUX-out*" etc.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.1 *recordDestination@mediaType*

Namespace: srs **Property Data Type**: CSV (xsd:string) **Multi-Valued**: *NO*

Description: The *recordDestionation@mediaType* property indicates the type of media that is to be used for the recording.

If the media type of the specified record destination is fixed (not removable), then the value of the *recordDestination@mediaType* property MUST match the actual physical media type of the record destination. This single value can be retrieved through the *GetAllowedValues()* action.

If the media of the specified record destination is manually removable (requires explicit user intervention) then the currently inserted physical media MUST match one of the values in the *recordDestination@mediaType* property. In other words, the specified list of media types indicates those that are acceptable for the recording. If the current physical media does not match one of the acceptable media types, then the recording MUST NOT take place on this record destination.

If the specified record destination supports automatic swapping of media, such as a jukebox recorder, then the *recordDestination@mediaType* property indicates which media type(s) MUST be used for the recording. Recording MUST occur on the available media type that appears earliest in the list. If none of the specified media types is available for recording, then the recording MUST NOT take place on this record destination.

If recording can not take place as described above, then lower preference record destinations MAY be used (see Section B.3.1.3, "*recordDestination@preference*"). The set of allowed values for the *recordDestination@mediaType* property can be retrieved through the *GetAllowedValues()* action.

Examples: "*HDD*", "*DVD-RW*"

Default Value: Vendor-defined.

Sort Order: Sequenced Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.1.1 allowedValueList for the *recordDestination@mediaType* Property

One of the allowed values for the *AVTransport::RecordStorageMedium* state variable MUST be specified. Please refer to the AVTransport service specification for the table of allowed values.

B.3.1.2 *recordDestination@targetURL*

Namespace: srs **Property Data Type**: xsd:anyURI **Multi-Valued**: *NO*

Description: The *recordDestination@targetURL* property MUST contain a URL that identifies the location, such as the location of a directory, where the recorded content is to be stored.

Examples:

"`file:///D:/MyDocuments/MyVideos`"

"`http://10.0.0.1/MyDocuments/MyVideos`"

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.3 *recordDestination@preference*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *recordDestination@preference* property is useful when multiple *recordDestination* properties are specified within the same *recordSchedule* or *recordTask* object. In this case, the values

indicate the preference order of the multiple record destinations. Higher numbers indicate lower preference. The values do not have to be contiguous.

If multiple *recordDestination@preference* properties have the same value, then the order of preference in which their associated record destinations are chosen is device-dependent.

If the *recordDestination@preference* property is not supported by an implementation, then the order of preference of all specified record destinations is device-dependent.

Default Value: Vendor-defined.

Sort Order: Numeric.

Input: The desired setting.

Output: The current setting.

B.3.2 *desiredRecordQuality*

Namespace: srs **Property Data Type**: CSV (xsd:string) **Multi-Valued**: *NO*

Description: The *desiredRecordQuality* property is used to express the desired or preferred recording quality level(s) for a particular *recordSchedule*. Multiple recording quality levels can be specified in the comma-separated value list of the *desiredRecordQuality* property. If there is more than one value specified, then the values indicate the desired recording quality, in order of preference, highest preference first. The value "*AUTO*" MUST be supported by all implementations. When "*AUTO*" is included in the list, it MUST appear as the last value in the list and indicates that if none of the preceding values are available, then the ScheduledRecording service is free to use any recording quality level to maximize the probability that the recording actually takes place. When the "*AUTO*" value is the only value in the list, then the ScheduledRecording service is free to use any recording quality level.

There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The *desiredRecordQuality* property is used in conjunction with the *desiredRecordQuality@type* to allow implementations to express these variations. However, since the *desiredRecordQuality* property can appear only once, the acceptable recording quality levels for a particular *recordSchedule* are restricted to a single type variation.

If an implementation is capable of encoding or transcoding, then it MAY do so in order to achieve the desired recording quality.

Example: Assume a (hypothetical) implementation that supports the type variations "*DEFAULT*", "*ATSC*" and "*QLEVEL*" for the *desiredRecordQuality@type* property. The following table expresses the supported *desiredRecordQuality* property values for those variations and also indicates how the different type variations interrelate for this particular implementation:

Table B-9: *desiredRecordQuality* Example

"*DEFAULT*"	"*ATSC*"	"*QLEVEL*"
"*HD*"	"*1080p30*"	"*Q1*"
	"*1080p24*"	
	"*1080i60*"	
	"*720p60*"	"*Q2*"
	"*720p30*"	
	"*720p24*"	
"*ED*"	"*480p60*"	"*Q3*"
"*SD*"	"*480p30*"	
	"*480p24*"	
	"*480i60*"	
"*AUTO*"	"*AUTO*"	"*AUTO*"

- Specifying "*HD,ED*" in the *desiredRecordQuality* property and "*DEFAULT*" in the *desiredRecordQuality@type* property will result in the following:
  - If possible, the recording will be made using "*HD*" quality. In this case, it is up to the implementation to determine exactly which recording quality level within the "*HD*" range will be used for the recording.
  - If recording using "*HD*" quality is not possible, the recording will be made using "*ED*" quality, if possible. Again, it is up to the implementation to determine exactly which recording quality level within the "*ED*" range will be used for the recording.
  - If the recording cannot be made in either "*HD*" or "*ED*" quality, then no recording will be made.
- Specifying "*ED,SD,AUTO*" in the *desiredRecordQuality* property and "*DEFAULT*" in the *desiredRecordQuality@type* property will result in the following:
  - If possible, the recording will be made using "*ED*" quality. It is up to the implementation to determine exactly which recording quality level within the "*ED*" range will be used for the recording.
  - If that is not possible, the recording will be made using "*SD*" quality, if possible. It is up to the implementation to determine exactly which recording quality level within the "*SD*" range will be used for the recording.
  - If the recording cannot be made in either "*ED*" or "*SD*" quality, then the recording will be made using any other available recording quality.
- Specifying "*720p60*" in the *desiredRecordQuality* property and "*ATSC*" in the *desiredRecordQuality@type* property will result in the following:
  - If possible, the recording will be made using "*720p60*" quality.
  - If that is not possible, no recording will be made.

When the ScheduledRecording service responds to a *GetAllowedValues()* action with *desiredRecordQuality* information, then the allowed values MUST be listed in order of quality from highest quality to lowest. The value "*AUTO*" MUST always be present and appear as the last item in the list.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on the allowedValueList for the *desiredRecordQuality* property. Ascending: lowest quality first.

Input: The desired setting.

Output: The current setting.

B.3.2.1 allowedValueList for the *desiredRecordQuality* Property

Table B-10: allowedValueList for the *desiredRecordQuality* Property

Value	R/O	Description
"*AUTO*"	*R*	If none of the quality levels preceding the "*AUTO*" value are available, then any recording quality level may be used. The "*AUTO*" value MUST always appear last in the list when present.
Vendor-defined	*X*	

B.3.2.2 *desiredRecordQuality@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The *desiredRecordQuality@type* property is used to express which type variation is used in its associated independent *desiredRecordQuality* property. The "*DEFAULT*" value MUST be supported and indicates which of the supported type variations is preferred by the device when expressing recording quality levels.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.2.2.1 allowedValueList for the *desiredRecordQuality@type* Property

Table B-11: allowedValueList for the *desiredRecordQuality@type* Property

Value	R/O	Description
"*DEFAULT*"	*R*	Indicates the type variation that is preferred by the device when expressing recording quality levels.
Vendor-defined	*X*	

B.4 Content Identification Related Properties

Table B-12: Content Identification Related Properties

Property Name	NS	Data Type	M-Val	Reference
scheduledCDSObjectID	srs	xsd:string	*NO*	Appendix B.4.1
scheduledCDSObjectID@link	srs	xsd:string	*NO*	Appendix B.4.1.1
scheduledChannelID	srs	xsd:string	*NO*	Appendix B.4.2
scheduledChannelID@type	srs	xsd:string	*NO*	Appendix B.4.2.1
scheduledStartDateTime	srs	xsd:string	*YES*	Appendix B.4.3
scheduledDuration	srs	xsd:string	*NO*	Appendix B.4.4
scheduledProgramCode	srs	xsd:string	*NO*	Appendix B.4.5
scheduledProgramCode@type	srs	xsd:string	*NO*	Appendix B.4.5.1

B.4.1 *scheduledCDSObjectID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledCDSObjectID* property contains the *didl-lite:@id* property value of the ContentDirectory service object from which relevant metadata information is extracted to create the *recordSchedule*.

Default Value: N/A – Required on input.

Sort Order: Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating *didl-lite:@id* values. If *didl-lite:@id* values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical.

Input: The desired setting.

Output: The current setting.

B.4.1.1 *scheduledCDSObjectID@link*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledCDSObjectID@link* contains a unique, vendor-defined link identifier that unambiguously links its *scheduledCDSObjectID* property to a particular *cdsReference* property instance within the same *recordSchedule* object. See Appendix B.17, "ContentDirectory Service Imported Properties" for details.

Default Value: N/A – Output only.

Sort Order: Same as *cdsReference@link*.

Input: N/A.

Output: The current setting.

B.4.2 *scheduledChannelID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledChannelID* property provides channel information for the *recordSchedule*. Its format depends on the *scheduledChannelID@type* property as follows:

If *scheduledChannelID@type* = "*ANALOG*" then the *scheduledChannelID* property contains the (analog) channel number.

Examples: "5", "7", etc.

If *scheduledChannelID@type* = "*DIGITAL*" then the *scheduledChannelID* property contains the (digital) channel number pair "<Major Channel Number>,<Minor Channel Number>".

Examples: "5,1", "5,2", etc.

If *scheduledChannelID@type* = "*FREQUENCY*" then the *scheduledChannelID* property contains the channel center frequency, expressed in Hz.

Examples: "150125000" (VHF band), "615000000" (UHF band), "96500000" (FM band), etc.

If *scheduledChannelID@type* = "*SI*" then the *scheduledChannelID* property contains the Service Information Triplet "<Network ID>,<Transport Stream ID>, <Service ID>", embedded in the content stream.

Examples: "0x1234,0xFEDC,0x0102", "12345,23456,32109", etc.

If *scheduledChannelID@type* = "*LINE*" then the *scheduledChannelID* property contains a vendor-defined label identifying the line input.

Examples: "Line 1", "AUX", "Front", "Rear", etc.

If *scheduledChannelID@type* = "*NETWORK*" then the *scheduledChannelID* property contains the URI that uniquely identifies the content to be recorded.

Examples: "`http://upnp-server/stream1.mp2/`", "`http://internet/stream2.mp2/`"

Default Value: N/A – Required on input.

Sort Order: *type* Relationship.

"*ANALOG*": Numeric.

"*DIGITAL*": Sequenced numeric.

"*FREQUENCY*": Numeric.

"*SI*": Sequenced lexical.

"*LINE*": Lexical.

"*NETWORK*": Lexical.

Vendor-defined: Vendor-defined sorting.

Input: The desired setting.

Output: The current setting.

B.4.2.1 *scheduledChannelID@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledChannelID@type* property determines the format that is used for the *scheduledChannelID* property as defined above.

Default Value: N/A – Required on input.

Sort Order: Property Specific, based on the order in Table B-13. Ascending: first table entry first. If there is a single vendor-defined value, it sorts in table position. If there are multiple vendor-defined values, they

sort lexically among themselves, all after the Table B-13 entries in ascending order and all before the Table B-13 entries in descending order.

Input: The desired setting.

Output: The current setting.

B.4.2.1.1 allowedValueList for the *scheduledChannelID@type* Property

Table B-13: allowedValueList for the *scheduledChannelID@type* Property

Value	R/O	Description
"*ANALOG*"	*O*	At least one value in these rows MUST be supported by a compliant ScheduledRecording service implementation. Control points should support *all* values in these rows.
"*DIGITAL*"	*O*	
"*FREQUENCY*"	*O*	
"*SI*"	*O*	
"*LINE*"	*O*	
"*NETWORK*"	*O*	
Vendor-defined	*X*	

B.4.3 *scheduledStartDateTime*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *scheduledStartDateTime* property indicates what date or day(s) and time the recording will take place. This property does not account for any recording time adjustments such as *scheduledStartDateTimeAdjust*, and device latencies.

The `sched-start` format syntax of the *scheduledStartDateTime* property is defined in Appendix D, "EBNF Syntax Definitions".

Examples: "02-07T15:30:00" (February 7th, 3:30pm), "2005-02-07T15:30:00" (February 7th, 2005, 3:30pm), "MONT15:30:00" (Mondays at 3:30pm), "T15:30:00" (Every day at 3:30pm)

Recording(s) will occur on the next occurrence(s) of the specified date or day(s) and time until the total number of desired recordings (as indicated by the *totalDesiredRecordTasks* property) has been made.

Note that the *scheduledStartDateTime* property is a multi-valued property. Therefore, multiple date&times can be specified for the same *recordSchedule*. Recording will occur on every next occurrence of any of the specified start date&times until the total number of desired recordings (as indicated by the *totalDesiredRecordTasks* property) has been made.

See Appendix B.7.1, "*totalDesiredRecordTasks*" for further details on the use of the *totalDesiredRecordTasks* property.

The value "*NOW*" is defined by this specification to indicate that the recording MUST start immediately (as soon as possible).

Default Value: N/A – Required on input.

Sort Order: Property Specific, in chronological order.

Input: The desired setting.

Output: The current setting.

B.4.4 *scheduledDuration*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledDuration* property indicates the scheduled duration of the recording. The `duration` format syntax of the *scheduledDuration* property is defined in Appendix D, "EBNF Syntax Definitions".

Examples: "P01:30:00" (one hour and thirty minutes), "P2D01:15:00" (two-day and seventy five minutes recording).

This property does not necessarily represent the exact recording duration but represents the scheduled recording duration. This property does not account for any recording time adjustments such as *scheduledDurationAdjust*, and device latencies.

Default Value: N/A – Required on input.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Input: The desired setting.

Output: The current setting.

B.4.5 *scheduledProgramCode*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledProgramCode* property indicates the program code provided by a program guide service of a particular program item. The format of the program code is defined by the program guide service. A *scheduledProgramCode@type* property MUST be specified with this property to identify the program guide service used.

Default Value: N/A – Required on input.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.4.5.1 *scheduledProgramCode@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledProgramCode@type* property indicates the type of the program guide service that defines the program code specified in the *scheduledProgramCode* property. The format of this property is "<ICANN registered domain>" "_" "<program code name>".

Example: "*epg.com_GuideCode*".

Default Value: N/A – Required on input.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.5 Matching Content Criteria Properties

Table B-14: Matching Content Criteria Properties

Property Name	NS	Data Type	M-Val	Reference
matchingName	srs	xsd:string	*NO*	Appendix B.5.1
matchingName@type	srs	xsd:string	*NO*	Appendix B.5.1.1
matchingName@subStringMatch	srs	xsd:boolean	*NO*	Appendix B.5.1.2
matchingID	srs	xsd:string	*NO*	Appendix B.5.2
matchingID@type	srs	xsd:string	*NO*	Appendix B.5.2.1

B.5.1 *matchingName*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingName* property contains (part of) the name of a program or series. To match the criteria of this *recordSchedule*, an external item's name information MUST (partially) match the specified *matchingName* value. Matching SHOULD be done using lexical matching (see Section 2.2.2.27, "Lexical Matching"). It MAY be done using simple non-case-sensitive matching (see Section 2.2.2.29, "Simple Non-case-sensitive Matching").

Example: "NFL Worldcup 2005", "Friends".

Default Value: N/A – Required on input.

Sort Order: Lexical.

Note: This is an exception to the normal rule of *type* Relationship sorting. The equivalent of *type* Relationship sorting may be achieved by including "`+srs:matchingName@type`" in the sort property list immediately in front of "`+srs:matchingName`".

Input: The desired setting.

Output: The current setting.

B.5.1.1 *matchingName@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: IF set to "*PROGRAM*", then the *matchingName* property contains a program name. If set to "*SERIES*", then the *matchingName* property contains a series name.

Default Value: N/A – Required on input.

Sort Order: Property Specific, based on the order in Table B-15. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting.

B.5.1.1.1 allowedValueList for the *matchingName@type* Property

Table B-15: allowedValueList for the *matchingName@type* Property

Value	R/O	Description
"*PROGRAM*"	*R*	

Value	R/O	Description
"*SERIES*"	*R*	

B.5.1.2 *matchingName@subStringMatch*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: If set to "*1*", the value specified in the *matchingName* property is used for a substring match search within the program or series name (title). If set to "*0*" the value specified in the *matchingName* property must match the program or series name exactly.

Default Value: "*1*".

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.5.2 *matchingID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingID* property contains the unique ID of a program or series. To match the criteria of this *recordSchedule*, an external item's ID information MUST match the specified *matchingID* value.

If the *matchingID@type* property is set to "*SI_PROGRAMID*", then the *matchingID* property is formatted as follows:

"<Network ID>,<Transport Stream ID>,<Service ID>,<Program ID>".

If the *matchingID@type* property is set to "*SI_SERIESID*", then the *matchingID* property is formatted as follows:

"<Network ID>,<Transport Stream ID>,<Service ID>,<Series ID>".

If the *matchingID@type* property is set to *<ICANN Name>*, then the *matchingID* property is formatted as follows:

"<Unique content ID, defined by the data provider>".

Default Value: N/A – Required on input.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.5.2.1 *matchingID@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingID@type* property indicates the type of the ID that is contained in the *matchingID* property.

Default Value: N/A – Required on input.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.5.2.1.1 allowedValueList for the *matchingID@type* Property

Table B-16: allowedValueList for the *matchingID@type* Property

Value	R/O	Description
"*SI_PROGRAMID*"	*R*	
"*SI_SERIESID*"	*R*	
<ICANN Name>_<Identifier>	*O*	<ICANN Name>: The ICANN name of the organization that defines the format and values of the *matchingID* property. <Identifier>: A unique identifier for the particular ID type, defined by that organization. Examples: "mycompany.com_ID1", "upnp.org_SpecialID".

B.6 Matching Qualifying Criteria Properties

Table B-17: Matching Qualifying Criteria Properties

Property Name	NS	Data Type	M-Val	Reference
matchingChannelID	srs	xsd:string	*YES*	Appendix B.6.1
matchingChannelID@type	srs	xsd:string	*NO*	Appendix B.6.1.1
matchingStartDateTimeRange	srs	xsd:string	*YES*	Appendix B.6.2
matchingDurationRange	srs	xsd:string	*YES*	Appendix B.6.3
matchingRatingLimit	srs	xsd:string	*YES*	Appendix B.6.4
matchingRatingLimit@type	srs	xsd:string	*NO*	Appendix B.6.4.2
matchingEpisodeType	srs	xsd:string	*NO*	Appendix B.6.5

B.6.1 *matchingChannelID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *matchingChannelID* property contains a *scheduledChannelID* value. Its format depends on the *matchingChannelID@type* property. To match the criteria of this *recordSchedule*, an external item's channel information (after translation into the format of a *scheduledChannelID* property) MUST match one of the specified *matchingChannelID* values. If this property is omitted from the *recordSchedule*, the external item's channel information is not taken into consideration to determine a match.

Default Value: N/A – Not used if omitted on input.

Sort Order: Same as *scheduledChannelID*.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.1.1 *matchingChannelID@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingChannelID@type* property determines the format that is used for the *matchingChannelID* property as defined in Appendix B.4.2, "*scheduledChannelID*" and Appendix B.4.2.1, "*scheduledChannelID@type*".

Default Value: N/A – Not used if omitted on input .

Sort Order: Same as *scheduledChannelID@type*.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.2 *matchingStartDateTimeRange*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *matchingStartDateTimeRange* property contains a date range. The `start-range` format syntax of the *matchingStartDateTimeRange* property is defined inAppendix D, "EBNF Syntax Definitions".

The value specified after the "/" MUST be equal or greater than the value specified before the "/".

To match the criteria of this *recordSchedule*, an external item's start date and time information MUST fall within one of the specified *matchingStartDateTimeRange* ranges. If this property is omitted from the *recordSchedule*, the external item's start date and time information is not taken into consideration to determine a match.

Note: The *matchingStartDateTimeRange* property is different from the *activePeriod* property in that the first identifies the actual matching criteria whereas the second identifies the period of time when potential matches are to be examined.

Default Value: N/A – Not used if omitted on input.

Sort Order: Sequenced Sort of two date&time subvalues separated by "/".

Both subvalues are sorted in chronological order.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.3 *matchingDurationRange*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

The *matchingDurationRange* property contains a duration range. The `duration-range` format syntax of the *matchingDurationRange* property is defined in Appendix D, "EBNF Syntax Definitions".

The value specified after the "/" MUST be equal or greater than the value specified before the "/".

To match the criteria of this *recordSchedule*, an external item's duration information (after translation into the format of a *scheduledDuration* property) MUST fall within the specified *matchingDurationRange* range. If this property is omitted from the *recordSchedule*, the external item's duration information is not taken into consideration to determine a match.

Default Value: N/A – Not used if omitted on input.

Sort Order: Sequenced Sort of two duration subvalues separated by "/".

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.4 *matchingRatingLimit*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *matchingRatingLimit* property indicates a maximum allowed rating. Several different rating systems are available. The rating system is indicated in the *matchingRatingLimit@type* property. The allowed values for the *matchingRatingLimit* property depend on the rating system used.

Common rating systems as well as their allowed rating values (in order of ascending restriction level beginning with the most lenient) for each rating system are defined below.

Other values MAY be specified using other rating systems identified by their ICANN domain names.

To match the criteria of this *recordSchedule*, an external item's rating information MUST be less than or equal to all of the specified *matchingRatingLimit* values. If this property is omitted from the *recordSchedule*, the external item's rating information is not taken into consideration to determine a match. If the external item does not contain rating information and this property is specified, the external item will not be recorded.

Default Value: N/A – Not used if omitted on input.

Sort Order: *type* Relationship.

For each value of *matchingRatingLimit@type*, based on the order in the table associated with the *matchingRatingLimit@type* property below. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.4.1 allowedValueList for the *matchingRatingLimit* Property

The allowed values for the *matchingRatingLimit* property depends on the *matchingRatingLimit@type* property. The following tables list the allowed values for each defined rating system.

Table B-18: allowedValueList for the *matchingRatingLimit* Property Using the MPAA Rating System (*matchingRatingLimit@type* = "*MPAA.ORG*")

Value	R/O	Description
"*G*"	*R*	General Audiences.
"*PG*"	*R*	Parental Guidance Suggested.
"*PG-13*"	*R*	Parents Strongly Cautioned.
"*R*"	*R*	Restricted.
"*NC-17*"	*R*	No One 17 and Under Admitted.
"*NR*"	*R*	Not Rated Yet.

Table B-19: allowedValueList for the *matchingRatingLimit* Property Using the RIAA Rating System (*matchingRatingLimit@type* = "*RIAA.ORG*")

Value	R/O	Description
""	*R*	Non-explicit Content
"*PA-EC*"	*R*	Parental Advisory – Explicit Content

Table B-20: allowedValueList for the *matchingRatingLimit* Property Using the ESRB Rating System (*matchingRatingLimit@type* = "*ESRB.ORG*")

Value	R/O	Description
"*EC*"	*R*	Early Childhood.
"*E*"	*R*	Everyone.
"*E10+*"	*R*	Everyone 10 and Older.
"*T*"	*R*	Teen.
"*M*"	*R*	Mature.
"*AO*"	*R*	Adults Only.
"*RP*"	*R*	Rating Pending.

Table B-21: allowedValueList for the *matchingRatingLimit* Property Using the TVGUIDELINES Rating System (*matchingRatingLimit@type* = "*TVGUIDELINES.ORG*")

Value	R/O	Description
"*TV-Y*"	*R*	All Children.
"*TV-Y7*"	*R*	Directed to Older Children.
"*TV-Y7FV*"	*R*	Directed to Older Children – Fantasy Violence.
"*TV-G*"	*R*	General Audience.
"*TV-PG*"	*R*	Parental Guidance Suggested.
"*TV-14*"	*R*	Parents Strongly Cautioned.
"*TV-MA*"	*R*	Mature Audience Only.

B.6.4.2 *matchingRatingLimit@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingRatingLimit@type* property indicates the rating system used in the *matchingRatingLimit* property. Several different rating systems are available. The allowed values for the *matchingRatingLimit* property depend on the rating system used.

Other rating systems MAY be specified using their ICANN domain names.

This is not a matching property. It is used in conjunction with the *matchingRatingLimit* property and identifies the used rating system.

Default Value: N/A – Required in input.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.6.4.2.1 allowedValueList for the *matchingRatingLimit@type* Property

Table B-22: allowedValueList for the *matchingRatingLimit@type* Property

Value	R/O	Description	Remarks
"*MPAA.ORG*"	*O*	The Motion Picture Association of America.	At least one value in these rows MUST be supported by a compliant ScheduledRecording service implementation. Control points should support all values in these rows.
"*RIAA.ORG*"	*O*	The Recording Industry Association of America.	
"*ESRB.ORG*"	*O*	The Entertainment Software Rating Board.	
"*TVGUIDELINES.ORG*"	*O*	TV Parental Guidelines.	
<ICANN Name>_<Identifier>	*X*	<ICANN Name>: The ICANN name of the organization that defines the rating. <Identifier>: A unique identifier for a particular rating system, defined by that organization. Examples: "mycompany.com_RS1", "upnp.org_ratingx".	

B.6.5 *matchingEpisodeType*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchingEpisodeType* property indicates the type of content to be recorded in terms of its broadcast novelty. To match the criteria of this *recordSchedule*, an external item's episode type information MUST match the specified *matchingEpisodeType* value. If this property is omitted from the *recordSchedule*, the external item's episode type information is not taken into consideration to determine a match. If the external item does not contain episode type information and this property is specified, the external item will not be recorded.

Default Value: N/A – Not used if omitted on input.

Sort Order: Property Specific, based on the order in Table B-23. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.5.1 allowedValueList for the *matchingEpisodeType* Property

Table B-23: allowedValueList for the *matchingEpisodeType* Property

Value	R/O	Description
"*ALL*"	*R*	All programs are recorded.
"*FIRST-RUN*"	*R*	Only programs that have an original air date equal to the current date are recorded.

Value	R/O	Description
"*REPEAT*"	*R*	Only programs that have an original air date earlier than the current date are recorded.

B.7 Content Control Properties

Table B-24: Content Control Properties

Property Name	NS	Data Type	M-Val	Reference
totalDesiredRecordTasks	srs	xsd:unsignedInt	*NO*	Appendix B.7.1
scheduledStartDateTimeAdjust	srs	xsd:string	*NO*	Appendix B.7.2
scheduledDurationAdjust	srs	xsd:string	*NO*	Appendix B.7.3
activePeriod	srs	xsd:string	*NO*	Appendix B.7.4
durationLimit	srs	xsd:string	*NO*	Appendix B.7.5
durationLimit@effect	srs	xsd:string	*NO*	Appendix B.7.5.1
channelMigration	srs	xsd:boolean	*NO*	Appendix B.7.6
timeMigration	srs	xsd:boolean	*NO*	Appendix B.7.7
allowDuplicates	srs	xsd:boolean	*NO*	Appendix B.7.8

B.7.1 *totalDesiredRecordTasks*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *totalDesiredRecordTasks* property indicates the maximum number of *recordTask* instances, associated with a given *recordSchedule* that will ever be generated over the lifetime of the *recordSchedule*. A value of 0 means that an unlimited number of *recordTask* instances can be spawned from the *recordSchedule*.

This property is used to enable or disable recurrence. If a value different from 1 is specified in the *totalDesiredRecordTasks* property, then the *recordSchedule* MUST remain active after the first *recordTask* has been spawned and MUST monitor its internal state to determine if the conditions that caused the first *recordTask* to be spawned are met again in the future. Whenever this happens, a new *recordTask* MUST be spawned until the total number of spawned *recordTask* instances reaches the value, specified in the *totalDesiredRecordTasks* property. The *activePeriod* property can be used to terminate this process prematurely.

Default Value: 1 (recurrence is disabled by default).

Sort Order: Numeric.

Input: The desired setting.

Output: The current setting.

B.7.2 *scheduledStartDateTimeAdjust*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledStartDateTimeAdjust* property indicates a time period to be applied as an adjustment to the scheduled start time. The `duration-adj` format syntax of the *scheduledStartDateTimeAdjust* property is defined in Appendix D, "EBNF Syntax Definitions".

Note that the *scheduledStartDateTimeAdjust* property can take on both positive and negative values. Negative values provide pre-roll functionality (notice the + sign in the formula below) whereas positive values allow for starting the recording a certain period of time into the recording. The actual scheduled start time is calculated as:

actualScheduledStartDateTime = *scheduledStartDateTime* + *scheduledStartDateTimeAdjust*

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: from longest negative elapsed time to longest positive elapsed time.

Input: The desired setting.

Output: The current setting.

B.7.3 *scheduledDurationAdjust*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduledDurationAdjust* property indicates a period of time to be applied as an adjustment to the scheduled duration time. The `duration-adj` format syntax of the *scheduledDurationAdjust* property is defined in Appendix D, "EBNF Syntax Definitions".

Note that the *scheduledDurationAdjust* property can take on both positive and negative values. Positive values provide post-roll functionality whereas negative values allow for ending the recording a certain time period before the end of the recording. The actual scheduled end time and actual scheduled duration are calculated as:

actualScheduledEndDateTime = *scheduledStartDateTime* + *scheduledDuration* + *scheduledDurationAdjust*

actualScheduledDuration = *actualScheduledEndDateTime* – *actualScheduledStartDateTime*

= *scheduledDuration* + *scheduledDurationAdjust* – *scheduledStartDateTimeAdjust*

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: from longest negative elapsed time to longest positive elapsed time.

Input: The desired setting.

Output: The current setting.

B.7.4 *activePeriod*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *activePeriod* property indicates the date&time range within which the *recordSchedule* is active; that is: the *recordSchedule* MUST NOT spawn any *recordTask* instances whose *actualStartDateTime* fall outside the period specified in the *activePeriod* property. The `start-range` format syntax of the *activePeriod* property is defined in Appendix D, "EBNF Syntax Definitions".

The value specified after the "/" MUST be equal or greater than the value specified before the "/".

A *recordSchedule* MUST not generate new *recordTask* instances for programs broadcast after the expiration date.

Note: The *activePeriod* property is different from the *matchingStartDateTimeRange* property in that the first identifies the period of time when potential matches are to be examined whereas the second identifies the actual matching criteria.

Default Value: "*NOW/INFINITY*".

Sort Order: Sequenced Sort of two date&time subvalues separated by "/".

Both subvalues are sorted in chronological order.

Input: The desired setting.

Output: The current setting.

B.7.5 *durationLimit*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *durationLimit* property indicates the maximum allowed duration of the recording. The `duration-long` format syntax of the *durationLimit* property is defined in Appendix D, "EBNF Syntax Definitions".

If the actual duration of the recording exceeds the value specified in the *durationLimit* property, then the ScheduledRecording service MUST stop recording and either delete the partially recorded content so far or preserve part of the recorded content depending on the current setting of the *durationLimit@effect* property.

If the *durationLimit* property is set to "*INFINITY*", then no limit is in effect.

Example: the value "P02:30:00" indicates that the recording MUST be stopped after two and a half hours.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Input: The desired setting.

Output: The current setting.

B.7.5.1 *durationLimit@effect*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *durationLimit@effect* property determines the behavior of the *recordSchedule* when the duration of the content exceeds the value specified in the *durationLimit* property.

If set to "*SKIP*", then the partially recorded content is deleted once the *actualDuration* of the recording exceeds the value specified in the *durationLimit* property.

If set to "*LAST*", then only the latest part (in length equal to the value specified in the *durationLimit* property) of the content is preserved, effectively deleting the first part of the recording.

If set to "*FIRST*", then only the initial part (in length equal to the value specified in the *durationLimit* property) of the content is preserved, effectively deleting the last part of the recording.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on the order in Table B-25. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting.

B.7.5.1.1 allowedValueList for the *durationLimit@effect* Property

Table B-25: allowedValueList for the *durationLimit@effect* Property

Value	R/O	Description
"*SKIP*"	*O*	At least one value in these rows MUST be supported. Control points should support all values in these rows.
"*LAST*"	*O*	
"*FIRST*"	*O*	

B.7.6 *channelMigration*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: A program's scheduled channel may change between the time the *recordSchedule* was created and the actual broadcast time. If this property is set to "*1*", then the ScheduledRecording service MUST automatically follow the program if it moves to another channel (The reservation will be tracking broadcast channel change). If this value is set to "*0*", then the ScheduledRecording service does not follow the program, and the recording will take place on the channel that was specified at the time the *recordSchedule* created the accociated *recordTask*.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.7.7 *timeMigration*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: A program's scheduled date&time may change between the time the *recordSchedule* was created and the actual broadcast time. If this property is set to "*1*", then the ScheduledRecording service MUST automatically follow the program if it moves to another date&time (The reservation will be tracking broadcast date&time change). If this value is set to "*0*", then the ScheduledRecording service does not follow the program, and the recording will take place at the date&time that was specified at the time the *recordSchedule* created the accociated *recordTask*.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.7.8 *allowDuplicates*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: If set to "*1*", then programs are recorded, even if a duplicate program has already been recorded as a result of the *recordSchedule*. If set to "*0*", no duplicates are recorded. Detection of duplicate programs is device- and EPG-dependent.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.8 Storage Related Properties

Table B-26: Storage Related Properties

Property Name	NS	Data Type	M-Val	Reference
persistedRecordings	srs	xsd:unsignedInt	*NO*	Appendix B.8.1
persistedRecordings@latest	srs	xsd:boolean	*NO*	Appendix B.8.1.1
persistedRecordings@preAllocation	srs	xsd:boolean	*NO*	Appendix B.8.1.2
persistedRecordings@storedLifetime	srs	xsd:string	*NO*	Appendix B.8.1.3

B.8.1 *persistedRecordings*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *persistedRecordings* property indicates the minimum number of recordings for a given *recordSchedule* that will be preserved at all times, once available. Even when the ScheduledRecording service needs to make space for other recordings, this minimum number of recordings (that is: the actual content) generated by the *recordSchedule* will not be deleted. However, if more recordings, associated with the *recordSchedule* exist, then these *excess* recordings MAY be deleted by the ScheduledRecording service. Whether the oldest or the newest *excess* recordings will be deleted depends on the value of the *persistedRecordings@latest* property.

Default Value: Vendor-defined.

Sort Order: Numeric.

Input: The desired setting.

Output: The current setting.

B.8.1.1 *persistedRecordings@latest*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *persistedRecordings@latest* property indicates whether newest or oldest recordings are preserved. If set to "*1*", then the newest recordings are preserved. The recordings prior to these MAY be deleted when more recent content is recorded.

If set to "*0*", then the oldest recordings are preserved. Older content will never be deleted to make room for newer content.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.8.1.2 *persistedRecordings@preAllocation*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *persistedRecordings@preAllocation* property indicates whether to reserve storage space on beforehand to accommodate for the number of recordings as indicated by the *persistedRecordings* property. If set to "*1*", *adequate* storage space is reserved. To reserve storage space, the ScheduledRecording service calculates a *best estimate* based on parameters such as record quality, start time and duration adjustment etc. However, the ScheduledRecording service can never *guarantee* that sufficient storage space is reserved to accommodate the total number of recordings, specified in the *persistedRecordings* property. If set to "*0*", no storage space is reserved.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.8.1.3 *persistedRecordings@storedLifetime*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *persistedRecordings@storedLifetime* property indicates the minimum time recorded content associated with a *recordSchedule* will be preserved after the recording completes. This will prohibit a recording from being deleted by the auto-delete operation within the specified time period. The `duration-any` format syntax of the *persistedRecordings@storedLifetime* property is defined in Appendix D, "EBNF Syntax Definitions".

If the value is set to "*INFINITY*", then the content MUST never be automatically deleted.

A value of "*ANY*" indicates that the content can be deleted at any time by the auto-delete operation. However, it is RECOMMENDED that a ScheduledRecording service implementation only deletes content when space is needed.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first. "*ANY*" is considered the shortest elapsed time possible; "*INFINITY*" is considered the longest elapsed time possible.

Input: The desired setting.

Output: The current setting.

B.9 Schedule State Properties

Table B-27: Schedule State Properties

Property Name	NS	Data Type	M-Val	Reference
scheduleState	srs	xsd:string	*NO*	Appendix B.9.1
scheduleState@currentErrors	srs	CSV (xsd:int)	*NO*	Appendix B.9.1.2
abnormalTasksExist	srs	xsd:boolean	*NO*	Appendix B.9.2

B.9.1 *scheduleState*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *scheduleState* property indicates the overall state of the *recordSchedule* itself.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on the order in Table B-28. Ascending: first table entry first.

Input: N/A.

Output: The current setting.

B.9.1.1 allowedValueList for the *scheduleState* Property

Table B-28: allowedValueList for the *scheduleState* Property

Value	R/O	Description
"*OPERATIONAL*"	*R*	*recordSchedule* is operating and spawning *recordTask* instances as scheduled.
"*COMPLETED*"	*R*	*recordSchedule* is completed and reached final disposition. No properties will change.
"*ERROR*"	*R*	*recordSchedule* ceases spawning *recordTask* instances due to error.

B.9.1.2 *scheduleState@currentErrors*

Namespace: srs **Property Data Type**: CSV (xsd:int) **Multi-Valued**: *NO*

Description: The *scheduleState@currentErrors* property indicates the current error(s) that cause the schedule to be in the "*ERROR*" state. This error list pertains specifically to the behavior of a *recordSchedule* and describes a recordSchedule's inability to create new tasks. When the *srs scheduleState* property has the value "*OPERATIONAL*", the *scheduleState@currentErrors* property MUST be empty. The list of error codes are listed in the *recordSchedule* error code section.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.9.1.2.1 allowedValueList for the *scheduleState@currentErrors* Property

Table B-29: allowedValueList for the *scheduleState@currentErrors* Property

Value	R/O	Description
0-99	*N/A*	Reserved
100	*R*	General error – an error is detected but the cause can not be identified.
101	*O*	The number of spawned *recordTask* instances has reached some device dependent limit.
102	*O*	EPG information not available.
103	*O*	*recordSchedule* is disabled by the user.
104	*O*	Insufficient memory – The system does not have enough system memory to create any additional *recordTask* instances.
105	*O*	General resource error – some system related resource is causing the *recordSchedule* to malfunction.
106-149	*O*	Reserved for future *recordSchedule* error codes.
150-199	*X*	Vendor extended *recordSchedule* error codes.

Value	R/O	Description
200 and above	*N/A*	Reserved for future extensions.

B.9.2 *abnormalTasksExist*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: If this property is set to "*1*", then that indicates that at least one abnormal *recordTask* exists for the *recordSchedule*. If this property is set to "*0*", then no abnormal *recordTask* exists for the *recordSchedule*. A *recordTask* is considered abnormal if it reaches any state other than "*IDLE.READY*", "*ACTIVE.RECORDING.FROMSTART.OK*" or "*DONE.FULL*".

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.10 Statistics Properties

Table B-30: Statistics Properties

Property Name	NS	Data Type	M-Val	Reference
currentRecordTaskCount	srs	xsd:unsignedInt	*NO*	Appendix B.10.1
totalCreatedRecordTasks	srs	xsd:unsignedInt	*NO*	Appendix B.10.2
totalCompletedRecordTasks	srs	xsd:unsignedInt	*NO*	Appendix B.10.3

B.10.1 *currentRecordTaskCount*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *currentRecordTaskCount* property indicates the number of existing *recordTask* instances that are currently associated with a given *recordSchedule*. Previously generated *recordTask* instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service are not taken into account.

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

B.10.2 *totalCreatedRecordTasks*

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

Description: The *totalCreatedRecordTasks* property indicates how many *recordTask* instances have been created during the lifetime of the associated *recordSchedule*. This includes previously generated *recordTask* instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service.

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

B.10.3 totalCompletedRecordTasks

Namespace: srs **Property Data Type**: xsd:unsignedInt **Multi-Valued**: NO

Description: The totalCompletedRecordTasks property indicates how many recordTask instances have been completed (that is: reached any of the "DONE.xxx" states, during the lifetime of the associated recordSchedule. This includes previously generated recordTask instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service.

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

B.11 Task General Properties

Table B-31: Task General Properties

Property Name	NS	Data Type	M-Val	Reference
recordScheduleID	srs	xsd:string	NO	Appendix B.11.1
recordedCDSObjectID	srs	xsd:string	NO	Appendix B.11.2
recordedCDSObjectID@link	srs	xsd:string	NO	Appendix B.11.2.1

B.11.1 recordScheduleID

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: NO

Description: The recordScheduleID property contains the value of the @id property of the recordSchedule that generated the recordTask.

Default Value: N/A – Output only.

Sort Order: Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating @id values. If @id values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical.

Input: N/A.

Output: The current setting.

B.11.2 recordedCDSObjectID

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: NO

Description: The recordedCDSObjectID property contains the didl-lite:@id property value of the ContentDirectory service object that represents the content recorded by the recordTask.

Default Value: N/A – Output only.

Sort Order Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating *didl-lite:@id* values. If *didl-lite:@id* values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical.

Input: N/A.

Output: The current setting.

B.11.2.1 *recordedCDSObjectID@link*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *recordedCDSObjectID@link* contains a unique, vendor-defined link identifier that unambiguously links its *recordedCDSObjectID* property to a particular *cdsReference* property instance within the same *recordTask* object. See Appendix B.17, "ContentDirectory Service Imported Properties" for details.

Default Value: N/A – Output only.

Sort Order: Same as *cdsReference@link* property.

Input: N/A.

Output: The current setting.

B.12 Task Content Identification Properties

Table B-32: Task Content Identification Properties

Property Name	NS	Data Type	M-Val	Reference
taskCDSObjectID	srs	xsd:string	*NO*	Appendix B.12.1
taskCDSObjectID@link	srs	xsd:string	*NO*	Appendix B.12.1.1
taskChannelID	srs	xsd:string	*NO*	Appendix B.12.2
taskChannelID@type	srs	xsd:string	*NO*	Appendix B.12.2.1
taskStartDateTime	srs	xsd:string	*NO*	Appendix B.12.3
taskDuration	srs	xsd:string	*NO*	Appendix B.12.4
taskProgramCode	srs	xsd:string	*NO*	Appendix B.12.5
taskProgramCode@type	srs	xsd:string	*NO*	Appendix B.12.5.1
recordQuality	srs	xsd:string	*YES*	Appendix B.12.6
recordQuality@type	srs	xsd:string	*NO*	Appendix B.12.6.2

B.12.1 *taskCDSObjectID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskCDSObjectID* property contains the *didl-lite@id* property value of the ContentDirectory service object from which relevant metadata information was extracted to create the *recordSchedule* that generated this *recordTask*.

Default Value: N/A – Output only.

Sort Order: Lexical or Lexical Numeric.

Each implementation SHOULD use the sort method most appropriate for its method of generating *didl-lite:@id* values. If *didl-lite:@id* values contain a numeric (sub)string that contains values that increment with each new object creation, then use Lexical Numeric; otherwise, use Lexical.

Input: N/A.

Output: The current setting.

B.12.1.1 *taskCDSObjectID@link*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskCDSObjectID@link* contains a unique, vendor-defined link identifier that unambiguously links its *taskCDSObjectID* property to a particular *cdsReference* property instance within the same *recordTask* object. See Appendix B.17, "ContentDirectory Service Imported Properties" for details.

Default Value: N/A – Output only.

Sort Order: Same as *cdsReference@link* property.

Input: N/A.

Output: The current setting.

B.12.2 *taskChannelID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskChannelID* property indicates the actual channel that is used for the recording. Its format depends on the *taskChannelID@type* property. The possible formats and the dependency on the *taskChannelID@type* property are identical to the possible formats of the *scheduledChannelID* and its dependency on the *scheduledChannelID@type* property as described in Appendix B.4.2, "*scheduledChannelID*" and Appendix B.4.2.1, "*scheduledChannelID@type*".

Default Value: N/A – Output only.

Sort Order: Same as *scheduledChannelID* property.

Input: N/A.

Output: The current setting.

B.12.2.1 *taskChannelID@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskChannelID@type* property determines the format that is used for the *taskChannelID* property as defined above. See Appendix B.4.2.1, "*scheduledChannelID@type*" for detaisl and allowed values.

Default Value: N/A – Output only.

Sort Order: Same as *scheduledChannelID@type* property.

Input: N/A.

Output: The current setting.

B.12.3 *taskStartDateTime*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskStartDateTime* property indicates the actual start date&time (based on the current information) of the recording. This date&time does not include any adjustments. These are reflected in the *taskStartDateTimeAdjust* property. The `date-time` format syntax of the *taskStartDateTime* property is defined in Appendix D, "EBNF Syntax Definitions".

Default Value: N/A – Output only.

Sort Order: Property Specific, in chronological order.

Input: N/A.

Output: The current setting.

B.12.4 *taskDuration*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskDuration* property indicates the actual duration (based on the current information) of the recording. This duration does not include any adjustments. These are reflected in the *taskDurationAdjust* property. The `duration` format syntax of the *taskDuration* property is defined in Appendix D, "EBNF Syntax Definitions".

Default Value: N/A – Output only.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Input: N/A.

Output: The current setting.

B.12.5 *taskProgramCode*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskProgramCode* property indicates the actual program code that is used for the recording. The format is identical to the format of the *scheduledProgramCode* property. See Appendix B.4.5, "*scheduledProgramCode*" for details.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.12.5.1 *taskProgramCode@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskProgramCode@type* property indicates the type of the program guide service that defines the program code specified in the *taskProgramCode* property. The format is identical to the format of the *scheduledProgramCode@type* property. See Appendix B.4.5.1, "*scheduledProgramCode@type*" for details.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.12.6 *recordQuality*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *recordQuality* property expresses the recording quality level that is used for a particular *recordTask*.

When the *recordTask* is in the "*IDLE*" phase (the actual recording has not started yet), this property contains a best-known estimate of the recording quality for the recording. Therefore, the *recordQuality* property MUST contain one of the vendor-defined values supported by the ScheduledRecording service. The value "*AUTO*" is not allowed. If the implementation does not have enough information to generate a value with some accuracy, then the value "*UNKNOWN*" MUST be used.

When the *recordTask* is in the "*ACTIVE*" or "*DONE*" phase, the *recordQuality* property MUST contain one of the values supported by the implementation, that describes the actual recording quality. The values "*AUTO*" and "*UNKNOWN*" are not allowed.

There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The *recordQuality* property is used in conjunction with the *recordQuality@type* to allow implementations to express these type variations.

For each type variation, the allowed values for the *recordQuality* property MUST be the same as the allowed values supported for the corresponding type variation of the *desiredRecordQuality* property, except that "*UNKNOWN*" replaces "*AUTO*".

Note that the *recordQuality* property is a multi-valued property. Therefore, the actual recording quality level can be expressed using different type variations simultaneously. As a baseline, all implementations MUST support type variation "*DEFAULT*". All record quality levels expressed in a certain type variation MUST have equivalent quality levels expressed in all other type variations, supported by the implementation. If an implementation supports multiple type variations to express recording quality, then it MUST provide the recording quality level expressed in all supported type variations.

Example: Assume a (hypothetical) implementation that supports the type variations "*DEFAULT*", "*ATSC*" and "*QLEVEL*" for the *recordQuality@type* property. The following table expresses the supported *recordQuality* property values for those variations and also indicates how the different type variations interrelate for this particular implementation:

Table B-33: *recordQuality* Example

"*DEFAULT*"	"*ATSC*"	"*QLEVEL*"
"*HD*"	"*1080p30*"	"*Q1*"
	"*1080p24*"	
	"*1080i60*"	
	"*720p60*"	"*Q2*"
	"*720p30*"	
	"*720p24*"	
"*ED*"	"*480p60*"	"*Q3*"
"*SD*"	"*480p30*"	
	"*480p24*"	
	"*480i60*"	
"*UNKNOWN*"	"*UNKNOWN*"	"*UNKNOWN*"

- Assuming the actual recording quality of a *recordTask* is "*720p60*" (as an example), then the *recordTask* object MUST include three instances of the *recordQuality* property as illustrated by the following XML fragment:

```
<recordQuality type="DEFAULT">HD</recordQuality>
<recordQuality type="ATSC">720p60</recordQuality>
<recordQuality type="QLEVEL">Q2</recordQuality>
```

- Assuming the actual recording quality of a *recordTask* is "*480p60*", then the *recordTask* object MUST include three instances of the *recordQuality* property as illustrated by the following XML fragment:

```
<recordQuality type="DEFAULT">ED</recordQuality>
<recordQuality type="ATSC">480p60</recordQuality>
<recordQuality type="QLEVEL">Q3</recordQuality>
```

When the ScheduledRecording service responds to a *GetAllowedValues()* action with *recordQuality* information, then the allowed values MUST be listed in order of quality from highest quality to lowest.

Default Value: N/A – Output only.

Sort Order: *type* Relationship.

Input: N/A.

Output: The current setting.

B.12.6.1 allowedValueList for the *recordQuality* Property

Table B-34: allowedValueList for the *recordQuality* Property

Value	R/O	Description
"*UNKNOWN*"	*R*	The recording quality is unknown by the ScheduledRecording service. Only applicable when the *recordTask* is in the "*IDLE*" phase.
Vendor-defined	*X*	

B.12.6.2 *recordQuality@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The *recordQuality@type* property is used to express which type variation is used in its associated independent *recordQuality* property. The "*DEFAULT*" value MUST be supported.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.13 Task Matched Content Criteria Properties

Table B-35: Task Matched Content Criteria Properties

Property Name	NS	Data Type	M-Val	Reference
matchedName	srs	xsd:string	*NO*	Appendix B.13.1
matchedName@type	srs	xsd:string	*NO*	Appendix B.13.1.1
matchedID	srs	xsd:string	*NO*	Appendix B.13.2
matchedID@type	srs	xsd:string	*NO*	Appendix B.13.2.1

B.13.1 *matchedName*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchedName* property contains the full matched name of a program or series. This is the full program or series name of the external item that (partially) matched the name specified in the *matchingName* property of the *recordSchedule*.

Default Value: N/A – Output only.

Sort Order: Same as *matchingName* property.

Input: N/A.

Output: The current setting.

B.13.1.1 *matchedName@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: IF set to "*PROGRAM*", then the *matchedName* property contains a program name. If set to "*SERIES*", then the *matchedName* property contains a series name. The format is identical to the format of the *matchingName@type* property. See Appendix B.5.1.1, "*matchingName@type*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *matchingName@type* property.

Input: N/A.

Output: The current setting.

B.13.2 *matchedID*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchedID* property contains the matched ID of a program or series. This is the ID of the external item that matched the ID specified in the *matchingID* property of the *recordSchedule*. The format is identical to the format of the *matchingID* property. See Appendix B.5.2, "*matchingID*" for details.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.13.2.1 *matchedID@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchedID@type* property indicates the type of the ID that is contained in the *matchedID* property. The format of this property is identical to the format of the matchingID@type property. See Appendix B.5.2.1, "*matchingID@type*" for details.

Default Value: N/A – Output only.

Sort Order: Sorting on this property is meaningless and will be ignored.

Input: N/A.

Output: The current setting.

B.14 Task Matched Qualifying Criteria Properties

Table B-36: Task Matched Qualifying Criteria Properties

Property Name	NS	Data Type	M-Val	Reference
matchedRating	srs	xsd:string	*YES*	Appendix B.14.1
matchedRating@type	srs	xsd:string	*NO*	Appendix B.14.2
matchedEpisodeType	srs	xsd:string	*NO*	Appendix B.14.3

B.14.1 *matchedRating*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *YES*

Description: The *matchedRating* property contains the actual rating of the recording. This is the rating of the external item that matched (was less or equal to) a rating limit specified in one of the *matchingRatingLimit* properties of the *recordSchedule*. The format is identical to the format of the *matchingRatingLimit* property. See Appendix B.6.4, "*matchingRatingLimit* for details.

Default Value: N/A – Output only.

Sort Order: Same as *matchingRatingLimit* property.

Input: N/A.

Output: The current setting.

B.14.2 *matchedRating@type*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchedRating@type* property indicates the rating system used in the *matchedRating* property. The format is identical to the format of the *matchingRatingLimit@type* property. See Appendix B.6.4.2, "*matchingRatingLimit@type*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *matchingRatingLimit@type* property.

Input: N/A.

Output: The current setting.

B.14.3 *matchedEpisodeType*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *matchedEpisodeType* property contains the actual episode type of the recording. This is the episode type of the external item that matched episode type specified in the *matchingEpisodeType* property of the *recordSchedule*. The format is identical to the format of the *matchingEpisodeType* property. See Appendix B.6.5, "*matchingEpisodeType*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *matchingEpisodeType* property.

Input: N/A.

Output: The current setting.

B.15 Task Matched Content Control Properties

Table B-37: Task Matched Content Control Properties

Property Name	NS	Data Type	M-Val	Reference
taskStartDateTimeAdjust	srs	xsd:string	*NO*	Appendix B.15.1
taskDurationAdjust	srs	xsd:string	*NO*	Appendix B.15.2
taskDurationLimit	srs	xsd:string	*NO*	Appendix B.15.3
taskDurationLimit@effect	srs	xsd:string	*NO*	Appendix B.15.4
taskChannelMigration	srs	xsd:boolean	*NO*	Appendix B.15.5
taskTimeMigration	srs	xsd:boolean	*NO*	Appendix B.15.6

B.15.1 *taskStartDateTimeAdjust*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskStartDateTimeAdjust* property is set to the value of the *scheduledStartDateTimeAdjust* property of the parent *recordSchedule*. The format is identical to the format of the *scheduledStartDateTimeAdjust* property. See Appendix B.7.2, "*scheduledStartDateTimeAdjust*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *scheduledStartDateTimeAdjust* property.

Input: N/A.

Output: The current setting.

B.15.2 *taskDurationAdjust*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskDurationAdjust* property is set to the value of the *scheduledDurationAdjust* property of the parent *recordSchedule*. The format is identical to the format of the *scheduledDurationAdjust* property. See Appendix B.7.3, "*scheduledDurationAdjust*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *scheduledDurationAdjust* property.

Input: N/A.

Output: The current setting.

B.15.3 *taskDurationLimit*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

The *taskDurationLimit* property is set to the value of the *durationLimit* property of the parent *recordSchedule*. The format is identical to the format of the *durationLimit* property. See Appendix B.7.5, "*durationLimit*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *durationLimit* property.

Input: N/A.

Output: The current setting.

B.15.4 *taskDurationLimit@effect*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

The *taskDurationLimit@effect* property is set to the value of the *durationLimit@effect* property of the parent *recordSchedule*. The format is identical to the format of the *durationLimit@effect* property. See Appendix B.7.5.1, "*durationLimit@effect*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *durationLimit@effect* property.

Input: N/A.

Output: The current setting.

B.15.5 *taskChannelMigration*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskChannelMigration* property is set to the value of the *channelMigration* property of the parent *recordSchedule*. The format is identical to the format of the *channelMigration* property. See Appendix B.7.6, "*channelMigration*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *channelMigration* property.

Input: N/A.

Output: The current setting.

B.15.6 *taskTimeMigration*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskTimeMigration* property is set to the value of the *timeMigration* property of the parent *recordSchedule*. The format is identical to the format of the *timeMigration* property. See Appendix B.7.7, "*timeMigration*" for details.

Default Value: N/A – Output only.

Sort Order: Same as *timeMigration* property.

Input: N/A.

Output: The current setting.

B.16 Task State Properties

Table B-38: State Related Properties

Property Name	NS	Data Type	M-Val	Reference
taskState	srs	xsd:string	*NO*	Appendix B.16.1
taskState@phase	srs	xsd:string	*NO*	Appendix B.16.1.2
taskState@startDateTimeMet	srs	xsd:boolean	*NO*	Appendix B.16.1.3
taskState@endDateTimeMet	srs	xsd:boolean	*NO*	Appendix B.16.1.4
taskState@recording	srs	xsd:boolean	*NO*	Appendix B.16.1.5
taskState@someBitsRecorded	srs	xsd:boolean	*NO*	Appendix B.16.1.6
taskState@someBitsMissing	srs	xsd:boolean	*NO*	Appendix B.16.1.7
taskState@firstBitsRecorded	srs	xsd:boolean	*NO*	Appendix B.16.1.8
taskState@lastBitsRecorded	srs	xsd:boolean	*NO*	Appendix B.16.1.9
taskState@fatalError	srs	xsd:boolean	*NO*	Appendix B.16.1.10
taskState@currentErrors	srs	CSV (xsd:int)	*NO*	Appendix B.16.1.11
taskState@errorHistory	srs	CSV (xsd:int)	*NO*	Appendix B.16.1.12
taskState@pendingErrors	srs	CSV (xsd:int)	*NO*	Appendix B.16.1.13
taskState@infoList	srs	CSV (xsd:int)	*NO*	Appendix B.16.1.14

B.16.1 *taskState*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskState* property indicates the overall state of the *recordTask*.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on the order in Table B-39. Ascending: first table entry first.

Input: N/A.

Output: The current setting.

B.16.1.1 allowedValueList for the *taskState* Property

This section defines the normative allowed values for the *taskState* property. Each of these values represents a semantically meaningful combination of values for some of the "low-level" state properties (that is: *taskState@xxx*). Although it is possible to derive the value of the *taskState* property from some of the "low-level" *taskState@xxx* properties, the *taskState* property provides a more convenient mechanism to determine the current state of the *recordTask*.

The definition of each state is shown in the table below. This table represents the normative definitions of the various states. Although some of the low-level state properties have been declared as optional, their underlying semantics and their significance towards the definition of each valid taskState value is not diminished. Regardless of whether or not a given device is able to expose all of the low-level state properties, every device has a conceptual notion of property semantics. For example, some devices may not be able to support the *taskState@lastBitsRecorded* property, however, even these devices have an internal concept that the last bits of the content have or have not been recorded.

In some cases, a specific low-level state property does not contribute to the definition of a given state. In other words, the low-level property can have any value without affecting the semantics of the state. This situation is indicated by a "-" in the table entry.

The "Ø" symbol is used to indicate an empty attribute. The "{}" symbol is used when the attribute is not empty.

Following this table, a more intuitive informational description of each state value and their support level is described.

Table B-39: allowedValueList for the *taskState* Property

Value	@phase	@recording	@someBitsRecorded	@someBitsMissing	@fatalError	@currentErrors	@pendingErrors	@errorHistory	@startDateTimeMet *	@endDateTimeMet *	@firstBitsRecorded *	@lastBitsRecorded *
"IDLE.READY"	"IDLE"	0	0	0	0	Ø	Ø	Ø	0	0	0	0
"IDLE.ATRISK"	"IDLE"	0	0	0	0	Ø	{}	Ø	0	0	0	0
"ACTIVE.TRANSITION.FROMSTART"	"ACTIVE"	0	0	0	0	Ø	-	Ø	1	0	0	0
"ACTIVE.TRANSITION.RESTART"	"ACTIVE"	0	-	1	0	Ø	-	{}	1	0	-	0
"ACTIVE.RECORDING.FROMSTART.OK"	"ACTIVE"	1	1	0	0	Ø	Ø	Ø	1	0	1	0
"ACTIVE.RECORDING.FROMSTART.ATRISK"	"ACTIVE"	1	1	0	0	Ø	{}	Ø	1	0	1	0
"ACTIVE.RECORDING.RESTART.OK"	"ACTIVE"	1	1	1	0	Ø	Ø	{}	1	0	-	0
"ACTIVE.RECORDING.RESTART.ATRISK"	"ACTIVE"	1	1	1	0	Ø	{}	{}	1	0	-	0
"ACTIVE.NOTRECORDING"	"ACTIVE"	0	-	1	0	{}	-	{}	1	0	-	0
"DONE.FULL"	"DONE"	0	1	0	0	Ø	-	Ø	1	1	1	1
"DONE.PARTIAL"	"DONE" "DONE"	0 0	1 1	1 1	0 1	Ø Ø	- -	{} {}	1 -	1 0	- -	- -
"DONE.EMPTY"	"DONE" "DONE"	0 0	0 0	1 1	0 1	Ø Ø	- -	{} {}	- -	1 0	0 0	0 0

* Some implementations may not expose these individual properties to the control point. However, in this case, all visible external behavior of the device MUST be as if it implemented all of the properties as specified in the table above.

In the following table, a more intuitive informational description of each state value and its support level is described.

Table B-40: allowedValueList for the *taskState* Property

Value	R/O	Description
"IDLE.READY"	R	The *recordTask* is waiting for the start time to be reached. No errors have been detected.
"IDLE.ATRISK"	O	The *recordTask* is waiting for the start time to be reached while some pending errors exist.

Value	R/O	Description
"*ACTIVE.TRANSITION.FROMSTART*"	*O*	The device's record mechanism has been initiated to record the content from its beginning but no actual recording has occurred.
"*ACTIVE. TRANSITION.RESTART*"	*O*	The device's record mechanism has been re-initiated following some content loss from previous error conditions.
"*ACTIVE.RECORDING.FROMSTART.OK*"	*R*	The device's record mechanism is currently continuously recording from the beginning. No current or pending errors exist.
"*ACTIVE.RECORDING.FROMSTART.ATRISK*"	*O*	The device's record mechanism is currently continuously recording from the beginning. Some pending errors are detected.
"*ACTIVE.RECORDING.RESTART.OK*"	*O*	The device's record mechanism is currently recording content, following some content loss from previous error conditions. No current or pending errors exist.
"*ACTIVE.RECORDING.RESTART.ATRISK*"	*O*	The device's record mechanism is currently recording content following some content loss from previous error conditions. One or more pending errors are detected, which will block the recording in the future.
"*ACTIVE.NOTRECORDING*"	*O*	The device's record mechanism is currently NOT recording content due to one or more error conditions.
"*DONE.FULL*"	*R*	The *recordTask* has reached its final disposition and no other property or attribute changes will occur. All of the content has been recorded.
"*DONE.PARTIAL*"	*R*	The *recordTask* has reached its final disposition and no other property or attribute changes will occur. The content is only partially recorded due to error(s).
"*DONE.EMPTY*"	*R*	The *recordTask* has reached its final disposition and no other property or attribute changes will occur. No content has been recorded at all due to error conditions.

B.16.1.2 *taskState@phase*

Namespace: srs **Property Data Type**: xsd:string **Multi-Valued**: *NO*

Description: The *taskState@phase* property indicates the current phase of a *recordTask* within its normal lifetime. The following allowed values for this property are sequentially assigned at the appropriate points

in time within the *recordTask*'s normal lifetime: "*IDLE*" → "*ACTIVE*" → "*DONE*". In certain cases, some of the phase values may be skipped, for example, when a fatal error is detected.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on the order in Table B-41. Ascending: first table entry first.

Input: N/A.

Output: The current setting.

B.16.1.2.1 allowedValueList for the *taskState@phase* Property

Table B-41: allowedValueList for the *taskState@phase* Property

Value	R/O	Description
"*IDLE*"	*R*	Indicates that the *recordTask*'s start time has not yet been reached.
"*ACTIVE*"	*R*	Indicates that the *recordTask* is in between the "*IDLE*" and "*DONE*" phases. Typically, the *recordTask*'s content is (partially) available and an attempt is made to record the remaining content.
"*DONE*"	*R*	Indicates that the *recordTask*'s final disposition has been reached. For example, the *recordTask*'s end time has been reached or a fatal error has occurred. Once the device reaches this phase, no additional state changes occure.

B.16.1.3 *taskState@startDateTimeMet*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@startDateTimeMet* property indicates whether the *recordTask*'s *actualStartDateTime* has been reached. See Section 2.2, "Terms" for the definition of *actualStartDateTime*.

If a *recordTask* has reached the "*DONE*" phase, this property indicates the last status before the *recordTask* has reached the "*DONE*" phase. Note: if the *recordTask* terminates prematurely (that is: reaches the "*DONE*" phase before the start time is reached, for example, due to a fatal error), this property is not updated.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.4 *taskState@endDateTimeMet*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@endDateTimeMet* property indicates whether the *recordTask*'s *actualEndDateTime* has been reached. See Section 2.2, "Terms" for the definition of *actualEndDateTime*.

If a *recordTask* has reached the "*DONE*" phase, this property indicates the last status before the *recordTask* has reached the "*DONE*" phase. Note: if the *recordTask* terminates prematurely (that is:

reaches the "*DONE*" phase before the end time is reached, for example, due to a fatal error), this property is not updated.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.5 *taskState@recording*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@recording* property indicates whether one of the device's record destinations is currently recording the content identified by the *recordTask*.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.6 *taskState@someBitsRecorded*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@someBitsRecorded* property indicates whether some portion of the content identified by the *recordTask* has been recorded.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.7 *taskState@someBitsMissing*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@someBitsMissing* property indicates whether some portion of the content identified by the *recordTask* has not been recorded. This property will be "*0*" as long as all the bits that have been available so far have also been recorded.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.8 *taskState@firstBitsRecorded*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@firstBitsRecorded* property indicates whether the first portion of the content identified by the *recordTask* has been recorded.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.9 *taskState@lastBitsRecorded*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@lastBitsRecorded* property indicates whether the ending portion of the content identified by the *recordTask* has been recorded.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.10 *taskState@fatalError*

Namespace: srs **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

Description: The *taskState@fatalError* property indicates whether a fatal error has occurred. A fatal error is defined to be an error condition that causes the *recordTask* to terminate before its *actualEndDateTime* has been reached.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.11 *taskState@currentErrors*

Namespace: srs **Property Data Type**: CSV (xsd:int) **Multi-Valued**: *NO*

Description: The *taskState@currentErrors* property identifies the CSV list of errors that are currently preventing the *recordTask* from recording. An empty CSV list indicates that there are no errors currently blocking the recording. Multiple errors are listed in order of occurrence starting with the oldest error and ending with the most recent.

When the errors are not resolved before reaching the "*DONE*" phase, they MAY be persisted in the "*DONE*" phase. If a device persists current errors, the value of this property MUST be set to the value that this property had immediately prior to entering the "*DONE*" phase.If a device does not persist current errors, the *taskState@currentErrors* MUST be empty in the "*DONE*" phase.

By definition, this property MUST be empty while in the "*IDLE*" phase. The current errors are also copied to the *taskState@errorHistory* property.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.16.1.11.1 allowedValueList for the *taskState@currentErrors* Property and Other Error Properties

The following table defines error codes for all error properties of a *recordTask*, such as *taskState@currentErrors*, *taskState@errorHistory*, etc. to expose error conditions. This error list can be extended in the future or by vendors. The errors are grouped into separate categories and labeled 1xx, 2xx, 3xx, and 4xx groups, each group representing the nature of errors; that is: general errors, media errors, system errors and content errors, respectively. The grouping of error codes allows a control point to be able to understand the nature of errors when an unknown error code (that is: extended specification or vendor extended) is encountered. For example, if an unknown error is labeled 3xx, it can be interpreted by the control point as 300.

Table B-42: allowedValueList for the *taskState@xxx* Properties

Value	R/O	Description
Non-positive	*N/A*	These error codes are reserved for future use. Control points should gracefully ignore any non-positive error codes.
001-099	*N/A*	Reserved.
100-199	*N/A*	General Error Code Group - arbitrary errors, which do not belong to other groups.
100	*R*	General Problem – a problem is confirmed, but no specific reason can be identified.
101	*O*	Disabled - the *recordTask* is disabled by the user.
102	*O*	The *recordTask*'s enable/disable behavior is overriding the default behavior specified by the associated *recordSchedule*.
103-149	*N/A*	Reserved for future General Error Codes.
150-199	*N/A*	Reserved for vendor-defined General Error Codes.
200-299	*N/A*	Media Error Code Group - arbitrary media related errors.
200	*O*	General Media Problem – some trouble related to media is detected. Replacing the media may likely resolve it.
201	*O*	No Media – necessary media is missing from the recording device.
202	*O*	Media Write Protect - write access to the recording media is prohibited.
203	*O*	Insufficient Media Space - recording media does not have enough available space to complete the *recordTask*.
204	*O*	Media Low Space - the recording media has low available space and the *recordTask* may fail. The criteria to determine "low space" is vendor dependent and may be independent from the size of the scheduled content to record.
205-249	*N/A*	Reserved for future Media ErrorCodes.
250-299	*N/A*	Reserved for vendor-defined Media Error Codes.
300-399	*N/A*	System Error Code Group - arbitrary system related error.
300	*O*	General System Problem – a problem related to the system is detected. It may affect all *recordTask* instances in the ScheduledRecording service.
301	*O*	Insufficient Memory- the system does not have enough system memory to complete the *recordTask*.

Value	R/O	Description
302	*O*	Insufficient Processing - the system does not have enough CPU power to execute the *recordTask*.
303	*O*	Low Memory - the system has low available memory and the *recordTask* may fail. The criteria to determine "low memory" is vendor dependent and may be independent from the size of the scheduled content to record.
304	*O*	Low Processing - the system has low available CPU power and the *recordTask* may fail. The criteria to determine "low processing" is vendor dependent and may be independent from the size of the scheduled content to record.
305	*O*	Signal Lost - the system has lost the input signal.
306	*O*	Low Signal - The system has low input signal and the *recordTask* may fail. The criteria to determine "low processing" is vendor dependent.
307	*O*	No EPG - the system lost access to the EPG.
308-349	*N/A*	Reserved for future System Error Codes.
350-399	*N/A*	Reserved for vendor-defined System Error Codes.
400-499	*N/A*	Content Error Code Group - arbitrary errors related to the content program to be recorded.
400	*O*	General Content Problem – a problem related to the content is detected. It may be associated with the content that is being recorded.
401	*O*	Conflicting Program Loser – there are other conflicting programs with overlapping time period, and the current *recordTask* is superseded by the conflicting program.
402	*O*	Conflicting Program Winner - there are other conflicting programs with overlapping time period, and the current *recordTask* superseded the conflicting program.
403	*O*	PPV (Pay per View) - the content is PPV and some procedures are needed for the *recordTask* to begin.
404	*O*	Content Rescheduled - the originally scheduled content has been preempted.
405-449	*N/A*	Reserved for future Content Error Codes.
450-499	*N/A*	Reserved for vendor-defined Content Error Codes.
500 and above	*N/A*	Reserved for future new category information extensions.

B.16.1.12 *taskState@errorHistory*

Namespace: srs **Property Data Type**: CSV (xsd:int) **Multi-Valued**: *NO*

Description: The *taskState@errorHistory* property identifies the CSV list of errors that have (at any time) prevented the *recordTask* from completing successfully. This includes both past and current recording errors. Multiple errors are listed in order of occurrence starting with the oldest error and ending with the most recent. An empty list indicates that none of the *recordTask*'s content has yet been prevented from being recorded. By definition, this list will always be empty while in the "*IDLE*" phase. Note: Any errors listed in *taskState@currentErrors* MUST also be copied to and persisted in this property.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.16.1.12.1 allowedValueList for the *taskState@errorHistory* Property

See Appendix B.16.1.11.1, "allowedValueList for the *taskState@currentErrors* Property" for details.

B.16.1.13 *taskState@pendingErrors*

Namespace: srs **Property Data Type**: CSV (xsd:int) **Multi-Valued**: *NO*

Description: The *taskState@pendingErrors* property identifies the CSV list of errors that may prevent the *recordTask* from completing successfully at some time in the future unless resolved. An empty CSV list means that no pending errors have been detected. The list of errors that the device is able to detect before they actually occur may be obtained via the *GetAllowedValues()* action.

Those devices that are not able to detect any pending errors before they actually occur MAY always return an empty list for the value of this property. In this case, the value returned by *GetAllowedValues()* for this property MUST also be an empty list.

If any of these pending errors actually occur, they MUST be added to the *taskState@currentErrors* list and *taskState@ErrorHistory* and removed from this list. When the pending errors did not occur, these errors MAY be persisted to the "*DONE*" phase. If a device does not persist any pending errors that have not occurred yet, then the *taskState@pendingErrors* MUST be empty in the "*DONE*" phase. Otherwise the value of this property MUST be set to the value that this property had immediately prior to entering the "*DONE*" phase.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.16.1.13.1 allowedValueList for the *taskState@pendingErrors* Property

See Appendix B.16.1.11.1, "allowedValueList for the *taskState@currentErrors* Property" for details.

B.16.1.14 *taskState@infoList*

Namespace: srs **Property Data Type**: CSV (xsd:int) **Multi-Valued**: *NO*

Description: The *taskState@infoList* property identifies the CSV list of additional conditions that have been detected but will not block the current *recordTask*, for example, conflict winner.

The list of possible information that the device is able to detect may be obtained via the *GetAllowedValues()* action.

Devices that are not able to detect any additional information MUST always return an empty list. In this case, the value returned by *GetAllowedValues()* for this property MUST also be an empty list.

Note: a device can also use *additionalStatusInfo* to expose information in text format.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.16.1.14.1 allowedValueList for the *taskState@infoList* Property

See Appendix B.16.1.11.1, "allowedValueList for the *taskState@currentErrors* Property" for details.

B.17 ContentDirectory Service Imported Properties

ContentDirectory service properties are imported through the *cdsReference* multi-valued property. The main reason to import properties (metadata) from ContentDirectory service objects into a *recordSchedule* or *recordTask* object is to make that object self-contained; that is: a control point can retrieve relevant metadata from the ScheduledRecording service object without having to first extract the object IDs of external ContentDirectory service objects and then retrieve the metadata from these objects via additional actions. In addition, even when the referenced object in the ContentDirectory service is deleted, its metadata is still preserved within the ScheduledRecording service. It is the responsibility of the device to maintain consistency between the actual ContentDirectory service object's metadata and the metadata contained in the corresponding *cdsReference* property.

The *cdsReference* property MUST contain a *valid* (it MUST contain all the REQUIRED properties as dictated by the DIDL-Lite Schema; also, if dependent properties are imported, their independent properties MUST be imported as well.) and properly escaped *DIDL-Lite XML Document* as defined in the ContentDirectory service specification. (Care must be taken to correctly define namespaces.)

The *DIDL-Lite XML Document* describes a device-dependent (sub)set of imported properties (metadata) of the ContentDirectory service object that is referenced by the linked *xxxCDSObjectID* property. The information contained in the *DIDL-Lite XML Document* MUST exactly match the *DIDL-Lite XML Document* that would be returned in the *Result* argument of the *ContentDirectory::Browse()* action with its input arguments set as follows:

ObjectID: The linked *xxxCDSObjectID* property value.

BrowseFlag: Set to "*BrowseMetaData*".

Filter: Set to the list of property names that are imported from the ContentDirectory service by the ScheduledRecording service.

StartingIndex: 0.

RequestedCount: 0.

SortCriteria: "", the empty string.

The following example illustrates the possible content of a *cdsReference* property in the context of a *recordSchedule* object (expressed in XML).

```
<?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">
   <item id="sched001">
      <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
      <title>My Schedule</title>
      …
      <scheduledCDSObjectID link="schedObj001">
         epg001
      </scheduledCDSObjectID>
      …
      <cdsReference link="schedObj001">

<!--
```

```
The following DIDL-Lite XML Document needs to be interpreted as a simple
string and therefore needs to be properly escaped
-->

        &lt;?xml version="1.0" encoding="UTF-8"?&gt;
        &lt;DIDL-Lite
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
         xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"&gt;
           &lt;item id="epg001" parentID="container007"
            restricted="0"&gt;
              &lt;dc:title&gt;Friends&lt;/dc:title&gt;
              &lt;upnp:class&gt;
                 object.item.epgItem.videoProgram
              &lt;/upnp:class&gt;
              …
           &lt;/item&gt;
        &lt;/DIDL-Lite&gt;

<!-- End of DIDL-Lite XML Document -->

      </cdsReference>
      …
   </item>
</srs>
```

The next example illustrates the possible content of two *cdsReference* property instances relating to the *taskCDSObjectID* and *recordedCDSObjectID* property in the context of a *recordTask* object (expressed in XML).

```
<?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">
   <item id="Task001">
      <class>OBJECT.RECORDTASK</class>
      <title>My Task</title>
      …
      <taskCDSObjectID link="tskObj001">
         epg001
      </taskCDSObjectID>
      …
      <recordedCDSObjectID link="recObj001">
         rec001
      </recordedCDSObjectID>
      …
      <cdsReference link="tskObj001">

<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple
string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;?DIDL-Lite
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
          xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"&gt;
            &lt;?item id="epg001" parentID="container007"
             restricted="0"&gt;
               &lt;?dc:title&gt;Friends&lt;?/dc:title&gt;
               &lt;?upnp:class&gt;
                  object.item.epgItem.videoProgram
               &lt;?/upnp:class&gt;
               …
            &lt;?/item&gt;
         &lt;?/DIDL-Lite&gt;

<!-- End of DIDL-Lite XML Document -->

      </cdsReference>
      …
      <cdsReference link="recObj001">

<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple
string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;?DIDL-Lite
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
```

```
          xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"&gt;
            &lt;?item id="rec001" parentID="cnt009" restricted="0"&gt;
               &lt;?dc:title&gt;My Show&lt;?/dc:title&gt;
               &lt;?upnp:class&gt;
                  object.item.videoItem
               &lt;?/upnp:class&gt;
               …
            &lt;?/item&gt;
         &lt;?/DIDL-Lite&gt;

<!-- End of DIDL-Lite XML Document -->

      </cdsReference>
      …
   </item>
</srs>
```

Appendix C. AV Working Committee Class Definitions (Normative)

C.1 Class Hierarchy

The ScheduledRecording service exposes a class hierarchy which is used to type all objects that can be retrieved from it. Each class is named using a string of the form described in Appendix D.3, "Class Name Syntax" below.

For a particular class, some properties are REQUIRED, others are OPTIONAL and some are PROHIBITED.

A class that is derived from another class MUST include all of the member properties of the parent class. The definition of a derived class MAY make some optional properties of the base class REQUIRED.

Each class definition includes a list of properties. Each property is expressed in XML as either an XML Element or an XML Attribute. Some independent properties are multi-valued for a class, meaning that, in an XML instance of the class, the property may occur more than once.

This Appendix defines the base class *object* from which all other classes are derived. No object of this abstract class can be instantiated. From the object class, two classes are derived; the *object.recordSchedule* class and the *object.recordTask* class.

The abstract *object.recordSchedule* class and its two derived abstract classes *object.recordSchedule.direct* and *object.recordSchedule.query* make up the basic hierarchy from which all other *recordSchedule* classes are derived. These three classes can not be instantiated (no object can exist within the ScheduledRecording service that has its *class* property set to "*OBJECT.RECORDSCHEDULE*", "*OBJECT.RECORDSCHEDULE.DIRECT*" or "*OBJECT.RECORDSCHEDULE.QUERY*").

The *object.recordTask* class is used to type all *recordTask* objects in the ScheduledRecording service. The *object.recordTask* class has no derived classes defined yet.

In addition to these classes, a number of classes are derived from the *object.recordSchedule.direct* and *object.recordSchedule.query* classes. Figure 7 below shows the hierarchy of these classes.

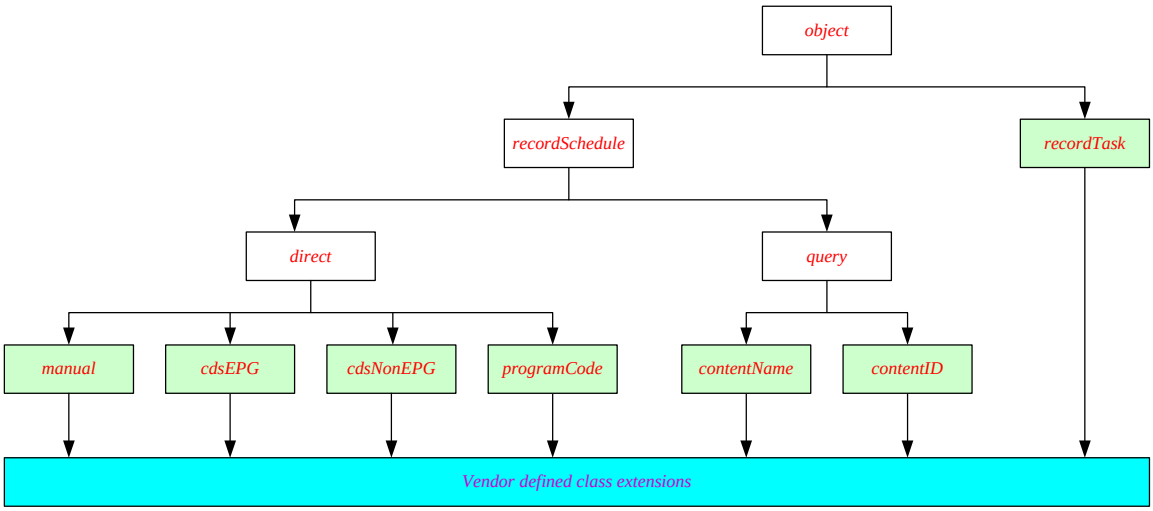

Figure 7: Class hierarchy for the ScheduledRecording service.

Vendors MAY extend the functionality, provided by the standard record classes, by adding vendor-defined properties. Any device that adds a property whose description matches one of the AV Working Committee-defined property descriptions MUST use the AV Working Committee-defined property name. In addition,

any device that uses a property name from the ScheduledRecording service specification MUST use it with the same semantics as the AV Working Committee-defined description of that property. In order to accommodate vendor-defined properties, control points should *gracefully ignore* any properties whose names and semantics they do not understand.

When adding properties, it is RECOMMENDED that vendors create a vendor-defined derived class with a vendor-defined class name, rather than adding the properties to the existing standard class without creating a vendor-defined class. This provides a simple mechanism for control points to determine if a class has been extended by simply examining the *class* property value. In all cases, vendor-defined classes MUST remain fully compatible with the standard class from which they were derived. In other words, control points that do not understand the specifics of the vendor-defined additions should still be able to interact with an instance of the vendor-defined derived class object as if it were an instance of that standard class.

Vendor-defined classes MUST always be derived from standard classes that can be instantiated (the green-colored boxes in Figure 7). It is therefore PROHIBITED to derive vendor-defined classes directly from classes, such as "*OBJECT.RECORDSCHEDULE*", "*OBJECT.RECORDSCHEDULE.DIRECT*", and "*OBJECT.RECORDSCHEDULE.QUERY*". It is allowed to derive vendor-defined classes from class "*OBJECT.RECORDTASK*".

All standard classes and vendor-defined derived classes supported by a particular ScheduledRecording service implementation MUST be individually listed in the allowedValueList of the *class* property. (This list can be retrieved via the *GetAllowedValues()* action.) Implementations are REQUIRED to support all intermediate classes in a chain of derived classes. For example, if an implementation supports a vendor-defined class "*OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG.HDTV.LOCAL*", then it MUST also support the "*OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG.HDTV*" and "*OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG*" classes.

As indicated in Appendix D.3, "Class Name Syntax", strict naming conventions MUST be followed when naming derived vendor-defined classes. Vendor-defined class names MUST be based on one of the instantiatable class names in this specification. Therefore, *all* vendor-defined class names MUST start with one of the string values, defined in Table B-2, "allowedValueList for the *class* Property".

Control points that need to examine *class* property values, should be prepared to handle vendor-defined class names. Therefore, control points should *never* use simple string matching to determine the actual class of an object. Rather, they should parse the class name from left to right and determine if there is at least a left substring match with one of the string values defined in Table B-2, "allowedValueList for the *class* Property". If such a match is found, the remaining characters in the class name can be examined for potential matches with vendor-defined class names of which the control point is aware. If no such match is found, the control point can treat the object as if it were an instance of the most specialized class for which a match was found.

C.1.1 Relationships between Classes and Properties

The following tables present a complete overview of all the defined properties and in which classes these properties are actually used (member properties).

For a particular class, some properties are REQUIRED, others are OPTIONAL and some are PROHIBITED. Every instance of a class MUST have a value for each supported REQUIRED or OPTIONAL member property of that class (see Section 2.2, "Terms").

The support level of a member property defines how the member property MUST be used in the arguments of an action when that action is invoked. The support level of a member property can be different for *recordSchedule*, *recordScheduleParts*, and *recordTask* usage.

The *recordScheduleParts* support level for the specified class indicates the use of a member property when a control point requests to create a *recordSchedule*. If a member property is defined as REQUIRED for *recordScheduleParts* usage, an argument of type *A_ARG_TYPE_RecordScheduleParts* MUST contain that member property and the ScheduledRecording service MUST support it. If it is defined as OPTIONAL, the ScheduledRecording service MAY support the member property and a control point may specify or omit

the member property in a request message even if the member property is supported by the ScheduledRecording service. PROHIBITED or unsupported OPTIONAL member properties specified in an argument of type *A_ARG_TYPE_RecordScheduleParts* MUST be gracefully ignored by the ScheduledRecording service. The set of properties that are supported for an argument of type *A_ARG_TYPE_RecordScheduleParts* can be retrieved by specifying "*A_ARG_TYPE_RecordScheduleParts*" in the *DataTypeID* argument when invoking the *GetPropertyList()* action. The support level for each of those supported member properties of each class can be retrieved by invoking the *GetAllowedValues()* action.

The *recordSchedule* support level for the specified class indicates the use of a member property when a control point retrieves a *recordSchedule* object. If a member property is defined as REQUIRED for *recordSchedule* usage, an argument of type *A_ARG_TYPE_RecordSchedule* MUST contain that member property and the ScheduledRecording service MUST support it. OPTIONAL supported member properties that are enumerated in the *Filter* argument MUST also be specified in the argument. If the resulting XML is not a valid document, other OPTIONAL properties MUST be added to create the smallest valid XML document. If the action does not have a *Filter* argument (like the *CreateRecordSchedule()* action), the action MUST return *all* OPTIONAL supported member properties (as if the *Filter* argument were set to "*:*"). If a control point does not specify a supported OPTIONAL member property in a request, the ScheduledRecording service MUST add it into the response and provide its default setting. The set of properties that are supported for an argument of type *A_ARG_TYPE_RecordSchedule* can be retrieved by specifying "*A_ARG_TYPE_RecordSchedule*" in the *DataTypeID* argument when invoking the *GetPropertyList()* action. The support level for each of those supported member properties of each class can be retrieved by invoking the *GetAllowedValues()* action.

The *recordTask* support level for the specified class indicates the use of a member property when a control point retrieves a *recordTask* object. If a member property is defined as REQUIRED for *recordTask* usage, an argument of type *A_ARG_TYPE_RecordTask* MUST contain that member property and the ScheduledRecording service MUST support it. OPTIONAL supported member properties that are enumerated in the *Filter* argument MUST also be specified in the argument. If the resulting XML is not a valid document, other OPTIONAL properties MUST be added to create the smallest valid XML document. The set of properties that are supported for an argument of type *A_ARG_TYPE_RecordTask* can be retrieved by specifying "*A_ARG_TYPE_RecordTask*" in the *DataTypeID* argument when invoking the *GetPropertyList()* action. The support level for each of those supported member properties of each class can be retrieved by invoking the *GetAllowedValues()* action.

Dependent properties are PROHIBITED if their associated independent property does not exist. They can be REQUIRED or OPTIONAL when the independent property does exist.

C.1.2 *recordScheduleParts* Properties

The following table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED or UNDEFINED) of a property when used in an argument of type *A_ARG_TYPE_RecordScheduleParts* for each class. The √ mark indicates that the property's support level is inherited from the parent class. The coloring still indicates the support level.

Table C-1: Class Properties Overview for *recordScheduleParts* usage

R REQUIRED O OPTIONAL P PPROHIBITED U UNDEFINED √ INHERITED **Property Name**	object	.recordSchedule	..direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
Common Properties											
Base Properties											
@id	R	√	√	√	√	√	√		√	√	√
title	R	√	√	√	√	√	√		√	√	√
class	R	√	√	√	√	√	√		√	√	√
additionalStatusInfo	U	P	√	√	√	√	√		√	√	√
cdsReference	U	P	√	√	√	√	√		√	√	√
cdsReference@link	U	P	√	√	√	√	√		√	√	√
Priority Properties											
priority	U	P	√	√	√	√	√		√	√	√
priority@orderedValue	U	P	√	√	√	√	√		√	√	√
desiredPriority	O	√	√	√	√	√	√		√	√	√
desiredPriority@type	R	√	√	√	√	√	√		√	√	√
Output Control Properties											
recordDestination	U	O	√	√	√	√	√		√	√	√
recordDestination@mediaType	U	O	√	√	√	√	√		√	√	√
recordDestination@targetURL	U	O	√	√	√	√	√		√	√	√
recordDestination@preference	U	O	√	√	√	√	√		√	√	√
desiredRecordQuality	O	√	√	√	√	√	√		√	√	√
desiredRecordQuality@type	R	√	√	√	√	√	√		√	√	√
Schedule Only Properties											
Content ID Related Properties											
scheduledCDSObjectID	U	√	√	P	R	R	P		P	√	√
scheduledCDSObjectID@link	U	P	√	√	√	√	√		√	√	√
scheduledChannelID	U	√	√	R	P	P	P		P	√	√
scheduledChannelID@type	U	√	√	R	P	P	P		P	√	√
scheduledStartDateTime	U	√	√	R	P	R	P		P	√	√
scheduledDuration	U	√	√	R	P	R	P		P	√	√
scheduledProgramCode	U	√	√	P	P	P	R		P	√	√
scheduledProgramCode@type	U	√	√	P	P	P	R		P	√	√
Matching Content Criteria											

R REQUIRED O OPTIONAL P PPROHIBITED U UNDEFINED √ INHERITED **Property Name**	object	.recordSchedule	..direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
matchingName	U	√	P	√	√	√	√		√	R	P
matchingName@type	U	√	P	√	√	√	√		√	R	P
matchingName@subStringMatch	U	√	P	√	√	√	√		√	O	P
matchingID	U	√	P	√	√	√	√		√	P	R
matchingID@type	U	√	P	√	√	√	√		√	P	R
Matching Qualifying Criteria											
matchingChannelID	U	√	P	√	√	√	√		√	O	P
matchingChannelID@type	U	√	P	√	√	√	√		√	R	P
matchingStartDateTimeRange	U	√	P	√	√	√	√		√	O	O
matchingDurationRange	U	√	P	√	√	√	√		√	O	O
matchingRatingLimit	U	√	P	√	√	√	√		√	O	O
matchingRatingLimit@type	U	√	P	√	√	√	√		√	R	R
matchingEpisodeType	U	√	P	√	√	√	√		√	O	O
Content Control Properties											
totalDesiredRecordTasks	U	O	√	√	√	√	√		√	√	√
scheduledStartDateTimeAdjust	U	O	√	√	√	√	√		√	√	√
scheduledDurationAdjust	U	O	√	√	√	√	√		√	√	√
activePeriod	U	√	√	O	O	O	P		O	√	√
durationLimit	U	√	√	P	O	P	P		O	√	√
durationLimit@effect	U	√	√	P	O	P	P		O	√	√
channelMigration	U	√	√	P	O	P	P		O	√	√
timeMigration	U	√	√	P	O	P	P		O	√	√
allowDuplicates	U	√	P	√	√	√	√		O	√	√
Storage Related Properties											
persistedRecordings	U	O	√	√	√	√	√		√	√	√
persistedRecordings@latest	U	O	√	√	√	√	√		√	√	√
persistedRecordings@preAllocation	U	O	√	√	√	√	√		√	√	√
persistedRecordings@storedLifetime	U	O	√	√	√	√	√		√	√	√
Schedule State Properties											
scheduleState	U	P	√	√	√	√	√		√	√	√
scheduleState@currentErrors	U	P	√	√	√	√	√		√	√	√
abnormalTasksExist	U	P	√	√	√	√	√		√	√	√

R REQUIRED O OPTIONAL P PPROHIBITED U UNDEFINED √ INHERITED **Property Name**	object	..recordSchedule	...direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		...query	...contentName	...contentID
Statistics Properties											
currentRecordTaskCount	U	P	√	√	√	√	√		√	√	√
totalCreatedRecordTasks	U	P	√	√	√	√	√		√	√	√
totalCompletedRecordTasks	U	P	√	√	√	√	√		√	√	√
Task Only Properties											
General Properties											
recordScheduleID	U	P	√	√	√	√	√		√	√	√
recordedCDSObjectID	U	P	√	√	√	√	√		√	√	√
recordedCDSObjectID@link	U	P	√	√	√	√	√		√	√	√
Content ID Related Properties											
taskCDSObjectID	U	P	√	√	√	√	√		√	√	√
taskCDSObjectID@link	U	P	√	√	√	√	√		√	√	√
taskChannelID	U	P	√	√	√	√	√		√	√	√
taskChannelID@type	U	P	√	√	√	√	√		√	√	√
taskStartDateTime	U	P	√	√	√	√	√		√	√	√
taskDuration	U	P	√	√	√	√	√		√	√	√
taskProgramCode	U	P	√	√	√	√	√		√	√	√
taskProgramCode@type	U	P	√	√	√	√	√		√	√	√
recordQuality	U	P	√	√	√	√	√		√	√	√
recordQuality@type	U	P	√	√	√	√	√		√	√	√
Matched Content Criteria											
matchedName	U	P	√	√	√	√	√		√	√	√
matchedName@type	U	P	√	√	√	√	√		√	√	√
matchedID	U	P	√	√	√	√	√		√	√	√
matchedID@type	U	P	√	√	√	√	√		√	√	√
Matched Qualifying Criteria											
matchedRating	U	P	√	√	√	√	√		√	√	√
matchedRating@type	U	P	√	√	√	√	√		√	√	√
matchedEpisodeType	U	P	√	√	√	√	√		√	√	√
Content Control Properties											
taskStartDateTimeAdjust	U	P	√	√	√	√	√		√	√	√
taskDurationAdjust	U	P	√	√	√	√	√		√	√	√

R REQUIRED O OPTIONAL P PPROHIBITED U UNDEFINED √ INHERITED Property Name	object	.recordSchedule	..direct	..manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
taskDurationLimit	U	P	√	√	√	√	√		√	√	√
taskDurationLimit@effect	U	P	√	√	√	√	√		√	√	√
taskChannelMigration	U	P	√	√	√	√	√		√	√	√
taskTimeMigration	U	P	√	√	√	√	√		√	√	√
Task State Properties											
taskState	U	P	√	√	√	√	√		√	√	√
taskState@phase	U	P	√	√	√	√	√		√	√	√
taskState@startDateTimeMet	U	P	√	√	√	√	√		√	√	√
taskState@endDateTimeMet	U	P	√	√	√	√	√		√	√	√
taskState@recording	U	P	√	√	√	√	√		√	√	√
taskState@someBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@someBitsMissing	U	P	√	√	√	√	√		√	√	√
taskState@firstBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@lastBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@fatalError	U	P	√	√	√	√	√		√	√	√
taskState@currentErrors	U	P	√	√	√	√	√		√	√	√
taskState@errorHistory	U	P	√	√	√	√	√		√	√	√
taskState@pendingErrors	U	P	√	√	√	√	√		√	√	√
taskState@infoList	U	P	√	√	√	√	√		√	√	√

C.1.3 *recordSchedule* Properties

The next table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED or UNDEFINED) of a property when used in an argument of type *A_ARG_TYPE_RecordSchedule* for each class. The √ mark indicates that the property's support level is inherited from the parent class. The coloring still indicates the support level.

Table C-2: Class Properties Overview for *recordSchedule* usage

R REQUIRED O OPTIONAL PROHIBITED U Undefined √ Inherited **Property Name**	object	..recordSchedule	..direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
Common Properties											
Base Properties											
@id	R	√	√	√	√	√	√		√	√	√
title	R	√	√	√	√	√	√		√	√	√
class	R	√	√	√	√	√	√		√	√	√
additionalStatusInfo	U	√	√	√	√	√	√		√	√	√
cdsReference	U	√	√	P	O	O	P		P	√	√
cdsReference@link	U	√	√	P	R	R	P		P	√	√
Priority Properties											
priority	U	R	√	√	√	√	√		√	√	√
priority@orderedValue	U	O	√	√	√	√	√		√	√	√
desiredPriority	O	√	√	√	√	√	√		√	√	√
desiredPriority@type	R	√	√	√	√	√	√		√	√	√
Output Control Properties											
recordDestination	U	R	√	√	√	√	√		√	√	√
recordDestination@mediaType	U	R	√	√	√	√	√		√	√	√
recordDestination@targetURL	U	O	√	√	√	√	√		√	√	√
recordDestination@preference	U	R	√	√	√	√	√		√	√	√
desiredrecordQuality	O	√	√	√	√	√	√		√	√	√
desiredrecordQuality@type	R	√	√	√	√	√	√		√	√	√
Schedule Only Properties											
Content ID Related Properties											
scheduledCDSObjectID	U	√	√	P	R	R	P		P	√	√
scheduledCDSObjectID@link	U	√	√	P	O	O	P		P	√	√
scheduledChannelID	U	√	√	R	P	P	P		P	√	√
scheduledChannelID@type	U	√	√	R	P	P	P		P	√	√
scheduledStartDateTime	U	√	√	R	P	R	P		P	√	√
scheduledDuration	U	√	√	R	P	R	P		P	√	√
scheduledProgramCode	U	√	√	P	P	P	R		P	√	√
scheduledProgramCode@type	U	√	√	P	P	P	R		P	√	√
Matching Content Criteria											

R REQUIRED O OPTIONAL PROHIBITED U Undefined √ Inherited **Property Name**	object	..recordSchedule	...direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		...query	...contentName	...contentID
matchingName	U	√	P	√	√	√	√		√	R	P
matchingName@type	U	√	P	√	√	√	√		√	R	P
matchingName@subStringMatch	U	√	P	√	√	√	√		√	O	P
matchingID	U	√	P	√	√	√	√		√	P	R
matchingID@type	U	√	P	√	√	√	√		√	P	R
Matching Qualifying Criteria											
matchingChannelID	U	√	P	√	√	√	√		√	O	P
matchingChannelID@type	U	√	P	√	√	√	√		√	R	P
matchingStartDateTimeRange	U	√	P	√	√	√	√		√	O	O
matchingDurationRange	U	√	P	√	√	√	√		√	O	O
matchingRatingLimit	U	√	P	√	√	√	√		√	O	O
matchingRatingLimit@type	U	√	P	√	√	√	√		√	R	R
matchingEpisodeType	U	√	P	√	√	√	√		√	O	O
Content Control Properties											
totalDesiredRecordTasks	U	O	√	√	√	√	√		√	√	√
scheduledStartDateTimeAdjust	U	O	√	√	√	√	√		√	√	√
scheduledDurationAdjust	U	O	√	√	√	√	√		√	√	√
activePeriod	U	√	√	O	O	O	P		O	√	√
durationLimit	U	√	√	P	O	P	P		O	√	√
durationLimit@effect	U	√	√	P	R	P	P		R	√	√
channelMigration	U	√	√	P	O	P	P		O	√	√
timeMigration	U	√	√	P	O	P	P		O	√	√
allowDuplicates	U	√	P	√	√	√	√		O	√	√
Storage Related Properties											
persistedRecordings	U	O	√	√	√	√	√		√	√	√
persistedRecordings@latest	U	O	√	√	√	√	√		√	√	√
persistedRecordings@preAllocation	U	O	√	√	√	√	√		√	√	√
persistedRecordings@storedLifetime	U	O	√	√	√	√	√		√	√	√
Schedule State Properties											
scheduleState	U	R	√	√	√	√	√		√	√	√
scheduleState@currentErrors	U	R	√	√	√	√	√		√	√	√
abnormalTasksExist	U	R	√	√	√	√	√		√	√	√

R REQUIRED O OPTIONAL PROHIBITED U Undefined √ Inherited **Property Name**	object	.recordSchedule	..direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
Statistics Properties											
currentRecordTaskCount	U	R	√	√	√	√	√		√	√	√
totalCreatedRecordTasks	U	O	√	√	√	√	√		√	√	√
totalCompletedRecordTasks	U	O	√	√	√	√	√		√	√	√
Task Only Properties											
General Properties											
recordScheduleID	U	P	√	√	√	√	√		√	√	√
recordedCDSObjectID	U	P	√	√	√	√	√		√	√	√
recordedCDSObjectID@link	U	P	√	√	√	√	√		√	√	√
Content ID Related Properties											
taskCDSObjectID	U	P	√	√	√	√	√		√	√	√
taskCDSObjectID@link	U	P	√	√	√	√	√		√	√	√
taskChannelID	U	P	√	√	√	√	√		√	√	√
taskChannelID@type	U	P	√	√	√	√	√		√	√	√
taskStartDateTime	U	P	√	√	√	√	√		√	√	√
taskDuration	U	P	√	√	√	√	√		√	√	√
taskProgramCode	U	P	√	√	√	√	√		√	√	√
taskProgramCode@type	U	P	√	√	√	√	√		√	√	√
recordQuality	U	P	√	√	√	√	√		√	√	√
recordQuality@type	U	P	√	√	√	√	√		√	√	√
Matched Content Criteria											
matchedName	U	P	√	√	√	√	√		√	√	√
matchedName@type	U	P	√	√	√	√	√		√	√	√
matchedID	U	P	√	√	√	√	√		√	√	√
matchedID@type	U	P	√	√	√	√	√		√	√	√
Matched Qualifying Criteria											
matchedRating	U	P	√	√	√	√	√		√	√	√
matchedRatingt@type	U	P	√	√	√	√	√		√	√	√
matchedEpisodeType	U	P	√	√	√	√	√		√	√	√
Content Control Properties											
taskStartDateTimeAdjust	U	P	√	√	√	√	√		√	√	√
taskDurationAdjust	U	P	√	√	√	√	√		√	√	√

R REQUIRED O OPTIONAL PROHIBITED U Undefined √ Inherited Property Name	object	.recordSchedule	..direct	...manual	...cdsEPG	...cdsNonEPG	...programCode		..query	...contentName	...contentID
taskDurationLimit	U	P	√	√	√	√	√		√	√	√
taskDurationLimit@effect	U	P	√	√	√	√	√		√	√	√
taskChannelMigration	U	P	√	√	√	√	√		√	√	√
taskTimeMigration	U	P	√	√	√	√	√		√	√	√
Task State Properties											
taskState	U	P	√	√	√	√	√		√	√	√
taskState@phase	U	P	√	√	√	√	√		√	√	√
taskState@startDateTimeMet	U	P	√	√	√	√	√		√	√	√
taskState@endDateTimeMet	U	P	√	√	√	√	√		√	√	√
taskState@recording	U	P	√	√	√	√	√		√	√	√
taskState@someBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@someBitsMissing	U	P	√	√	√	√	√		√	√	√
taskState@firstBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@lastBitsRecorded	U	P	√	√	√	√	√		√	√	√
taskState@fatalError	U	P	√	√	√	√	√		√	√	√
taskState@currentErrors	U	P	√	√	√	√	√		√	√	√
taskState@errorHistory	U	P	√	√	√	√	√		√	√	√
taskState@pendingErrors	U	P	√	√	√	√	√		√	√	√
taskState@infoList	U	P	√	√	√	√	√		√	√	√

C.1.4 *recordTask* Properties

The next table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED or UNDEFINED) of a property when used in an argument of type *A_ARG_TYPE_RecordTask*. The √ mark indicates that the property's support level is inherited from the parent class. The coloring still indicates the support level.

Table C-3: Class Properties Overview for *recordTask* usage

R REQUIRED O OPTIONAL PROHIBITED U Undefined √ Inherited Property Name	object	.recordTask
Common Properties		

R	REQUIRED
O	OPTIONAL
	PROHIBITED
U	Undefined
√	Inherited

Property Name	object	.recordTask
Base Properties		
@id	R	√
title	R	√
class	R	√
additionalStatusInfo	U	O
cdsReference	U	O
cdsReference@link	U	R
Priority Properties		
priority	U	R
priority@orderedValue	U	O
desiredPriority	O	√
desiredPriority@type	R	√
Output Control Properties		
recordDestination	U	R
recordDestination@mediaType	U	R
recordDestination@targetURL	U	O
recordDestination@preference	U	R
desiredrecordQuality	O	√
desiredrecordQuality@type	R	√
Schedule Only Properties		
Content ID Related Properties		
scheduledCDSObjectID	U	P
scheduledCDSObjectID@link	U	P
scheduledChannelID	U	P
scheduledChannelID@type	U	P
scheduledStartDateTime	U	P
scheduledDuration	U	P
scheduledProgramCode	U	P
scheduledProgramCode@type	U	P
Matching Content Criteria		
matchingName	U	P
matchingName@type	U	P
matchingName@subStringMatch	U	P

R	REQUIRED		
O	OPTIONAL		
	PROHIBITED		
U	Undefined		
√	Inherited		
Property Name		*object*	*.recordTask*
matchingID		U	P
matchingID@type		U	P
Matching Qualifying Criteria			
matchingChannelID		U	P
matchingChannelID@type		U	P
matchingStartDateTimeRange		U	P
matchingDurationRange		U	P
matchingRatingLimit		U	P
matchingRatingLimit@type		U	P
matchingEpisodeType		U	P
Content Control Properties			
totalDesiredRecordTasks		U	P
scheduledStartDateTimeAdjust		U	P
scheduledDurationAdjust		U	P
activePeriod		U	P
durationLimit		U	P
durationLimit@effect		U	P
channelMigration		U	P
timeMigration		U	P
allowDuplicates		U	P
Storage Related Properties			
persistedRecordings		U	P
persistedRecordings@latest		U	P
persistedRecordings@preAllocation		U	P
persistedRecordings@storedLifetime		U	P
Schedule State Properties			
scheduleState		U	P
scheduleState@currentErrors		U	P
abnormalTasksExist		U	P
Statistics Properties			
currentRecordTaskCount		U	P
totalCreatedRecordTasks		U	P
totalCompletedRecordTasks		U	P

R	REQUIRED		
O	OPTIONAL		
	PROHIBITED		
U	Undefined		
√	Inherited		
Property Name		*object*	*.recordTask*
Task Only Properties			
General Properties			
recordScheduleID		U	R
recordedCDSObjectID		U	O
recordedCDSObjectID@link		U	O
Content ID Related Properties			
taskCDSObjectID		U	O
taskCDSObjectID@link		U	O
taskChannelID		U	R
taskChannelID@type		U	R
taskStartDateTime		U	R
taskDuration		U	R
taskProgramCode		U	O
taskProgramCode@type		U	R
recordQuality		U	R
recordQuality@type		U	R
Matched Content Criteria			
matchedName		U	O
matchedName@type		U	R
matchedID		U	O
matchedID@type		U	R
Matched Qualifying Criteria			
matchedRating		U	O
matchedRatingt@type		U	R
matchedEpisodeType		U	O
Content Control Properties			
taskStartDateTimeAdjust		U	O
taskDurationAdjust		U	O
taskDurationLimit		U	O
taskDurationLimit@effect		U	R
taskChannelMigration		U	O
taskTimeMigration		U	O
Task State Properties			

R	REQUIRED
O	OPTIONAL
	PROHIBITED
U	Undefined
√	Inherited

Property Name	object	recordTask
taskState	U	R
taskState@phase	U	R
taskState@startDateTimeMet	U	O
taskState@endDateTimeMet	U	O
taskState@recording	U	R
taskState@someBitsRecorded	U	R
taskState@someBitsMissing	U	R
taskState@firstBitsRecorded	U	O
taskState@lastBitsRecorded	U	O
taskState@fatalError	U	R
taskState@currentErrors	U	R
taskState@errorHistory	U	R
taskState@pendingErrors	U	R
taskState@infoList	U	R

C.2 Class Definitions

The following sections define the standard record classes. The support level of the available properties for each class is also indicated. Vendors MAY add vendor-dependent properties to any of the defined classes. An instance of a normative class MUST NOT add properties, other than the properties already listed for each class definition below. In other words, a given instance of a record class can only have:

- The properties listed for that class (as per each definiton below).
- The properties that are defined members of the parent class.
- Vendor-defined properties that are using other XML namespace(s).
- ContentDirectory service properties imported with a normative namespace prefix (see Appendix B.17, “ContentDirectory Service Imported Properties”).

C.3 *object* Base Class

This is the abstract base class for the entire ScheduledRecording service class hierarchy. No object of this abstract class can be instantiated. The *object* class defines properties that are common to all ScheduledRecording service objects. The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for each property.

Table C-4: *object* Base Class Properties

Property Name	
Base Properties	
@id	R
title	R
class	R
additionalStatusInfo	U
cdsReference	U
cdsReference@link	U
Priority Properties	
priority	U
priority@orderedValue	U
desiredPriority	O
desiredPriority@type	R
Output Control Properties	
recordDestination	U
recordDestination@mediaType	U
recordDestination@targetURL	U
recordDestination@preference	U
desiredRecordQuality	O
desiredRecordQuality@type	R
Content ID Related Properties	
scheduledCDSObjectID	U
scheduledCDSObjectID@link	U
scheduledChannelID	U
scheduledChannelID@type	U
scheduledStartDateTime	U
scheduledDuration	U
scheduledProgramCode	U
scheduledProgramCode@type	U
Matching Content Criteria	
matchingName	U
matchingName@type	U
matchingName@subStringMatch	U
matchingID	U
matchingID@type	U
Matching Qualifying Criteria	
matchingChannelID	U
matchingChannelID@type	U
matchingStartDateTimeRange	U
matchingDurationRange	U
matchingRatingLimit	U
matchingRatingLimit@type	U
matchingEpisodeType	U
Content Control Properties	
totalDesiredRecordTasks	U
scheduledStartDateTimeAdjust	U
scheduledDurationAdjust	U
activePeriod	U
durationLimit	U
durationLimit@effect	U
channelMigration	U
timeMigration	U
allowDuplicates	U
Storage Related Properties	
persistedRecordings	U
persistedRecordings@latest	U
persistedRecordings@preAllocation	U
persistedRecordings@storedLifetime	U
Schedule State Properties	
scheduleState	U
scheduleState@currentErrors	U
abnormalTasksExist	U
Statistics Properties	
currentRecordTaskCount	U
totalCreatedRecordTasks	U
totalCompletedRecordTasks	U
Task General Properties	
recordScheduleID	U
recordedCDSObjectID	U
recordedCDSObjectID@link	U
Task Content ID Properties	
taskCDSObjectID	U
taskCDSObjectID@link	U
taskChannelID	U
taskChannelID@type	U
taskStartDateTime	U
taskDuration	U
taskProgramCode	U
taskProgramCode@type	U
recordQuality	U
recordQuality@type	U
Matched Content Criteria	
matchedName	U
matchedName@type	U
matchedID	U
matchedID@type	U
Matched Qualifying Criteria	
matchedRating	U
matchedRating@type	U
matchedEpisodeType	U
Content Control Properties	
taskStartDateTimeAdjust	U
taskDurationAdjust	U
taskDurationLimit	U
taskDurationLimit@effect	U
taskChannelMigration	U
taskTimeMigration	U
Task State Properties	
taskState	U
taskState@phase	U
taskState@startDateTimeMet	U
taskState@endDateTimeMet	U
taskState@recording	U
taskState@someBitsRecorded	U
taskState@someBitsMissing	U
taskState@firstBitsRecorded	U
taskState@lastBitsRecorded	U
taskState@fatalError	U
taskState@currentErrors	U
taskState@errorHistory	U
taskState@pendingErrors	U
taskState@infoList	U

C.3.1 *object.recordSchedule* Class

This is the abstract base class for the ScheduledRecording service record schedules class hierarchy. No object of this abstract class can be instantiated. The *object.recordSchedule* class defines properties that are common to all *object.recordSchedule* list entries. The table below lists all *recordSchedule*-related standard defined properties (*recordTask*-only properties are omitted from the table – see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-5: *object.recordSchedule* Base Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	P	O
cdsReference	P	√
cdsReference@link	P	√
Priority Properties		
priority	P	R
priority@orderedValue	P	O
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	O	R
recordDestination@mediaType	O	R
recordDestination@targetURL	O	O
recordDestination@preference	O	R
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	√	√
scheduledCDSObjectID@link	P	√
scheduledChannelID	√	√
scheduledChannelID@type	√	√
scheduledStartDateTime	√	√
scheduledDuration	√	√
scheduledProgramCode	√	√
scheduledProgramCode@type	√	√
Matching Content Criteria		
matchingName	√	√
matchingName@type	√	√
matchingName@subStringMatch	√	√
matchingID	√	√
matchingID@type	√	√

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	√	√
matchingChannelID@type	√	√
matchingStartDateTimeRange	√	√
matchingDurationRange	√	√
matchingRatingLimit	√	√
matchingRatingLimit@type	√	√
matchingEpisodeType	√	√
Content Control Properties		
totalDesiredRecordTasks	O	O
scheduledStartDateTimeAdjust	O	O
scheduledDurationAdjust	O	O
activePeriod	√	√
durationLimit	√	√
durationLimit@effect	√	√
channelMigration	√	√
timeMigration	√	√
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	O	O
persistedRecordings@latest	O	O
persistedRecordings@preAllocation	O	O
persistedRecordings@storedLifetime	O	O
Schedule State Properties		
scheduleState	P	R
scheduleState@currentErrors	P	R
abnormalTasksExist	P	R
Statistics Properties		
currentRecordTaskCount	P	R
totalCreatedRecordTasks	P	O
totalCompletedRecordTasks	P	O

C.3.1.1 *object.recordSchedule.direct* Class

The *object.recordSchedule.direct* abstract class is derived from the *object.recordSchedule* class. No object of this abstract class can be instantiated.

The main characteristic of the *object.recordSchedule.direct* class is that all the information that is needed to create associated *recordTask* instances is contained within the properties of the *recordSchedule*. The properties contain sufficient information to allow the ScheduledRecording service to translate this information into a deterministic set of *recordTask* properties. For example, if a ScheduledRecording service implementation supports the *object.recordSchedule.direct.programCode* class, the ScheduledRecording service is able to interpret the *scheduledProgramCode* property and derive the appropriate *taskStartDate*, *taskStartTime*, *taskDuration*, and *taskChannelID* *recordTask* properties from it. The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-6: *object.recordSchedule.direct* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	√
cdsReference@link	√	√
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	√	√
scheduledCDSObjectID@link	√	√
scheduledChannelID	√	√
scheduledChannelID@type	√	√
scheduledStartDateTime	√	√
scheduledDuration	√	√
scheduledProgramCode	√	√
scheduledProgramCode@type	√	√
Matching Content Criteria		
matchingName	P	P
matchingName@type	P	P
matchingName@subStringMatch	P	P
matchingID	P	P
matchingID@type	P	P

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	P	P
matchingChannelID@type	P	P
matchingStartDateTimeRange	P	P
matchingDurationRange	P	P
matchingRatingLimit	P	P
matchingRatingLimit@type	P	P
matchingEpisodeType	P	P
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	√	√
durationLimit	√	√
durationLimit@effect	√	√
channelMigration	√	√
timeMigration	√	√
allowDuplicates	P	P
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.1.1 *object.recordSchedule.direct.manual* Class

The *object.recordSchedule.direct.manual* class is used to create *recordSchedule* instances for manual scheduling of recordings. The content to be recorded is uniquely identified by the *scheduledChannelID*, *scheduledStartDateTime*, and *scheduledDuration* properties.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-7: *object.recordSchedule.direct.manual* Class Properties

Property Name	RSP	RS
Base Properties		
@id	R	R
title	R	R
class	R	R
additionalStatusInfo	O	O
cdsReference	O	P
cdsReference@link	O	P
Priority Properties		
priority	O	O
priority@orderedValue	O	O
desiredPriority	O	O
desiredPriority@type	O	O
Output Control Properties		
recordDestination	O	O
recordDestination@mediaType	O	O
recordDestination@targetURL	O	O
recordDestination@preference	O	O
desiredRecordQuality	O	O
desiredRecordQuality@type	O	O
Content ID Related Properties		
scheduledCDSObjectID	P	P
scheduledCDSObjectID@link	O	P
scheduledChannelID	R	R
scheduledChannelID@type	R	R
scheduledStartDateTime	R	R
scheduledDuration	R	R
scheduledProgramCode	P	P
scheduledProgramCode@type	P	P
Matching Content Criteria		
matchingName	O	O
matchingName@type	O	O
matchingName@subStringMatch	O	O
matchingID	O	O
matchingID@type	O	O

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	O	O
matchingChannelID@type	O	O
matchingStartDateTimeRange	O	O
matchingDurationRange	O	O
matchingRatingLimit	O	O
matchingRatingLimit@type	O	O
matchingEpisodeType	O	O
Content Control Properties		
totalDesiredRecordTasks	O	O
scheduledStartDateTimeAdjust	O	O
scheduledDurationAdjust	O	O
activePeriod	O	O
durationLimit	P	P
durationLimit@effect	P	P
channelMigration	P	P
timeMigration	P	P
allowDuplicates	O	O
Storage Related Properties		
persistedRecordings	O	O
persistedRecordings@latest	O	O
persistedRecordings@preAllocation	O	O
persistedRecordings@storedLifetime	O	O
Schedule State Properties		
scheduleState	O	O
scheduleState@currentErrors	O	O
abnormalTasksExist	O	O
Statistics Properties		
currentRecordTaskCount	O	O
totalCreatedRecordTasks	O	O
totalCompletedRecordTasks	O	O

C.3.1.1.2 *object.recordSchedule.direct.cdsEPG* Class

The *object.recordSchedule.direct.cdsEPG* class is used to create *recordSchedule* instances for scheduling of recordings, based on local EPG information. The content to be recorded is uniquely identified by the *scheduledCDSObjectID* property that MUST reference an EPG item (*object.item.epgItem* class) in an

associated ContentDirectory service. Most EPG item types currently defined identify only a single recording event. In the future, new EPG item types may be defined that identify multiple recording events.

The REQUIRED association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device. See also Appendix E, "ScheduledRecording Service Relationship to ContentDirectory Service" and Appendix F, "ScheduledRecording Service Relationship to EPG" for further details.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-8: *object.recordSchedule.direct.cdsEPG* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	O
cdsReference@link	√	R
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	R	R
scheduledCDSObjectID	√	O
scheduledChannelID	P	P
scheduledChannelID@type	P	P
scheduledStartDateTime	P	P
scheduledDuration	P	P
scheduledProgramCode	P	P
scheduledProgramCode@type	P	P
Matching Content Criteria		
matchingName	√	√
matchingName@type	√	√
matchingName@subStringMatch	√	√
matchingID	√	√
matchingID@type	√	√

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	√	√
matchingChannelID@type	√	√
matchingStartDateTimeRange	√	√
matchingDurationRange	√	√
matchingRatingLimit	√	√
matchingRatingLimit@type	√	√
matchingEpisodeType	√	√
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	O	O
durationLimit	O	O
durationLimit@effect	O	R
channelMigration	O	O
timeMigration	O	O
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.1.3 *object.recordSchedule.direct.cdsNonEPG* Class

The *object.recordSchedule.direct.cdsNonEPG* class is used to create *recordSchedule* instances for scheduling of recordings, for which (only) channel information is available in a local ContentDirectory database. The content to be recorded is uniquely identified by the *scheduledStartDateTime*, and *scheduledDuration* properties, supplemented with the *scheduledCDSObjectID* property that MUST

reference a ContentDirectory service object whose class is not "*object.item.epgItem*" or derived from that class. Additionally, the referenced ContentDirectory service object MUST identify content that will be available for recording at the time the recording is scheduled to start.

Examples of applicable ContentDirectory service objects are:

- A User Channel object that contains specific channel information.
- An object that represents an analog A/V input connection to the device.
- An object that represents an IP network program feed.
- An object that represents an already existing file.
- Etc.

The REQUIRED association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device. See also Appendix E, "ScheduledRecording Service Relationship to ContentDirectory Service" and Appendix F, "ScheduledRecording Service Relationship to EPG" for further details.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-9: *object.recordSchedule.direct.cdsNonEPG* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	O
cdsReference@link	√	R
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	R	R
scheduledCDSObjectID@link	√	O
scheduledChannelID	P	P
scheduledChannelID@type	P	P
scheduledStartDateTime	R	R
scheduledDuration	R	R
scheduledProgramCode	P	P
scheduledProgramCode@type	P	P
Matching Content Criteria		
matchingName	√	√
matchingName@type	√	√
matchingName@subStringMatch	√	√
matchingID	√	√
matchingID@type	√	√

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	√	√
matchingChannelID@type	√	√
matchingStartDateTimeRange	√	√
matchingDurationRange	√	√
matchingRatingLimit	√	√
matchingRatingLimit@type	√	√
matchingEpisodeType	√	√
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	O	O
durationLimit	P	P
durationLimit@effect	P	P
channelMigration	P	P
timeMigration	P	P
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.1.4 *object.recordSchedule.direct.programCode* Class

The *object.recordSchedule.direct.programCode* class is used to create *recordSchedule* instances for scheduling of recordings, based on program code information. The content to be recorded is uniquely identified by the *scheduledprogramCode* property that contains a unique code that can be translated by the ScheduledRecording service into a precise start date, start time, duration and channel for the recording event(s). However, most program code types currently defined identify only a single recording event. In the future, new program code types may be defined that identify multiple recording events.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED,

OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-10: *object.recordSchedule.direct.programCode* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	P
cdsReference@link	√	P
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	P	P
scheduledCDSObjectID@link	√	P
scheduledChannelID	P	P
scheduledChannelID@type	P	P
scheduledStartDateTime	P	P
scheduledDuration	P	P
scheduledProgramCode	R	R
scheduledProgramCode@type	R	R
Matching Content Criteria		
matchingName	√	√
matchingName@type	√	√
matchingName@subStringMatch	√	√
matchingID	√	√
matchingID@type	√	√

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	√	√
matchingChannelID@type	√	√
matchingStartDateTimeRange	√	√
matchingDurationRange	√	√
matchingRatingLimit	√	√
matchingRatingLimit@type	√	√
matchingEpisodeType	√	√
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	P	P
durationLimit	P	P
durationLimit@effect	P	P
channelMigration	P	P
timeMigration	P	P
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.2 *object.recordSchedule.query* Class

The *object.recordSchedule.query* abstract class is derived from the *recordSchedule* base class. No object of this abstract class can be instantiated.

The main characteristic of the *object.recordSchedule.query* class is that the properties of the *recordSchedule* are used as *matching criteria* to select items from external sources (like EPG databases, side-band metadata streams in digital broadcasts, etc.). After appropriate searching and matching, the metadata from these external items is used to populate *recordTask* instances. This process ensures that the *recordTask* properties match the rules set forth in the *recordSchedule*'s properties (matching criteria).

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED,

OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-11: *object.recordSchedule.query* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	P
cdsReference@link	√	P
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	P	P
scheduledCDSObjectID@link	√	P
scheduledChannelID	P	P
scheduledChannelID@type	P	P
scheduledStartDateTime	P	P
scheduledDuration	P	P
scheduledProgramCode	P	P
scheduledProgramCode@type	P	P
Matching Content Criteria		
matchingName	√	√
matchingName@type	√	√
matchingName@subStringMatch	√	√
matchingID	√	√
matchingID@type	√	√

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	√	√
matchingChannelID@type	√	√
matchingStartDateTimeRange	√	√
matchingDurationRange	√	√
matchingRatingLimit	√	√
matchingRatingLimit@type	√	√
matchingEpisodeType	√	√
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	O	O
durationLimit	O	O
durationLimit@effect	O	R
channelMigration	O	O
timeMigration	O	O
allowDuplicates	O	O
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.2.1 *object.recordSchedule.query.contentName* Class

The *object.recordSchedule.query.contentName* class is used to create *recordSchedule* instances for scheduling of recordings, based on program or series name information. The content to be recorded is determined by matching the value, specified in the *matchingName* property to the names of content items made available to the ScheduledRecording service by REQUIRED external resources like access to EPG databases, access to Service Information side-band data in digital broadcasts, etc. The matching process can be further restricted by providing a combination of Matching Qualifying Criteria properties. Any external content item MUST match those additional criteria to be considered a potential candidate for recording.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED,

OPTIONAL, REQUIRED, and UNDEFINED) in this class for *recordScheduleParts* (RSP) and *recordSchedule* (RS) usage for each property.

Table C-12: *object.recordSchedule.query.contentName* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	√
cdsReference@link	√	√
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	√	√
scheduledCDSObjectID@link	√	√
scheduledChannelID	√	√
scheduledChannelID@type	√	√
scheduledStartDatetime	√	√
scheduledDuration	√	√
scheduledProgramCode	√	√
scheduledProgramCode@type	√	√
Matching Content Criteria		
matchingName	R	R
matchingName@type	R	R
matchingName@subStringMatch	O	O
matchingID	P	P
matchingID@type	P	P

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	O	O
matchingChannelID@type	R	R
matchingStartDateTimeRange	O	O
matchingDurationRange	O	O
matchingRatingLimit	O	O
matchingRatingLimit@type	R	R
matchingEpisodeType	O	O
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	√	√
durationLimit	√	√
durationLimit@effect	√	√
channelMigration	√	√
timeMigration	√	√
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.1.2.2 *object.recordSchedule.query.contentID* Class

The *object.recordSchedule.query.contentID* class is used to create *recordSchedule* instances for scheduling of recordings, based on program or series ID information. The content to be recorded is determined by matching the value, specified in the *matchingID* property to the IDs of content items made available to the ScheduledRecording service by REQUIRED external resources like access to EPG databases, access Service Information side-band data in digital broadcasts, etc. The matching process can be further restricted by providing a combination of Matching Qualifying Criteria properties. Any external content item MUST match those additional criteria to be considered a potential candidate for recording.

The table below lists all standard defined properties (see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for both *input* and *output* for each property.

Table C-13: *object.recordSchedule.query.contentID* Class Properties

Property Name	RSP	RS
Base Properties		
@id	√	√
title	√	√
class	√	√
additionalStatusInfo	√	√
cdsReference	√	√
cdsReference@link	√	√
Priority Properties		
priority	√	√
priority@orderedValue	√	√
desiredPriority	√	√
desiredPriority@type	√	√
Output Control Properties		
recordDestination	√	√
recordDestination@mediaType	√	√
recordDestination@targetURL	√	√
recordDestination@preference	√	√
desiredRecordQuality	√	√
desiredRecordQuality@type	√	√
Content ID Related Properties		
scheduledCDSObjectID	√	√
scheduledCDSObjectID@link	√	√
scheduledChannelID	√	√
scheduledChannelID@type	√	√
scheduledStartDateTime	√	√
scheduledDuration	√	√
scheduledProgramCode	√	√
scheduledProgramCode@type	√	√
Matching Content Criteria		
matchingName	P	P
matchingName@type	P	P
matchingName@subStringMatch	P	P
matchingID	R	R
matchingID@type	R	R

Property Name	RSP	RS
Matching Qualifying Criteria		
matchingChannelID	P	P
matchingChannelID@type	P	P
matchingStartDateTimeRange	O	O
matchingDurationRange	O	O
matchingRatingLimit	O	O
matchingRatingLimit@type	R	R
matchingEpisodeType	O	O
Content Control Properties		
totalDesiredRecordTasks	√	√
scheduledStartDateTimeAdjust	√	√
scheduledDurationAdjust	√	√
activePeriod	√	√
durationLimit	√	√
durationLimit@effect	√	√
channelMigration	√	√
timeMigration	√	√
allowDuplicates	√	√
Storage Related Properties		
persistedRecordings	√	√
persistedRecordings@latest	√	√
persistedRecordings@preAllocation	√	√
persistedRecordings@storedLifetime	√	√
Schedule State Properties		
scheduleState	√	√
scheduleState@currentErrors	√	√
abnormalTasksExist	√	√
Statistics Properties		
currentRecordTaskCount	√	√
totalCreatedRecordTasks	√	√
totalCompletedRecordTasks	√	√

C.3.2 *object.recordTask* Class

This is the base class for the ScheduledRecording service record task class hierarchy. Currently, this is the only class defined in this hierarchy. All *recordTask* objects in the ScheduledRecording service are members of this class. The *object.recordTask* class defines properties that are common to all *recordTask* list entries.

A *recordTask* object represents an actual recording occurrence. More sophisticated ScheduledRecording service implementations MAY implement OPTIONAL actions that allow a control point to manipulate individual *recordTask* instances. For example, the OPTIONAL *DisableRecordTask()* action can be used to selectively disable (that is: recording task suspended and any actual recording MUST NOT occur) one or more *recordTask* instances, spawned from the same *recordSchedule* if not all recordings are desired.

A *recordTask* SHOULD be created by the ScheduledRecording service as soon as all necessary information (like EPG data) becomes available. It SHOULD be maintained at least until the *recordTask* has finished. It is RECOMMENDED to maintain all completed *recordTask* instances for a reasonable time or until space is needed so that control points can retrieve *recordTask* state information after the recording has finished.

One or more *recordTask* instances can be created per *recordSchedule*. Some *recordSchedule* instances may not have a *recordTask* because they have not scheduled any recordings yet.

The list of the *recordTask* instances can be obtained using the *BrowseRecordTasks()* action. A *recordTask* can be disabled using the *DisableRecordTask()* action.

Note that a *recordTask* is not created by a control point directly; therefore, the *input* support level below indicates PROHIBITED for all properties.

The table below lists all *recordTask*-related standard defined properties (*recordSchedule*-only properties are omitted from the table – see Appendix B, "AV Working Committee Extended Properties" for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, and REQUIRED) in this class for *recordTask* (RT) usage for each property.

Table C-14: *object.recordTask* Base Class Properties

Property Name	RT
Base Properties	
@id	√
title	√
class	√
additionalStatusInfo	O
cdsReference	O
cdsReference@link	R
Priority Properties	
priority	R
priority@orderedValue	O
desiredPriority	√
desiredPriority@type	√
Output Control Properties	
recordDestination	R
recordDestination@mediaType	R
recordDestination@targetURL	O
recordDestination@preference	R
desiredRecordQuality	√
desiredRecordQuality@type	√
General Properties	
recordScheduleID	R
recordedCDSObjectID	O
recordedCDSObjectID@link	O
Content ID Related Properties	
taskCDSObjectID	O
taskCDSObjectID@link	O
taskChannelID	R
taskChannelID@type	R
taskStartDateTime	R
taskDuration	R
taskProgramCode	O
taskProgramCode@type	R
recordQuality	R
recordQuality@type	R

Property Name	RT
Matched Content Criteria	
matchedName	O
matchedName@type	R
matchedID	O
matchedID@type	R
Matched Qualifying Criteria	
matchedRating	O
matchedRating@type	R
matchedEpisodeType	O
Content Control Properties	
taskStartDateTimeAdjust	O
taskDurationAdjust	O
taskDurationLimit	O
taskDurationLimit@effect	R
taskChannelMigration	O
taskTimeMigration	O
Task State Properties	
taskState	R
taskState@phase	R
taskState@startDateTimeMet	O
taskState@endDateTimeMet	O
taskState@recording	R
taskState@someBitsRecorded	R
taskState@someBitsMissing	R
taskState@firstBitsRecorded	O
taskState@lastBitsRecorded	O
taskState@fatalError	R
taskState@currentErrors	R
taskState@errorHistory	R
taskState@pendingErrors	R
taskState@infoList	R

Appendix D. EBNF Syntax Definitions (Normative)

The following sections define the syntax used for some of the properties and classes described in the previous sections. The syntax is formally defined using EBNF as described in Section 1.2.3, "Extended Backus-Naur Form".

D.1 Priority Syntax

Note: Due to possible future extensions, unknown value inputs MUST be gracefully ignored. In this case, the semantics of the "DEFAULT" value MUST be applied.

```
priority-value  ::= standard-value|
                    extended-value (* extended-value is only
                    applicable if priority@orderedValue is
                    supported *)
standard-value  ::= level|'DEFAULT'
level           ::= ('L' number)
number          ::= (* integer (n>0) *)
extended-value  ::= 'HIGHEST'|'LOWEST'|level-hi|level-low|object-id
level-hi        ::= level '_HI'
level-low       ::= level '_LOW'
object-id       ::= (* @id value *)
```

D.2 Date&time Syntax

```
sched-start     ::= date-time                |
                    day-of-yr-time           |
                    named-day-time           |
                    T-labeled-time           |
                    'NOW'
start-range     ::= (date-time|'NOW') '/' (date-time|'INFINITY')
date-time-range ::= date-time '/' date-time

duration        ::= 'P' [n 'D'] time
duration-long   ::= duration|'INFINITY'
duration-any    ::= duration|'INFINITY'|'ANY'
duration-adj    ::= ('+'|'-') duration
duration-range  ::= duration '/' duration-long

date-time       ::= yyyy '-' mm '-' dd T-labeled-time
day-of-yr-time  ::=          mm '-' dd T-labeled-time
named-day-time  ::=          named-day T-labeled-time

T-labeled-time  ::= 'T' time [zone]
time            ::= HH ':' MM ':' SS
zone            ::= 'Z'|(('+'|'-') HH ':' MM)
month-day       ::= mm '-' dd
named-day       ::= 'MON'|'TUE'|'WED'|'THU'|'FRI'|'SAT'|'SUN'|
                    'MON-FRI'|'MON-SAT'
n               ::= 1*DIGIT (* non-negative integer *)
yyyy            ::= 4DIGIT  (* 0001-9999 *)
mm              ::= 2DIGIT  (* 01-12 *)
```

```
dd              ::=  2DIGIT  (* 01-28, 01-29, 01-30, 01-31
                             based on month/year *)
HH              ::=  2DIGIT  (* 00-23 *)
MM              ::=  2DIGIT  (* 00-59 *)
SS              ::=  2DIGIT  (* 00-59 *)
```

D.3 Class Name Syntax

```
className       ::=  'OBJECT.' (sName|tName)
sName           ::=  'RECORDSCHEDULE.' (dName|qName)
tName           ::=  'RECORDTASK' ('.' shortName)*
dName           ::=  'DIRECT.' directName ('.' shortName)*
qName           ::=  'QUERY.' queryName ('.' shortName)*
directName      ::=  'MANUAL'|'CDSEPG'|'CDSNONEPG'|'PROGRAMCODE'
queryName       ::=  'CONTENTNAME'|'CONTENTID'
shortName       ::=  (* valid XML 1.0 name, excluding the characters
                     '.' (UTF-8 code 0x2E)
                     and
                     ':' (UTF-8 code 0x3A) *)
```

Appendix E. ScheduledRecording Service Relationship to ContentDirectory Service (Informative)

As noted in the specification, the only formal relationship between a ScheduledRecording service and a ContentDirectory service is through the *object.recordSchedule.direct.cdsEPG* and *object.recordSchedule.direct.cdsNonEPG* classes. The reason for keeping the ScheduledRecording service and ContentDirectory service as separate services is because they serve different purposes. The ScheduledRecording service is a service for creating a schedule of recording operations whereas the ContentDirectory service is a service for exposing content and its metadata. Therefore, the only formal dependency on a ContentDirectory service is to accommodate the case where a control point identifies recordable content on a ContentDirectory service and then instructs a sibling ScheduledRecording service to record that content.

Although a ScheduledRecording service and a ContentDirectory service are generally separated at the protocol layer, the two services can often interact in an out-of-band manner to realize some additional usages.

Showing Recorded Content in a ContentDirectory service: Vendors who are interested in making recorded content discoverable and network-consumable can expose the recorded content through the associated ContentDirectory service. The exact location where the recorded content will be exposed is determined by the implementation and is vendor-dependent.

Sending recorded bits to a ContentDirectory service: One methodology for sending recorded content to a ContentDirectory service (that is completely separate from the ScheduledRecording service) is to do the following: Start the process by having the control point invoke the *ContentDirectory::CreateObject()* action and obtain a *res@importUri* where binary data can be deposited via HTTP-POST. As a second step, the control point uses the *CreateRecordSchedule()* action with the appropriate destination type (*recordDestination* = "*MyNAS*", *recordDestination@mediaType* = "*HDD*", *recordDestination@targetURL* = *res@importUri*) to accommodate a URI that accepts HTTP-POST transmissions. When the ScheduledRecording service begins to record (or finishes recording) the ScheduledRecording service implementation can transmit the recorded bits using an HTTP-POST transaction. When the transmission is complete, the ContentDirectory service updates its metadata to allow rendering endpoints to play the content.

Scheduled recording from an external location: Vendors who want to use an external location as a source of recordable content can achieve this use case in the following manner. The control point obtains a URI that represents content that can be recorded. The control point creates a manual *recordSchedule* with the appropriate scheduling information and the URI as the input source in the *scheduledChannelID* property. At the instructed time, the ScheduledRecording service will download or stream the content data bytes from the URI to complete the recording.

Appendix F. ScheduledRecording Service Relationship to EPG (Informative)

ScheduledRecording service implementations are NOT REQUIRED to be tied to an Electronic Program Guide (EPG), as demonstrated by the *object.recordSchedule.direct.manual* class. However the subject of EPG data is an important discussion point for achieving a variety of use cases. This appendix does not exhaustively cover every relationship between a ScheduledRecording service and EPG, but it does discuss how the out-of-band EPG can fit into various use cases.

For scenarios where a control point creates a manual *recordSchedule*, the EPG directly provides information to the user. In some setups, the user may have to read an EPG in order to manually provide the control point with the scheduling and tuning input values. In other setups, the control point may have access to EPG data, allowing the control point to provide a user interface that is focused on the EPG, hiding the control point input values from the user. By design, the *object.recordSchedule.direct.manual* class does not require an EPG on the ScheduledRecording service because the ScheduledRecording service can resolve a manual recording type to discrete *recordTask* instances, without any additional information.

For scenarios where the control point creates a *object.recordSchedule.direct.cdsEPG* or *object.recordSchedule.query.contentName/contentID* class *recordSchedule*, the user still interacts with the EPG in some way. In some setups, the user will need to obtain a well-defined value (program or series ID, program title, etc.) from the EPG. In setups where the control point has access to EPG data, the user may not need to know about those well-defined values. Regardless of how the control point acquires the well-defined values, the ScheduledRecording service still needs to be able to translate this higher-level information into a *recordSchedule* object. In some setups, the ScheduledRecording service will have access to a complete EPG to assist with the creation of individual *recordSchedule* instances. In other setups, the ScheduledRecording service may have access to limited scheduling information on a broadcast stream that allows the ScheduledRecording service to know when to perform a recording operation. Intentionally, these types of recording types are designed around a particular EPG or broadcast system.

For scenarios where the control point creates a *object.recordSchedule.query.contentName* class of *recordSchedule*, the ScheduledRecording service will likely have direct access to EPG data. The reason is that this type of *recordSchedule* allows the control point to specify values that are not well-defined but still convey the desired content for recording. Therefore, the ScheduledRecording service generally needs to have additional intelligence to translate the *recordSchedule* into discrete *recordTask* instances. Often this process will include continually cross-referencing the *recordSchedule* properties with information in an EPG, and generating a *recordTask* instance every time a match is found between the matching criteria of the *recordSchedule* and an EPG item.

For scenarios where the control point creates a *object.recordSchedule.direct.cdsEPG* class of *recordSchedule*, the ScheduledRecording service and its sibling ContentDirectory service will likely have access to some kind of EPG data. The ContentDirectory service uses the EPG to expose recordable content to the control point/user. The user chooses a recordable object and then instructs the ScheduledRecording service to record using the *didl-lite:@id* value of the recordable object. Although it is generally useful for a ContentDirectory service to expose as much scheduling metadata as possible, the EPG data that is exposed by the ContentDirectory service is determined by the ContentDirectory/ScheduledRecording service implementer. As such, the only thing that a user needs is a control point that is capable of representing the recordable objects found in the ContentDirectory service.

Appendix G. AVDT Examples (Informative)

The following sections contain full-fledged examples of *AVDT XML Document* instances that a particular implementation might return in response to the invocation of the *GetAllowedValues()* action. All AV Working Committee defined values are printed using the *forum* character style. All device dependent values are printed using the *vendor* character style.

Note: These examples may be used as a starting point for real life implementations. Vendors may delete OPTIONAL property definitions that they do not support and add, delete and/or modify device dependent values to match their implementation.

G.1 *A_ARG_TYPE_RecordSchedule* AVDT Example

Note: This *A_ARG_TYPE_RecordSchedule* example is marked by a grey background.

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordSchedule", "*:*")
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordSchedule</dataStructType>

   <fieldTable>
      <field>
         <name>srs:@id</name>
         <dataType maxSize="256">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:title</name>
         <dataType maxSize="128">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>
```

```
      <field>
         <name>srs:class</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.DIRECT.MANUAL
               </allowedValue>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG
               </allowedValue>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG
               </allowedValue>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE
               </allowedValue>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME
               </allowedValue>
               <allowedValue>
                  OBJECT.RECORDSCHEDULE.QUERY.CONTENTID
               </allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:additionalInfo</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:cdsReference</name>
         <dataType maxSize="8192">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>
                     OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG
                  </value>
                  <value>
                     OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG
                  </value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:cdsReference@link</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
```

```
            <dependentField>
               <name>srs:cdsReference</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:Priority</name>
         <dataType maxSize="8">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>L1</allowedValue>
               <allowedValue>L2</allowedValue>
               <allowedValue>L3</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:priority@orderedValue</name>
         <dataType>xsd:unsignedInt</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:priority</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueRange>
               <minimum>1</minimum>
               <maximum>64</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredPriority</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredPriority@type</name>
               <valueList>
                  <value>PREDEF</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>L1</allowedValue>
               <allowedValue>L2</allowedValue>
               <allowedValue>L3</allowedValue>
               <allowedValue>HIGHEST</allowedValue>
               <allowedValue>LOWEST</allowedValue>
               <allowedValue>L1_HI</allowedValue>
               <allowedValue>L1_LO</allowedValue>
               <allowedValue>L2_HI</allowedValue>
```

```
            <allowedValue>L2_LO</allowedValue>
            <allowedValue>L3_HI</allowedValue>
            <allowedValue>L3_LO</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:desiredPriority@type</name>
            <valueList>
               <value>OBJECTID</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:desiredPriority@type</name>
      <dataType maxSize="16">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:desiredPriority</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>PREDEF</allowedValue>
            <allowedValue>OBJECTID</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination</name>
      <dataType maxSize="1024">xsd:string</dataType>
      <minCountTotal>1</minCountTotal>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <allowedValueList>
            <allowedValue>Hard Disk 1</allowedValue>
            <allowedValue>Hard Disk 2</allowedValue>
            <allowedValue>DVD Drive</allowedValue>
            <allowedValue>Remote Media Jukebox</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@mediaType</name>
      <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
      <maxCountTotal>3</maxCountTotal>
      <minListSizeTotal>1</minListSizeTotal>
      <maxListSizeTotal>4</maxListSizeTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <valueList>
               <value>HardDisk 1</value>
               <value>HardDisk 2</value>
```

```
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <minListSize>1</minListSize>
         <allowedValueList>
            <allowedValue>HDD</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <valueList>
               <value>DVD Drive</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <minListSize>1</minListSize>
         <maxListSize>4</maxListSize>
         <allowedValueList>
            <allowedValue>DVD+RW</allowedValue>
            <allowedValue>DVD-RW</allowedValue>
            <allowedValue>DVD-R</allowedValue>
            <allowedValue>DVD+R</allowedValue>
            <allowedValue>CD-R</allowedValue>
            <allowedValue>CD-RW</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <valueList>
               <value>Remote Media Jukebox</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <minListSize>1</minListSize>
         <maxListSize>2</maxListSize>
         <allowedValueList>
            <allowedValue>CD-R</allowedValue>
            <allowedValue>CD-RW</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@targetURL</name>
      <dataType>xsd:anyURI</dataType>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <anyValue></anyValue>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@preference</name>
      <dataType>xsd:unsignedInt</dataType>
```

```
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueRange>
               <minimum>1</minimum>
               <maximum>3</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredRecordQuality</name>
         <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
         <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredRecordQuality@type</name>
               <valueList>
                  <value>DEFAULT</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>4</maxListSize>
            <allowedValueList>
               <allowedValue>HD</allowedValue>
               <allowedValue>ED</allowedValue>
               <allowedValue>SD</allowedValue>
               <allowedValue>AUTO</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredRecordQuality@type</name>
               <valueList>
                  <value>ATSC</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>11</maxListSize>
            <allowedValueList>
               <allowedValue>1080p30</allowedValue>
               <allowedValue>1080p24</allowedValue>
               <allowedValue>1080i60</allowedValue>
               <allowedValue>720p60</allowedValue>
               <allowedValue>720p30</allowedValue>
               <allowedValue>720p24</allowedValue>
               <allowedValue>480p60</allowedValue>
               <allowedValue>480p30</allowedValue>
               <allowedValue>480p24</allowedValue>
               <allowedValue>480i60</allowedValue>
               <allowedValue>AUTO</allowedValue>
            </allowedValueList>
```

```
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredRecordQuality@type</name>
               <valueList>
                  <value>QLEVEL</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>4</maxListSize>
            <allowedValueList>
               <allowedValue>Q1</allowedValue>
               <allowedValue>Q2</allowedValue>
               <allowedValue>Q3</allowedValue>
               <allowedValue>AUTO</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredRecordQuality@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredRecordQuality</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>ATSC</allowedValue>
               <allowedValue>QLEVEL</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledCDSObjectID</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledCDSObjectID@link</name>
         <dataType maxSize="256">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduledCDSObjectID</name>
               <anyValue></anyValue>
```

```
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledChannelID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledChannelID@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduledChannelID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>ANALOG</allowedValue>
               <allowedValue>DIGITAL</allowedValue>
               <allowedValue>FREQUENCY</allowedValue>
               <allowedValue>SI</allowedValue>
               <allowedValue>LINE</allowedValue>
               <allowedValue>NETWORK</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledStartDateTime</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <maxCountTotal>2</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledDuration</name>
         <dataType maxSize="64">xsd:string</dataType>
```

```
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledProgramCode</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>
                     OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE
                  </value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledProgramCode@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduledProgramCode</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingName</name>
         <dataType maxSize="128">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingName@type</name>
```

```
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchingName</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>PROGRAM</allowedValue>
               <allowedValue>SERIES</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingName@subStringMatch</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchingName</name>
               <anyValue></anyValue>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingID@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchingID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>SI_PROGRAMID</allowedValue>
               <allowedValue>SI_SERIESID</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingChannelID</name>
         <dataType maxSize="256">xsd:string</dataType>
```

```
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingChannelID@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchingChannelID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>ANALOG</allowedValue>
               <allowedValue>DIGITAL</allowedValue>
               <allowedValue>FREQUENCY</allowedValue>
               <allowedValue>SI</allowedValue>
               <allowedValue>LINE</allowedValue>
               <allowedValue>NETWORK</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingStartDateTimeRange</name>
         <dataType maxSize="64">xsd:string</dataType>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchingDurationRange</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
```

```
      </field>

      <field>
         <name>srs:matchingRatingLimit</name>
         <dataType maxSize="16">xsd:string</dataType>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:matchingRatingLimit@type</name>
               <valueList>
                  <value>MPAA.ORG</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>G</allowedValue>
               <allowedValue>PG</allowedValue>
               <allowedValue>PG-13</allowedValue>
               <allowedValue>R</allowedValue>
               <allowedValue>NC-17</allowedValue>
               <allowedValue>NR</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:matchingRatingLimit@type</name>
               <valueList>
                  <value>RIAA.ORG</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue></allowedValue>
               <allowedValue>PA-EC</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:matchingRatingLimit@type</name>
```

```
            <valueList>
               <value>ESRB.ORG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>EC</allowedValue>
            <allowedValue>E</allowedValue>
            <allowedValue>E10+</allowedValue>
            <allowedValue>T</allowedValue>
            <allowedValue>M</allowedValue>
            <allowedValue>AO</allowedValue>
            <allowedValue>RP</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <dependentField>
            <name>srs:matchingRatingLimit@type</name>
            <valueList>
               <value>TVGUIDELINES.ORG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>TV-Y</allowedValue>
            <allowedValue>TV-Y7</allowedValue>
            <allowedValue>TV-Y7FV</allowedValue>
            <allowedValue>TV-G</allowedValue>
            <allowedValue>TV-PG</allowedValue>
            <allowedValue>TV-14</allowedValue>
            <allowedValue>TV-MA</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingRatingLimit@type</name>
      <dataType maxSize="32">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchingRatingLimit</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>MPAA.ORG</allowedValue>
            <allowedValue>RIAA.ORG</allowedValue>
            <allowedValue>ESRB.ORG</allowedValue>
            <allowedValue>TVGUIDLINES.ORG</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>
```

```
      <field>
         <name>srs:matchingEpisodeType</name>
         <dataType maxSize="9">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowedValueList>
               <allowedValue>ALL</allowedValue>
               <allowedValue>FIRST_RUN</allowedValue>
               <allowedValue>REPEAT</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:totalDesiredRecordTasks</name>
         <dataType>xsd:unsignedInt</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledStartDateTimeAdjust</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledDurationAdjust</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:activePeriod</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>
```

```
      <field>
         <name>srs:durationLimit</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:durationLimit@effect</name>
         <dataType maxSize="8">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:durationLimit</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>FIRST</allowedValue>
               <allowedValue>LAST</allowedValue>
               <allowedValue>SKIP</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:channelMigration</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:timeMigration</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
                  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
```

```
            </valueList>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:allowDuplicates</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings</name>
      <dataType>xsd:unsignedInt</dataType>
      <allowedValueDescriptor>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@latest</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@preAllocation</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@storedLifetime</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
```

```
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduleState</name>
         <dataType maxSize="16">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>OPERATIONAL</allowedValue>
               <allowedValue>COMPLETED</allowedValue>
               <allowedValue>ERROR</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduleState@currentErrors</name>
         <dataType csv="xsd:int">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduleState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <maxListSize>UNBOUNDED</maxListSize>
            <allowedValueRange>
               <minimum>100</minimum>
               <maximum>105</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduleState@abnormalTasksExist</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:scheduleState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:currentRecordTaskCount</name>
         <dataType>xsd:unsignedInt</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:totalCreatedRecordTasks</name>
```

```
         <dataType>xsd:unsignedInt</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:totalCompletedRecordTasks</name>
         <dataType>xsd:unsignedInt</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>
```

G.2 *A_ARG_TYPE_RecordTask* AVDT Example

Note: This *A_ARG_TYPE_RecordTask* example is marked by a light turquoise background.

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordTask", "*:*")
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordTask</dataStructType>

   <fieldTable>
      <field>
         <name>srs:@id</name>
         <dataType maxSize="256">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:title</name>
         <dataType maxSize="128">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
```

```
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:class</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>OBJECT.RECORDTASK</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:additionalInfo</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:cdsReference</name>
         <dataType maxSize="8192">xsd:string</dataType>
         <maxCountTotal>2</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskCDSObjectID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordedCDSObjectID</name>
               <anyValue></anyValue>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:cdsReference@link</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <maxCountTotal>2</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:cdsReference</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:Priority</name>
```

```
         <dataType maxSize="8">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>L1</allowedValue>
               <allowedValue>L2</allowedValue>
               <allowedValue>L3</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:priority@orderedValue</name>
         <dataType>xsd:unsignedInt</dataType>
         <allowedValueDescriptor>
            <allowedValueRange>
               <minimum>1</minimum>
               <maximum>64</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredPriority</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredPriority@type</name>
               <valueList>
                  <value>PREDEF</value>
               </valueList>
            </dependentField>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>L1</allowedValue>
               <allowedValue>L2</allowedValue>
               <allowedValue>L3</allowedValue>
               <allowedValue>HIGHEST</allowedValue>
               <allowedValue>LOWEST</allowedValue>
               <allowedValue>L1_HI</allowedValue>
               <allowedValue>L1_LO</allowedValue>
               <allowedValue>L2_HI</allowedValue>
               <allowedValue>L2_LO</allowedValue>
               <allowedValue>L3_HI</allowedValue>
               <allowedValue>L3_LO</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredPriority@type</name>
               <valueList>
                  <value>OBJECTID</value>
               </valueList>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
```

```
         <name>srs:desiredPriority@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredPriority</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>PREDEF</allowedValue>
               <allowedValue>OBJECTID</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>Hard Disk 1</allowedValue>
               <allowedValue>Hard Disk 2</allowedValue>
               <allowedValue>DVD Drive</allowedValue>
               <allowedValue>Remote Media Jukebox</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination@mediaType</name>
         <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
         <maxCountTotal>3</maxCountTotal>
         <minListSizeTotal>1</minListSizeTotal>
         <maxListSizeTotal>4</maxListSizeTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <valueList>
                  <value>HardDisk 1</value>
                  <value>HardDisk 2</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <allowedValueList>
               <allowedValue>HDD</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <valueList>
                  <value>DVD Drive</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>4</maxListSize>
```

```
            <allowedValueList>
               <allowedValue>DVD+RW</allowedValue>
               <allowedValue>DVD-RW</allowedValue>
               <allowedValue>DVD-R</allowedValue>
               <allowedValue>DVD+R</allowedValue>
               <allowedValue>CD-R</allowedValue>
               <allowedValue>CD-RW</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <valueList>
                  <value>Remote Media Jukebox</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>2</maxListSize>
            <allowedValueList>
               <allowedValue>CD-R</allowedValue>
               <allowedValue>CD-RW</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination@targetURL</name>
         <dataType>xsd:anyURI</dataType>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <anyValue></anyValue>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordDestination@preference</name>
         <dataType>xsd:unsignedInt</dataType>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordDestination</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueRange>
               <minimum>1</minimum>
               <maximum>3</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredRecordQuality</name>
         <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
```

```
        <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
        <allowedValueDescriptor>
           <dependentField>
              <name>srs:desiredRecordQuality@type</name>
              <valueList>
                 <value>DEFAULT</value>
              </valueList>
           </dependentField>
           <minCount>1</minCount>
           <minListSize>1</minListSize>
           <maxListSize>4</maxListSize>
           <allowedValueList>
              <allowedValue>HD</allowedValue>
              <allowedValue>ED</allowedValue>
              <allowedValue>SD</allowedValue>
              <allowedValue>AUTO</allowedValue>
           </allowedValueList>
        </allowedValueDescriptor>
        <allowedValueDescriptor>
           <dependentField>
              <name>srs:desiredRecordQuality@type</name>
              <valueList>
                 <value>ATSC</value>
              </valueList>
           </dependentField>
           <minCount>1</minCount>
           <minListSize>1</minListSize>
           <maxListSize>11</maxListSize>
           <allowedValueList>
              <allowedValue>1080p30</allowedValue>
              <allowedValue>1080p24</allowedValue>
              <allowedValue>1080i60</allowedValue>
              <allowedValue>720p60</allowedValue>
              <allowedValue>720p30</allowedValue>
              <allowedValue>720p24</allowedValue>
              <allowedValue>480p60</allowedValue>
              <allowedValue>480p30</allowedValue>
              <allowedValue>480p24</allowedValue>
              <allowedValue>480i60</allowedValue>
              <allowedValue>AUTO</allowedValue>
           </allowedValueList>
        </allowedValueDescriptor>
        <allowedValueDescriptor>
           <dependentField>
              <name>srs:desiredRecordQuality@type</name>
              <valueList>
                 <value>QLEVEL</value>
              </valueList>
           </dependentField>
           <minCount>1</minCount>
           <minListSize>1</minListSize>
           <maxListSize>4</maxListSize>
           <allowedValueList>
              <allowedValue>Q1</allowedValue>
              <allowedValue>Q2</allowedValue>
              <allowedValue>Q3</allowedValue>
              <allowedValue>AUTO</allowedValue>
           </allowedValueList>
        </allowedValueDescriptor>
```

```
      </field>

      <field>
         <name>srs:desiredRecordQuality@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:desiredRecordQuality</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>ATSC</allowedValue>
               <allowedValue>QLEVEL</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordScheduleID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordedCDSObjectID</name>
         <dataType maxSize="8192">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordedCDSObjectID@link<</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordedCDSObjectID</name>
               <anyValue></anyValue>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskCDSObjectID</name>
         <dataType maxSize="8192">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskCDSObjectID@link<</name>
         <dataType maxSize="1024">xsd:string</dataType>
```

```
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskCDSObjectID</name>
               <anyValue></anyValue>
            </dependentField>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskChannelID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskChannelID@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskChannelID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>ANALOG</allowedValue>
               <allowedValue>DIGITAL</allowedValue>
               <allowedValue>FREQUENCY</allowedValue>
               <allowedValue>SI</allowedValue>
               <allowedValue>LINE</allowedValue>
               <allowedValue>NETWORK</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskStartDateTime</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskDuration</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskProgramCode</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
```

```
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskProgramCode@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskProgramCode</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordQuality</name>
         <dataType maxSize="16">xsd:string</dataType>
         <minCountTotal>3</minCountTotal>
         <maxCountTotal>3</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordQuality@type</name>
               <valueList>
                  <value>DEFAULT</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:taskState@phase</name>
               <valueList>
                  <value>IDLE</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>HD</allowedValue>
               <allowedValue>ED</allowedValue>
               <allowedValue>SD</allowedValue>
               <allowedValue>UNKNOWN</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordQuality@type</name>
               <valueList>
                  <value>DEFAULT</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:taskState@phase</name>
               <valueList>
                  <value>ACTIVE</value>
                  <value>DONE</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>HD</allowedValue>
```

```
            <allowedValue>ED</allowedValue>
            <allowedValue>SD</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordQuality@type</name>
            <valueList>
               <value>ATSC</value>
            </valueList>
         </dependentField>
         <dependentField>
            <name>srs:taskState@phase</name>
            <valueList>
               <value>IDLE</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>1080p30</allowedValue>
            <allowedValue>1080p24</allowedValue>
            <allowedValue>1080i60</allowedValue>
            <allowedValue>720p60</allowedValue>
            <allowedValue>720p30</allowedValue>
            <allowedValue>720p24</allowedValue>
            <allowedValue>480p60</allowedValue>
            <allowedValue>480p30</allowedValue>
            <allowedValue>480p24</allowedValue>
            <allowedValue>480i60</allowedValue>
            <allowedValue>UNKNOWN</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordQuality@type</name>
            <valueList>
               <value>ATSC</value>
            </valueList>
         </dependentField>
         <dependentField>
            <name>srs:taskState@phase</name>
            <valueList>
               <value>ACTIVE</value>
               <value>DONE</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>1080p30</allowedValue>
            <allowedValue>1080p24</allowedValue>
            <allowedValue>1080i60</allowedValue>
            <allowedValue>720p60</allowedValue>
            <allowedValue>720p30</allowedValue>
            <allowedValue>720p24</allowedValue>
            <allowedValue>480p60</allowedValue>
            <allowedValue>480p30</allowedValue>
            <allowedValue>480p24</allowedValue>
            <allowedValue>480i60</allowedValue>
         </allowedValueList>
```

```
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordQuality@type</name>
               <valueList>
                  <value>QLEVEL</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:taskState@phase</name>
               <valueList>
                  <value>IDLE</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>Q1</allowedValue>
               <allowedValue>Q2</allowedValue>
               <allowedValue>Q3</allowedValue>
               <allowedValue>UNKNOWN</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordQuality@type</name>
               <valueList>
                  <value>QLEVEL</value>
               </valueList>
            </dependentField>
            <dependentField>
               <name>srs:taskState@phase</name>
               <valueList>
                  <value>ACTIVE</value>
                  <value>DONE</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>Q1</allowedValue>
               <allowedValue>Q2</allowedValue>
               <allowedValue>Q3</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:recordQuality@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:recordQuality</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>3</minCount>
            <maxCount>3</maxCount>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>ATSC</allowedValue>
               <allowedValue>QLEVEL</allowedValue>
            </allowedValueList>
```

```
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedName</name>
         <dataType maxSize="128">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedName@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchedName</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>PROGRAM</allowedValue>
               <allowedValue>SERIES</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedID@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchedID</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>SI_PROGRAMID</allowedValue>
               <allowedValue>SI_SERIESID</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedRating</name>
         <dataType maxSize="16">xsd:string</dataType>
         <maxCountTotal>2</maxCountTotal>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchedRating@type</name>
               <valueList>
                  <value>MPAA.ORG</value>
```

```
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>G</allowedValue>
            <allowedValue>PG</allowedValue>
            <allowedValue>PG-13</allowedValue>
            <allowedValue>R</allowedValue>
            <allowedValue>NC-17</allowedValue>
            <allowedValue>NR</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchedRating@type</name>
            <valueList>
               <value>RIAA.ORG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue></allowedValue>
            <allowedValue>PA-EC</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchedRating@type</name>
            <valueList>
               <value>ESRB.ORG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>EC</allowedValue>
            <allowedValue>E</allowedValue>
            <allowedValue>E10+</allowedValue>
            <allowedValue>T</allowedValue>
            <allowedValue>M</allowedValue>
            <allowedValue>AO</allowedValue>
            <allowedValue>RP</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchedRating@type</name>
            <valueList>
               <value>TVGUIDELINES.ORG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>TV-Y</allowedValue>
            <allowedValue>TV-Y7</allowedValue>
            <allowedValue>TV-Y7FV</allowedValue>
            <allowedValue>TV-G</allowedValue>
            <allowedValue>TV-PG</allowedValue>
            <allowedValue>TV-14</allowedValue>
            <allowedValue>TV-MA</allowedValue>
```

```
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedRating@type</name>
         <dataType maxSize="32">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:matchedRating</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>MPAA.ORG</allowedValue>
               <allowedValue>RIAA.ORG</allowedValue>
               <allowedValue>ESRB.ORG</allowedValue>
               <allowedValue>TVGUIDLINES.ORG</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:matchedEpisodeType</name>
         <dataType maxSize="8">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>ALL</allowedValue>
               <allowedValue>FIRST_RUN</allowedValue>
               <allowedValue>REPEAT</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskStartDateTimeAdjust</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskDurationAdjust</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskDurationLimit</name>
         <dataType maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskDurationLimit@effect</name>
```

```
         <dataType maxSize="8">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskDurationLimit</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>FIRST</allowedValue>
               <allowedValue>LAST</allowedValue>
               <allowedValue>SKIP</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskChannelMigration</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskTimeMigration</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState</name>
         <dataType maxSize="64">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue>IDLE.READY</allowedValue>
               <allowedValue>IDLE.ATRISK</allowedValue>
               <allowedValue>ACTIVE.TRANSITION.FROMSTART</allowedValue>
               <allowedValue>ACTIVE.TRANSITION.RESTART</allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.FROMSTART.OK
               </allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.FROMSTART.ATRISK
               </allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.RESTART.OK
               </allowedValue>
               <allowedValue>
                  ACTIVE.RECORDING.RESTART.ATRISK
               </allowedValue>
               <allowedValue>ACTIVE.NOTRECORDING</allowedValue>
               <allowedValue>DONE.FULL</allowedValue>
               <allowedValue>DONE.PARTIAL</allowedValue>
               <allowedValue>DONE.EMPTY</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>
```

```
      <field>
         <name>srs:taskState@phase</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue>IDLE</allowedValue>
               <allowedValue>ACTIVE</allowedValue>
               <allowedValue>DONE</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@startDateTimeMet</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@endDateTimeMet</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@recording</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@someBitRecorded</name>
         <dataType>xsd:boolean</dataType>
```

```
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@someBitsMissing</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@firstBitsRecorded</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@lastBitsRecorded</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@fatalError</name>
         <dataType>xsd:boolean</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
```

```
      </field>

      <field>
         <name>srs:taskState@currentErrors</name>
         <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue></allowedValue>
               <allowedValue>100</allowedValue>
               <allowedValue>101</allowedValue>
               <allowedValue>102</allowedValue>
               <!-- Additional vendor defined values go hear -->
            </allowedValueList>
            <allowedValueRange>
               <minimum>200</minimum>
               <maximum>204</maximum>
               <step>1</step>
            </allowedValueRange>
            <allowedValueRange>
               <minimum>300</minimum>
               <maximum>307</maximum>
               <step>1</step>
            </allowedValueRange>
            <allowedValueRange>
               <minimum>400</minimum>
               <maximum>404</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:taskState@errorHistory</name>
         <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:taskState</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <allowedValueList>
               <allowedValue></allowedValue>
               <allowedValue>100</allowedValue>
               <allowedValue>101</allowedValue>
               <allowedValue>102</allowedValue>
               <!-- Additional vendor defined values go here -->
            </allowedValueList>
            <allowedValueRange>
               <minimum>200</minimum>
               <maximum>204</maximum>
               <step>1</step>
            </allowedValueRange>
            <allowedValueRange>
               <minimum>300</minimum>
               <maximum>307</maximum>
```

```
            <step>1</step>
         </allowedValueRange>
         <allowedValueRange>
            <minimum>400</minimum>
            <maximum>404</maximum>
            <step>1</step>
         </allowedValueRange>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:taskState@pendingErrors</name>
      <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:taskState</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue></allowedValue>
            <allowedValue>100</allowedValue>
            <allowedValue>101</allowedValue>
            <allowedValue>102</allowedValue>
            <!-- Additional vendor defined values go here -->
         </allowedValueList>
         <allowedValueRange>
            <minimum>200</minimum>
            <maximum>204</maximum>
            <step>1</step>
         </allowedValueRange>
         <allowedValueRange>
            <minimum>300</minimum>
            <maximum>307</maximum>
            <step>1</step>
         </allowedValueRange>
         <allowedValueRange>
            <minimum>400</minimum>
            <maximum>404</maximum>
            <step>1</step>
         </allowedValueRange>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:taskState@infoList</name>
      <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:taskState</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue></allowedValue>
            <allowedValue>100</allowedValue>
            <allowedValue>101</allowedValue>
            <allowedValue>102</allowedValue>
            <!-- Additional vendor defined values go here -->
         </allowedValueList>
```

```
            <allowedValueRange>
               <minimum>200</minimum>
               <maximum>204</maximum>
               <step>1</step>
            </allowedValueRange>
            <allowedValueRange>
               <minimum>300</minimum>
               <maximum>307</maximum>
               <step>1</step>
            </allowedValueRange>
            <allowedValueRange>
               <minimum>400</minimum>
               <maximum>404</maximum>
               <step>1</step>
            </allowedValueRange>
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>
```

G.3 *A_ARG_TYPE_RecordScheduleParts* AVDT Example

Note: This *A_ARG_TYPE_RecordScheduleParts* example is marked by a white background.

Request:

```
GetAllowedValues("A_ARG_TYPE_RecordScheduleParts", "*:*" )
```

The following response will be generated:

Response:

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

   <contextID>
      uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
   </contextID>

   <dataStructType>A_ARG_TYPE_RecordScheduleParts</dataStructType>

   <fieldTable>
      <field>
         <name>srs:@id</name>
         <dataType maxSize="256">xsd:string</dataType>
         <minCountTotal>1</minCountTotal>
         <allowedValueDescriptor>
            <allowedValueList>
               <allowedValue></allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>
```

```
<field>
   <name>srs:title</name>
   <dataType maxSize="128">xsd:string</dataType>
   <minCountTotal>1</minCountTotal>
   <allowedValueDescriptor>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:class</name>
   <dataType maxSize="64">xsd:string</dataType>
   <minCountTotal>1</minCountTotal>
   <allowedValueDescriptor>
      <allowedValueList>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.DIRECT.MANUAL
         </allowedValue>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG
         </allowedValue>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG
         </allowedValue>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE
         </allowedValue>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME
         </allowedValue>
         <allowedValue>
            OBJECT.RECORDSCHEDULE.QUERY.CONTENTID
         </allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:desiredPriority</name>
   <dataType maxSize="1024">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField defaultDependency="1">
         <name>srs:desiredPriority@type</name>
         <valueList>
            <value>PREDEF</value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <defaultValue>DEFAULT</defaultValue>
      <allowedValueList>
         <allowedValue>DEFAULT</allowedValue>
         <allowedValue>L1</allowedValue>
         <allowedValue>L2</allowedValue>
         <allowedValue>L3</allowedValue>
         <allowedValue>HIGHEST</allowedValue>
         <allowedValue>LOWEST</allowedValue>
         <allowedValue>L1_HI</allowedValue>
         <allowedValue>L1_LO</allowedValue>
         <allowedValue>L2_HI</allowedValue>
```

```
            <allowedValue>L2_LO</allowedValue>
            <allowedValue>L3_HI</allowedValue>
            <allowedValue>L3_LO</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:desiredPriority@type</name>
            <valueList>
               <value>OBJECTID</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:desiredPriority@type</name>
      <dataType maxSize="16">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:desiredPriority</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <defaultValue>PREDEF</defaultValue>
         <allowedValueList>
            <allowedValue>PREDEF</allowedValue>
            <allowedValue>OBJECTID</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination</name>
      <dataType maxSize="1024">xsd:string</dataType>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <defaultValue>Hard Disk 2</defaultValue>
         <allowedValueList>
            <allowedValue>Hard Disk 1</allowedValue>
            <allowedValue>Hard Disk 2</allowedValue>
            <allowedValue>DVD Drive</allowedValue>
            <allowedValue>Remote Media Jukebox</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@mediaType</name>
      <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
      <maxCountTotal>3</maxCountTotal>
      <maxListSizeTotal>4</maxListSizeTotal>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:recordDestination</name>
            <valueList>
               <value>HardDisk 1</value>
               <value>HardDisk 2</value>
```

```
            </valueList>
         </dependentField>
         <maxListSize>1</maxListSize>
         <defaultValue>HDD</defaultValue>
         <allowedValueList>
            <allowedValue>HDD</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <valueList>
               <value>DVD Drive</value>
            </valueList>
         </dependentField>
         <maxListSize>4</maxListSize>
         <allowedValueList>
            <allowedValue>DVD+RW</allowedValue>
            <allowedValue>DVD-RW</allowedValue>
            <allowedValue>DVD-R</allowedValue>
            <allowedValue>DVD+R</allowedValue>
            <allowedValue>CD-R</allowedValue>
            <allowedValue>CD-RW</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <valueList>
               <value>Network Jukebox Recorder</value>
            </valueList>
         </dependentField>
         <maxListSize>2</maxListSize>
         <defaultValue>CD-R</defaultValue>
         <allowedValueList>
            <allowedValue>CD-R</allowedValue>
            <allowedValue>CD-RW</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@targetURL</name>
      <dataType>xsd:anyURI</dataType>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:recordDestination</name>
            <anyValue></anyValue>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:recordDestination@preference</name>
      <dataType>xsd:unsignedInt</dataType>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
```

```
            <name>srs:recordDestination</name>
            <anyValue></anyValue>
         </dependentField>
         <defaultValue>2</defaultValue>
         <allowedValueRange>
            <minimum>1</minimum>
            <maximum>3</maximum>
            <step>1</step>
         </allowedValueRange>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:desiredRecordQuality</name>
      <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
      <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:desiredRecordQuality@type</name>
            <valueList>
               <value>DEFAULT</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <minListSize>1</minListSize>
         <maxListSize>4</maxListSize>
         <defaultValue>AUTO</defaultValue>
         <allowedValueList>
            <allowedValue>HD</allowedValue>
            <allowedValue>ED</allowedValue>
            <allowedValue>SD</allowedValue>
            <allowedValue>AUTO</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:desiredRecordQuality@type</name>
            <valueList>
               <value>ATSC</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <minListSize>1</minListSize>
         <maxListSize>11</maxListSize>
         <allowedValueList>
            <allowedValue>1080p30</allowedValue>
            <allowedValue>1080p24</allowedValue>
            <allowedValue>1080i60</allowedValue>
            <allowedValue>720p60</allowedValue>
            <allowedValue>720p30</allowedValue>
            <allowedValue>720p24</allowedValue>
            <allowedValue>480p60</allowedValue>
            <allowedValue>480p30</allowedValue>
            <allowedValue>480p24</allowedValue>
            <allowedValue>480i60</allowedValue>
            <allowedValue>AUTO</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
      <allowedValueDescriptor>
```

```
            <dependentField>
               <name>srs:desiredRecordQuality@type</name>
               <valueList>
                  <value>QLEVEL</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <minListSize>1</minListSize>
            <maxListSize>4</maxListSize>
            <allowedValueList>
               <allowedValue>Q1</allowedValue>
               <allowedValue>Q2</allowedValue>
               <allowedValue>Q3</allowedValue>
               <allowedValue>AUTO</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:desiredRecordQuality@type</name>
         <dataType maxSize="16">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField defaultDependency="1">
               <name>srs:desiredRecordQuality</name>
               <anyValue></anyValue>
            </dependentField>
            <minCount>1</minCount>
            <defaultValue>DEFAULT</defaultValue>
            <allowedValueList>
               <allowedValue>DEFAULT</allowedValue>
               <allowedValue>ATSC</allowedValue>
               <allowedValue>QLEVEL</allowedValue>
            </allowedValueList>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledCDSObjectID</name>
         <dataType maxSize="1024">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
               </valueList>
            </dependentField>
            <minCount>1</minCount>
            <allowAny></allowAny>
         </allowedValueDescriptor>
      </field>

      <field>
         <name>srs:scheduledChannelID</name>
         <dataType maxSize="256">xsd:string</dataType>
         <allowedValueDescriptor>
            <dependentField>
               <name>srs:class</name>
               <valueList>
                  <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
```

```
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:scheduledChannelID@type</name>
      <dataType maxSize="16">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:scheduledChannelID</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>ANALOG</allowedValue>
            <allowedValue>DIGITAL</allowedValue>
            <allowedValue>FREQUENCY</allowedValue>
            <allowedValue>SI</allowedValue>
            <allowedValue>LINE</allowedValue>
            <allowedValue>NETWORK</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:scheduledStartDateTime</name>
      <dataType maxSize="64">xsd:string</dataType>
      <maxCountTotal>2</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:scheduledDuration</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
            </valueList>
         </dependentField>
         <minCount>1</minCount>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>
```

```
<field>
   <name>srs:scheduledProgramCode</name>
   <dataType maxSize="64">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>
               OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE
            </value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:scheduledProgramCode@type</name>
   <dataType maxSize="16">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:scheduledProgramCode</name>
         <anyValue></anyValue>
      </dependentField>
      <minCount>1</minCount>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:matchingName</name>
   <dataType maxSize="128">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:matchingName@type</name>
   <dataType maxSize="16">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:matchingName</name>
         <anyValue></anyValue>
      </dependentField>
      <minCount>1</minCount>
      <allowedValueList>
         <allowedValue>PROGRAM</allowedValue>
         <allowedValue>SERIES</allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
</field>
```

```
<field>
   <name>srs:matchingName@subStringMatch</name>
   <dataType>xsd:boolean</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:matchingName</name>
         <anyValue></anyValue>
      </dependentField>
      <defaultValue>1</defaultValue>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:matchingID</name>
   <dataType maxSize="256">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:matchingID@type</name>
   <dataType maxSize="16">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:matchingID</name>
         <anyValue></anyValue>
      </dependentField>
      <minCount>1</minCount>
      <allowedValueList>
         <allowedValue>SI_PROGRAMID</allowedValue>
         <allowedValue>SI_SERIESID</allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:matchingChannelID</name>
   <dataType maxSize="256">xsd:string</dataType>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
         </valueList>
      </dependentField>
      <allowAny></allowAny>
   </allowedValueDescriptor>
</field>

<field>
```

```
      <name>srs:matchingChannelID@type</name>
      <dataType maxSize="16">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchingChannelID</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>ANALOG</allowedValue>
            <allowedValue>DIGITAL</allowedValue>
            <allowedValue>FREQUENCY</allowedValue>
            <allowedValue>SI</allowedValue>
            <allowedValue>LINE</allowedValue>
            <allowedValue>NETWORK</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingStartDateTimeRange</name>
      <dataType maxSize="64">xsd:string</dataType>
      <maxCountTotal>3</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingDurationRange</name>
      <dataType maxSize="16">xsd:string</dataType>
      <maxCountTotal>4</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingRatingLimit</name>
      <dataType maxSize="16">xsd:string</dataType>
      <maxCountTotal>2</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
```

```
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
         </valueList>
      </dependentField>
      <dependentField>
         <name>srs:matchingRatingLimit@type</name>
         <valueList>
            <value>MPAA.ORG</value>
         </valueList>
      </dependentField>
      <allowedValueList>
         <allowedValue>G</allowedValue>
         <allowedValue>PG</allowedValue>
         <allowedValue>PG-13</allowedValue>
         <allowedValue>R</allowedValue>
         <allowedValue>NC-17</allowedValue>
         <allowedValue>NR</allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
         </valueList>
      </dependentField>
      <dependentField>
         <name>srs:matchingRatingLimit@type</name>
         <valueList>
            <value>RIAA.ORG</value>
         </valueList>
      </dependentField>
      <allowedValueList>
         <allowedValue></allowedValue>
         <allowedValue>PA-EC</allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:class</name>
         <valueList>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
            <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
         </valueList>
      </dependentField>
      <dependentField>
         <name>srs:matchingRatingLimit@type</name>
         <valueList>
            <value>ESRB.ORG</value>
         </valueList>
      </dependentField>
      <allowedValueList>
         <allowedValue>EC</allowedValue>
         <allowedValue>E</allowedValue>
         <allowedValue>E10+</allowedValue>
         <allowedValue>T</allowedValue>
         <allowedValue>M</allowedValue>
         <allowedValue>AO</allowedValue>
         <allowedValue>RP</allowedValue>
      </allowedValueList>
```

```
      </allowedValueDescriptor>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <dependentField>
            <name>srs:matchingRatingLimit@type</name>
            <valueList>
               <value>TVGUIDELINES.ORG</value>
            </valueList>
         </dependentField>
         <allowedValueList>
            <allowedValue>TV-Y</allowedValue>
            <allowedValue>TV-Y7</allowedValue>
            <allowedValue>TV-Y7FV</allowedValue>
            <allowedValue>TV-G</allowedValue>
            <allowedValue>TV-PG</allowedValue>
            <allowedValue>TV-14</allowedValue>
            <allowedValue>TV-MA</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingRatingLimit@type</name>
      <dataType maxSize="32">xsd:string</dataType>
      <maxCountTotal>2</maxCountTotal>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:matchingRatingLimit</name>
            <anyValue></anyValue>
         </dependentField>
         <minCount>1</minCount>
         <allowedValueList>
            <allowedValue>MPAA.ORG</allowedValue>
            <allowedValue>RIAA.ORG</allowedValue>
            <allowedValue>ESRB.ORG</allowedValue>
            <allowedValue>TVGUIDLINES.ORG</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:matchingEpisodeType</name>
      <dataType maxSize="8">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <allowedValueList>
            <allowedValue>ALL</allowedValue>
            <allowedValue>FIRST_RUN</allowedValue>
```

```
            <allowedValue>REPEAT</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:totalDesiredRecordTasks</name>
      <dataType>xsd:unsignedInt</dataType>
      <allowedValueDescriptor>
         <defaultValue>1</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:scheduledStartDateTimeAdjust</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <defaultValue>+P00:00:00</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:scheduledDurationAdjust</name>
      <dataType maxSize="16">xsd:string</dataType>
      <allowedValueDescriptor>
         <defaultValue>+P00:00:00</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:activePeriod</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <defaultValue>NOW/INFINITY</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:durationLimit</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
```

```
                <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <defaultValue>INFINITY</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:durationLimit@effect</name>
      <dataType maxSize="8">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:durationLimit</name>
            <anyValue></anyValue>
         </dependentField>
         <defaultValue>FIRST</defaultValue>
         <allowedValueList>
            <allowedValue>FIRST</allowedValue>
            <allowedValue>LAST</allowedValue>
            <allowedValue>SKIP</allowedValue>
         </allowedValueList>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:channelMigration</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <defaultValue>1</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:timeMigration</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <defaultValue>1</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
```

```
      <name>srs:allowDuplicates</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField>
            <name>srs:class</name>
            <valueList>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
               <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
            </valueList>
         </dependentField>
         <defaultValue>1</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings</name>
      <dataType>xsd:unsignedInt</dataType>
      <allowedValueDescriptor>
         <defaultValue>0</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@latest</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
         </dependentField>
         <defaultValue>1</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@preAllocation</name>
      <dataType>xsd:boolean</dataType>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
         </dependentField>
         <defaultValue>0</defaultValue>
         <allowAny></allowAny>
      </allowedValueDescriptor>
   </field>

   <field>
      <name>srs:persistedRecordings@storedLifetime</name>
      <dataType maxSize="64">xsd:string</dataType>
      <allowedValueDescriptor>
         <dependentField defaultDependency="1">
            <name>srs:persistedRecordings</name>
            <anyValue></anyValue>
         </dependentField>
         <defaultValue>ANY</defaultValue>
         <allowAny></allowAny>
```

```
         </allowedValueDescriptor>
      </field>

   </fieldTable>
</AVDT>
```