
**Information technology — UPnP
Device Architecture —**

**Part 20-14:
Audio video device control protocol —
Level 4 — Scheduled recording service**

Technologies de l'information — Architecture de dispositif UPnP —

*Partie 20-14: Protocole de contrôle de dispositif audio-vidéo —
Niveau 4 — Service d'enregistrement programmé*





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

CONTENTS

1	Scope	1
2	Normative references	2
3	Terms, definitions, symbols and abbreviations	5
3.1	Provisioning terms	5
3.2	Terms specific to ScheduledRecording service	5
3.2.1	CDS.....	5
3.2.2	EPG.....	6
3.2.3	SRS.....	6
3.2.4	CDS object	6
3.2.5	User Channel.....	6
3.2.6	Channel Group.....	6
3.2.7	Channel Line-up.....	6
3.2.8	object	6
3.2.9	class.....	6
3.2.10	object Modification	6
3.2.11	<u>recordSchedule</u>	6
3.2.12	Conflicting <u>recordSchedule</u>	6
3.2.13	<u>recordTask</u>	7
3.2.14	Conflicting <u>recordTask</u>	7
3.2.15	<u>recordScheduleParts</u>	7
3.2.16	Property-set Data Types.....	7
3.2.17	Property.....	7
3.2.18	Member Property	8
3.2.19	Supported Member Property.....	8
3.2.20	Multi-valued property.....	8
3.2.21	Single-valued property	8
3.2.22	XML Document	8
3.2.23	XML Fragment	9
3.2.24	actualScheduledStartDateTime.....	9
3.2.25	actualStartDateTime.....	9
3.2.26	actualScheduledEndDateTime	9
3.2.27	actualEndDateTime	9
3.2.28	actualScheduledDuration.....	9
3.2.29	Lexical Sort Order	10
3.2.30	Lexical Matching	10
3.2.31	Simple Non-case-sensitive Sort Order	10
3.2.32	Simple Non-case-sensitive Matching	11
3.2.33	Numeric Sort Order	11
3.2.34	Boolean Sort Order	11
3.2.35	Sequenced Sort	11
3.2.36	Sequenced Lexical Sort.....	11
3.2.37	Sequenced Numeric Sort.....	11
3.2.38	Lexical Numeric Sort	11
3.2.39	<u>type</u> Relationship Sort	12
3.3	Symbols	12
4	Notations and Conventions	12

ISO/IEC 29341-20-14:2017(E)

4.1	Notation	12
4.1.1	Data Types	12
4.1.2	Strings Embedded in Other Strings	12
4.1.3	Extended Backus-Naur Form	13
4.2	Derived Data Types	13
4.2.1	Summary	13
4.2.2	CSV Lists	14
4.3	Management of XML Namespaces in Standardized DCPs	15
4.3.1	Namespace Prefix Requirements	19
4.3.2	Namespace Names, Namespace Versioning and Schema Versioning	20
4.3.3	Namespace Usage Examples	22
4.4	Vendor-defined Extensions	22
4.4.1	Vendor-defined Action Names	22
4.4.2	Vendor-defined State Variable Names	23
4.4.3	Vendor-defined XML Elements and attributes	23
4.4.4	Vendor-defined Property Names	23
5	Service Modeling Definitions	23
5.1	ServiceType	23
5.2	ScheduledRecording Service Architecture	23
5.2.1	<u>recordSchedule</u>	23
5.2.2	<u>recordTask</u>	25
5.3	State Variables	26
5.3.1	State Variable Overview	26
5.3.2	<u>SortCapabilities</u>	28
5.3.3	<u>SortLevelCapability</u>	28
5.3.4	<u>StateUpdateID</u>	28
5.3.5	<u>LastChange</u>	29
5.3.6	<u>A ARG TYPE PropertyList</u>	32
5.3.7	<u>A ARG TYPE DataTypeID</u>	32
5.3.8	<u>A ARG TYPE ObjectID</u>	32
5.3.9	<u>A ARG TYPE ObjectIDList</u>	32
5.3.10	<u>A ARG TYPE PropertyInfo</u>	33
5.3.11	<u>A ARG TYPE Index</u>	33
5.3.12	<u>A ARG TYPE Count</u>	33
5.3.13	<u>A ARG TYPE SortCriteria</u>	33
5.3.14	<u>A ARG TYPE RecordSchedule</u>	33
5.3.15	<u>A ARG TYPE RecordTask</u>	34
5.3.16	<u>A ARG TYPE RecordScheduleParts</u>	34
5.4	Eventing and Moderation	35
5.5	Actions	35
5.5.1	Action Overview	35
5.5.2	<u>GetSortCapabilities()</u>	36
5.5.3	<u>GetPropertyList()</u>	37
5.5.4	<u>GetAllowedValues()</u>	38
5.5.5	<u>GetStateUpdateID()</u>	40
5.5.6	<u>BrowseRecordSchedules()</u>	40
5.5.7	<u>BrowseRecordTasks()</u>	45
5.5.8	<u>CreateRecordSchedule()</u>	47

5.5.9	<u>DeleteRecordSchedule()</u>	49
5.5.10	<u>GetRecordSchedule()</u>	50
5.5.11	<u>EnableRecordSchedule()</u>	51
5.5.12	<u>DisableRecordSchedule()</u>	52
5.5.13	<u>DeleteRecordTask()</u>	54
5.5.14	<u>GetRecordTask()</u>	54
5.5.15	<u>EnableRecordTask()</u>	55
5.5.16	<u>DisableRecordTask()</u>	57
5.5.17	<u>ResetRecordTask()</u>	58
5.5.18	<u>GetRecordScheduleConflicts()</u>	59
5.5.19	<u>GetRecordTaskConflicts()</u>	60
5.5.20	Common Error Codes	61
5.6	State Diagram of <u>recordTask</u>	62
5.6.1	A Full-Featured State Diagram	62
5.6.2	A Minimal-Implementation State Diagram	68
5.6.3	<u>recordTask</u> State Example	70
5.7	ScheduledRecording Service Priority Model	71
5.7.1	Introduction of the ScheduledRecording Service Priority Model	71
5.7.2	Ordered Priority within Each Priority Level	72
5.7.3	Setting the Initial Priority Level of a <u>recordSchedule</u>	73
5.7.4	Sorting <u>recordSchedule</u> Instances Based on their Current Priority Settings	75
5.8	Theory of Operation	76
5.8.1	Introduction	76
5.8.2	Checking the Capabilities of a ScheduledRecording Service	76
5.8.3	Adding a Scheduled Recording Entry to the List	88
5.8.4	Deleting a <u>recordSchedule</u>	106
5.8.5	Browsing <u>recordSchedule</u> and <u>recordTask</u> instances	106
5.8.6	Rating System	113
5.8.7	Conflict Detection and Resolution	113
6	XML Service Description	115
7	Test	125
Annex A	(normative) srs XML Document	126
A.1	<u>A_ARG_TYPE_RecordSchedule</u> AVDT XML Document	126
A.2	<u>A_ARG_TYPE_RecordTask</u> AVDT XML Document	127
A.3	<u>A_ARG_TYPE_RecordScheduleParts</u> AVDT XML Document	127
Annex B	(normative) AV Working Committee Extended Properties	129
B.1	Base Properties	129
B.1.1	<u>@id</u>	129
B.1.2	<u>title</u>	130
B.1.3	<u>class</u>	130
B.1.4	<u>additionalStatusInfo</u>	130
B.1.5	<u>cdsReference</u>	131
B.2	Priority Properties	132
B.2.1	<u>priority</u>	132
B.2.2	<u>desiredPriority</u>	133
B.3	Output Control Properties	135
B.3.1	<u>recordDestination</u>	135

ISO/IEC 29341-20-14:2017(E)

B.3.2	<u>desiredRecordQuality</u>	137
B.4	Content Identification Related Properties	140
B.4.1	<u>scheduledCDSObjectID</u>	140
B.4.2	<u>scheduledChannelID</u>	141
B.4.3	<u>scheduledStartDateTime</u>	143
B.4.4	<u>scheduledDuration</u>	144
B.4.5	<u>scheduledProgramCode</u>	144
B.5	Matching Content Criteria Properties	145
B.5.1	<u>matchingName</u>	145
B.5.2	<u>matchingID</u>	146
B.6	Matching Qualifying Criteria Properties	147
B.6.1	<u>matchingChannelID</u>	148
B.6.2	<u>matchingStartDateTimeRange</u>	149
B.6.3	<u>matchingDurationRange</u>	149
B.6.4	<u>matchingRatingLimit</u>	150
B.6.5	<u>matchingEpisodeType</u>	152
B.7	Content Control Properties	153
B.7.1	<u>totalDesiredRecordTasks</u>	153
B.7.2	<u>scheduledStartDateTimeAdjust</u>	154
B.7.3	<u>scheduledDurationAdjust</u>	154
B.7.4	<u>activePeriod</u>	155
B.7.5	<u>durationLimit</u>	155
B.7.6	<u>channelMigration</u>	156
B.7.7	<u>timeMigration</u>	157
B.7.8	<u>allowDuplicates</u>	157
B.8	Storage Related Properties	157
B.8.1	<u>persistedRecordings</u>	158
B.9	Schedule State Properties	159
B.9.1	<u>scheduleState</u>	159
B.9.2	<u>abnormalTasksExist</u>	161
B.10	Statistics Properties	161
B.10.1	<u>currentRecordTaskCount</u>	161
B.10.2	<u>totalCreatedRecordTasks</u>	161
B.10.3	<u>totalCompletedRecordTasks</u>	162
B.11	Task General Properties	162
B.11.1	<u>recordScheduleID</u>	162
B.11.2	<u>recordedCDSObjectID</u>	162
B.12	Task Content Identification Properties	163
B.12.1	<u>taskCDSObjectID</u>	164
B.12.2	<u>taskChannelID</u>	164
B.12.3	<u>taskStartDateTime</u>	165
B.12.4	<u>taskDuration</u>	166
B.12.5	<u>taskProgramCode</u>	166
B.12.6	<u>recordQuality</u>	166
B.13	Task Matched Content Criteria Properties	169
B.13.1	<u>matchedName</u>	169
B.13.2	<u>matchedID</u>	169
B.14	Task Matched Qualifying Criteria Properties	170
B.14.1	<u>matchedRating</u>	170

B.14.1.1	<u>matchedRating@type</u>	170
B.14.1.2	<u>matchedRating@equivalentAge</u>	171
B.14.2	<u>matchedEpisodeType</u>	171
B.15	Task Matched Content Control Properties	171
B.15.1	<u>taskStartDateTimeAdjust</u>	171
B.15.2	<u>taskDurationAdjust</u>	172
B.15.3	<u>taskDurationLimit</u>	172
B.15.3.1	<u>taskDurationLimit@effect</u>	172
B.15.4	<u>taskChannelMigration</u>	173
B.15.5	<u>taskTimeMigration</u>	173
B.16	Task State Properties	173
B.16.1	<u>taskState</u>	174
B.17	ContentDirectory Service Imported Properties	182
Annex C	(normative) AV Working Committee Class Definitions	186
C.1	Class Hierarchy	186
C.1.1	Relationships between Classes and Properties	188
C.1.2	<u>recordScheduleParts</u> Properties	189
C.1.3	<u>recordSchedule</u> Properties	193
C.1.4	<u>recordTask</u> Properties	197
C.2	Class Definitions	199
C.3	<u>object</u> Base Class	200
C.3.1	<u>object.recordSchedule</u> Class	201
C.3.2	<u>object.recordTask</u> Class	211
Annex D	(normative) EBNF Syntax Definitions	213
D.1	Priority Syntax	213
D.2	Date&time Syntax	213
D.3	Class Name Syntax	214
Annex E	(informative) ScheduledRecording Service Relationship to ContentDirectory Service	215
Annex F	(informative) ScheduledRecording Service Relationship to EPG	216
Annex G	(informative) AVDT Examples	217
G.1	<u>A_ARG_TYPE_RecordSchedule</u> AVDT Example	217
G.2	<u>A_ARG_TYPE_RecordTask</u> AVDT Example	234
G.3	<u>A_ARG_TYPE_RecordScheduleParts</u> AVDT Example	253
Annex H	(informative) Bibliography	269

List of Tables

Table 1 — Properties in XML.....	8
Table 2 — EBNF Operators.....	13
Table 3 — CSV Examples	15
Table 4 — Namespace Definitions.....	16
Table 5 — Schema-related Information.....	18
Table 6 — Default Namespaces for the AV Specifications	20
Table 7 — State Variables	26
Table 8 — Allowed values for the DataTypeID argument.....	28
Table 9 — Allowed Elements in <StateEvent> Element	29
Table 10 — Eventing and Moderation	35
Table 11 — Actions.....	35
Table 12 — Arguments for GetSortCapabilities()	36
Table 13 — Error Codes for GetSortCapabilities()	37
Table 14 — Arguments for GetPropertyList()	37
Table 15 — Error Codes for GetPropertyList()	38
Table 16 — Arguments for GetAllowedValues()	38
Table 17 — Error Codes for GetAllowedValues()	39
Table 18 — Arguments for GetStateUpdateID()	40
Table 19 — Error Codes for GetStateUpdateID()	40
Table 20 — Arguments for BrowseRecordSchedules()	40
Table 21 — Error Codes for BrowseRecordSchedules()	45
Table 22 — Arguments for BrowseRecordTasks()	45
Table 23 — Error Codes for BrowseRecordTasks()	47
Table 24 — Arguments for CreateRecordSchedule()	47
Table 25 — Error Codes for CreateRecordSchedule()	49
Table 26 — Arguments for DeleteRecordSchedule()	50
Table 27 — Error Codes for DeleteRecordSchedule()	50
Table 28 — Arguments for GetRecordSchedule()	50
Table 29 — Error Codes for GetRecordSchedule()	51
Table 30 — Arguments for EnableRecordSchedule()	52
Table 31 — Error Codes for EnableRecordSchedule()	52
Table 32 — Arguments for DisableRecordSchedule()	53
Table 33 — Error Codes for DisableRecordSchedule()	53
Table 34 — Arguments for DeleteRecordTask()	54
Table 35 — Error Codes for DeleteRecordTask()	54
Table 36 — Arguments for GetRecordTask()	55
Table 37 — Error Codes for GetRecordTask()	55
Table 38 — Arguments for EnableRecordTask()	56
Table 39 — Error Codes for EnableRecordTask()	56
Table 40 — Arguments for DisableRecordTask()	57
Table 41 — Error Codes for DisableRecordTask()	58

Table 42 — Arguments for <u>ResetRecordTask()</u>	58
Table 43 — Error Codes for <u>ResetRecordTask()</u>	59
Table 44 — Arguments for <u>GetRecordScheduleConflicts()</u>	59
Table 45 — Error Codes for <u>GetRecordScheduleConflicts()</u>	60
Table 46 — Arguments for <u>GetRecordTaskConflicts()</u>	60
Table 47 — Error Codes for <u>GetRecordTaskConflicts()</u>	61
Table 48 — Common Error Codes	61
Table 49 — <u>recordTask</u> State Timeline	71
Table 50 — Example 1: Fewer <u>recordSchedule</u> instances than the Number of Supported Priority Levels.	72
Table 51 — Example 2: More <u>recordSchedule</u> instances than the Number of Supported Priority Levels.	73
Table 52 — Existing <u>recordSchedule</u> Priorities	74
Table 53 — <u>desiredPriority</u> Property Set to “RS-C”	74
Table 54 — <u>desiredPriority</u> Property Set to “HIGHEST”, “L1_HI”, or “RS-A”	75
Table 55 — <u>desiredPriority</u> Property Set to “LOWEST”, “L3_LOW”, or “RS-B”	75
Table 56 — <u>desiredPriority</u> Property Set to “RS-C”	75
Table B.1 — Base Properties Overview	129
Table B.2 — Allowed values for the <u>class</u> Property	130
Table B.3 — Priority Properties	132
Table B.4 — Allowed values for the <u>priority</u> Property	132
Table B.5 — Primary allowed values for the <u>desiredPriority</u> Property	133
Table B.6 — Additional allowed values for the <u>desiredPriority</u> Property	134
Table B.7 — Allowed values for the <u>desiredPriority@type</u> Property	135
Table B.8 — Output Control Properties	135
Table B.9 — <u>desiredRecordQuality</u> Example	138
Table B.10 — Allowed values for the <u>desiredRecordQuality</u> Property	139
Table B.11 — Allowed values for the <u>desiredRecordQuality@type</u> Property	140
Table B.12 — Content Identification Related Properties	140
Table B.13 — Allowed values for the <u>scheduledChannelID@type</u> Property	142
Table B.14 — Matching Content Criteria Properties	145
Table B.15 — Allowed values for the <u>matchingName@type</u> Property	146
Table B.16 — Allowed values for the <u>matchingID@type</u> Property	147
Table B.17 — Matching Qualifying Criteria Properties	147
Table B.18 — Allowed values for the <u>matchingRatingLimit</u> Property Using the MPAA Rating System (<u>matchingRatingLimit@type</u> = “MPAA.ORG”)	150
Table B.19 — Allowed values for the <u>matchingRatingLimit</u> Property Using the RIAA Rating System (<u>matchingRatingLimit@type</u> = “RIAA.ORG”)	150
Table B.20 — Allowed values for the <u>matchingRatingLimit</u> Property Using the ESRB Rating System (<u>matchingRatingLimit@type</u> = “ESRB.ORG”)	151
Table B.21 — Allowed values for the <u>matchingRatingLimit</u> Property Using the TVGUIDELINES Rating System (<u>matchingRatingLimit@type</u> = “TVGUIDELINES.ORG”)	151
Table B.22 — Allowed values for the <u>matchingRatingLimit@type</u> Property	152
Table B.23 — Allowed values for the <u>matchingEpisodeType</u> Property	153
Table B.24 — Content Control Properties	

ISO/IEC 29341-20-14:2017(E)

Table B.25 — Allowed values for the durationLimit@effect Property	156
Table B.26 — Storage Related Properties	157
Table B.27 — Schedule State Properties	159
Table B.28 — Allowed values for the scheduleState Property	160
Table B.29 — Allowed values for the scheduleState@currentErrors Property	160
Table B.30 — Statistics Properties	161
Table B.31 — Task General Properties	162
Table B.32 — Task Content Identification Properties	163
Table B.33 — recordQuality Example	167
Table B.34 — Allowed values for the recordQuality Property	168
Table B.35 — Task Matched Content Criteria Properties	169
Table B.36 — Task Matched Qualifying Criteria Properties	170
Table B.37 — Task Matched Content Control Properties	171
Table B.38 — State Related Properties	173
Table B.39 — Allowed values for the attributes of the taskState Property	174
Table B.40 — Allowed values for the taskState Property	175
Table B.41 — Allowed values for the taskState@phase Property	176
Table B.42 — Allowed values for the taskState@xxx Properties	179
Table C.1 — Class Properties Overview for recordScheduleParts usage	189
Table C.2 — Class Properties Overview for recordSchedule usage	193
Table C.3 — Class Properties Overview for recordTask usage	197
Table C.4 — object Base Class Properties	200
Table C.5 — object.recordSchedule Base Class Properties	201
Table C.6 — object.recordSchedule.direct Class Properties	202
Table C.7 — object.recordSchedule.direct.manual Class Properties	203
Table C.8 — object.recordSchedule.direct.cdsEPG Class Properties	204
Table C.9 — object.recordSchedule.direct.cdsNonEPG Class Properties	206
Table C.10 — object.recordSchedule.direct.programCode Class Properties	207
Table C.11 — object.recordSchedule.query Class Properties	208
Table C.12 — object.recordSchedule.query.contentName Class Properties	209
Table C.13 — object.recordSchedule.query.contentID Class Properties	210
Table C.14 — object.recordTask Base Class Properties	212

List of Figures

Figure 1 — Creating a new recordSchedule	24
Figure 2 — Capability check	25
Figure 3 — Browse recordSchedule	25
Figure 4 — Delete a recordSchedule	25
Figure 5 — A Full-Featured State Diagram	63
Figure 6 — A Minimal-Implementation State Diagram	69
Figure C.1 — Class hierarchy for the ScheduledRecording service.	187

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see <http://www.iso.org/directives>).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of Standard, the meaning of the ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: [Foreword – Supplementary information](#)

ISO/IEC 29341-20-14 was prepared by UPnP Forum and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of ISO/IEC 29341 series, under the general title *Information technology — UPnP Device Architecture*, can be found on the [ISO web site](#).

Introduction

ISO and IEC draw attention to the fact that it is claimed that compliance with this document may involve the use of patents as indicated below.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of -these patent rights have assured ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC.

Intel Corporation has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Intel Corporation
Standards Licensing Department
5200 NE Elam Young Parkway
MS: JFS-98
USA – Hillsboro, Oregon 97124

Microsoft Corporation has informed IEC and ISO that it has patent applications or granted patents as listed below:

6101499 / US; 6687755 / US; 6910068 / US; 7130895 / US; 6725281 / US; 7089307 / US;
7069312 / US; 10/783 524 /US

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

Philips International B.V. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Philips International B.V. – IP&S
High Tech campus, building 44 3A21
NL – 5656 Eindhoven

NXP B.V. (NL) has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

NXP B.V. (NL)
High Tech campus 60
NL – 5656 AG Eindhoven

Matsushita Electric Industrial Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Matsushita Electric Industrial Co. Ltd.
1-3-7 Shiromi, Chuoh-ku
JP – Osaka 540-6139

Hewlett Packard Company has informed IEC and ISO that it has patent applications or granted patents as listed below:

5 956 487 / US; 6 170 007 / US; 6 139 177 / US; 6 529 936 / US; 6 470 339 / US; 6 571 388 / US; 6 205 466 / US

Information may be obtained from:

Hewlett Packard Company
1501 Page Mill Road
USA – Palo Alto, CA 94304

Samsung Electronics Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Digital Media Business, Samsung Electronics Co. Ltd.
416 Maetan-3 Dong, Yeongtang-Gu,
KR – Suwon City 443-742

Huawei Technologies Co., Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Huawei Technologies Co., Ltd.
Administration Building, Bantian Longgang District
Shenzhen – China 518129

Qualcomm Incorporated has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA – USA 92121

Telecom Italia S.p.A. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Telecom Italia S.p.A.
Via Reiss Romoli, 274
Turin - Italy 10148

Cisco Systems informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA – USA 95134

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29341-20-14:2017(E)

Original UPnP Document

Reference may be made in this document to original UPnP documents. These references are retained in order to maintain consistency between the specifications as published by ISO/IEC and by UPnP Implementers Corporation and later by UPnP Forum. The following table indicates the original UPnP document titles and the corresponding part of ISO/IEC 29341:

UPnP Document Title	ISO/IEC 29341 Part
UPnP Device Architecture 1.0	ISO/IEC 29341-1:2008
UPnP Device Architecture Version 1.0	ISO/IEC 29341-1:2011
UPnP Device Architecture 1.1	ISO/IEC 29341-1-1:2011
UPnP Device Architecture 2.0	ISO/IEC 29341-1-2
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2008
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2011
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaRenderer:2 Device	ISO/IEC 29341-3-2:2011
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2008
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2011
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2008
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2011
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2008
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2011
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP ScheduledRecording:2	ISO/IEC 29341-4-14:2011
UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2008
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2011
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16
UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2

UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13
UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11
UPnP ContentDirectory:3 Service	ISO/IEC 29341-14-12:2011
UPnP MediaServer:3 Device	ISO/IEC 29341-14-3:2011
UPnP ContentSync:1	ISO/IEC 29341-15-10:2011
UPnP Low Power Architecture:1	ISO/IEC 29341-16-1:2011
UPnP LowPowerProxy:1 Service	ISO/IEC 29341-16-10:2011

ISO/IEC 29341-20-14:2017(E)

UPnP LowPowerDevice:1 Service	ISO/IEC 29341-16-11:2011
UPnP QoS Architecture:3	ISO/IEC 29341-17-1:2011
UPnP QoSDevice:3 Service	ISO/IEC 29341-17-10:2011
UPnP QoSManager:3 Service	ISO/IEC 29341-17-11:2011
UPnP QoSPolicyHolder:3 Service	ISO/IEC 29341-17-12:2011
UPnP QoSDevice:3 Addendum	ISO/IEC 29341-17-13:2011
UPnP RemoteAccessArchitecture:1	ISO/IEC 29341-18-1:2011
UPnP InboundConnectionConfig:1 Service	ISO/IEC 29341-18-10:2011
UPnP RADAConfig:1 Service	ISO/IEC 29341-18-11:2011
UPnP RADASync:1 Service	ISO/IEC 29341-18-12:2011
UPnP RATAConfig:1 Service	ISO/IEC 29341-18-13:2011
UPnP RAClient:1 Device	ISO/IEC 29341-18-2:2011
UPnP RAServer:1 Device	ISO/IEC 29341-18-3:2011
UPnP RADiscoveryAgent:1 Device	ISO/IEC 29341-18-4:2011
UPnP SolarProtectionBlind:1 Device	ISO/IEC 29341-19-1:2011
UPnP TwoWayMotionMotor:1 Service	ISO/IEC 29341-19-10:2011
UPnP AV Architecture:2	ISO/IEC 29341-20-1
UPnP AVTransport:3 Service	ISO/IEC 29341-20-10
UPnP ConnectionManager:3 Service	ISO/IEC 29341-20-11
UPnP ContentDirectory:4 Device	ISO/IEC 29341-20-12
UPnP RenderingControl:3 Service	ISO/IEC 29341-20-13
UPnP ScheduledRecording:2 Service	ISO/IEC 29341-20-14
UPnP MediaRenderer:3 Service	ISO/IEC 29341-20-2
UPnP MediaServer:4 Device	ISO/IEC 29341-20-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-20-4
UPnP InternetGatewayDevice:2 Device	ISO/IEC 29341-24-1
UPnP WANIPConnection:2 Service	ISO/IEC 29341-24-10
UPnP WANIPv6FirewallControl:1 Service	ISO/IEC 29341-24-11
UPnP WANConnectionDevice:2 Service	ISO/IEC 29341-24-2
UPnP WANDevice:2 Device	ISO/IEC 29341-24-3
UPnP Telephony Architecture:2	ISO/IEC 29341-26-1
UPnP CallManagement:2 Service	ISO/IEC 29341-26-10
UPnP MediaManagement:2 Service	ISO/IEC 29341-26-11
UPnP Messaging:2 Service	ISO/IEC 29341-26-12
UPnP PhoneManagement:2 Service	ISO/IEC 29341-26-13
UPnP AddressBook:1 Service	ISO/IEC 29341-26-14
UPnP Calendar:1 Service	ISO/IEC 29341-26-15
UPnP Presense:1 Service	ISO/IEC 29341-26-16
UPnP TelephonyClient:2 Device	ISO/IEC 29341-26-2
UPnP TelephonyServer:2 Device	ISO/IEC 29341-26-3
UPnP Friendly Info Update:1 Service	ISO/IEC 29341-27-1
UPnP MultiScreen MultiScreen Architecture:1	ISO/IEC 29341-28-1
UPnP MultiScreen Application Management:1 Service	ISO/IEC 29341-28-10
UPnP MultiScreen Screen:1 Device	ISO/IEC 29341-28-2
UPnP MultiScreen Application Management:2 Service	ISO/IEC 29341-29-10
UPnP MultiScreen Screen:2 Device	ISO/IEC 29341-29-2
UPnP IoT Management and Control Architecture Overview:1	ISO/IEC 29341-30-1

ISO/IEC 29341-20-14:2017(E)

UPnP DataStore:1 Service	ISO/IEC 29341-30-10
UPnP IoT Management and Control Data Model:1 Service	ISO/IEC 29341-30-11
UPnP IoT Management and Control Transport Generic:1 Service	ISO/IEC 29341-30-12
UPnP IoT Management and Control:1 Device	ISO/IEC 29341-30-2
UPnP Energy Management:1 Service	ISO/IEC 29341-31-1

1 Scope

This service definition is compliant with the UPnP Device Architecture version 1.0 [14]. It defines a service type referred to herein as ScheduledRecording service.

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 shall 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.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

Latest version available

at: <http://www.upnp.org/schemas/av/AllowedTransformSettings.xsd>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] – *UPnP Device Architecture, version 1.0*, UPnP Forum, October 15, 2008.

Available at: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf>.

Latest version available at: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf>.

[15] – *XML Schema for ContentDirectory Structure and Metadata (DIDL-Lite)*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/didl-lite-v3-20130331.xsd>.

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

[16] – *XML Schema for ContentDirectory DeviceMode*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/dmo-v1-20101231.xsd>.

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

[17] – *XML Schema for ContentDirectory DeviceModeRequest*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/dmor-v1-20101231.xsd>.

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

[18] – *XML Schema for ContentDirectory DeviceModeStatus*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/dmos-v1-20101231.xsd>.

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

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

[20] – *XML Schema for ContentDirectory PermissionsInfo*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/pi-v1-20101231.xsd>.

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

[21] – *RenderingControl:3*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service-20130331.pdf>.

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

[22] – *XML Schema for RenderingControl LastChange Eventing*, UPnP Forum, December 31, 2010.

Available at: <http://www.upnp.org/schemas/av/rcs-event-v3-20101231.xsd>.

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

[23] – *XML Schema for ConnectionManager RendererInfo*, UPnP Forum, December 31, 2010.

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

[24] – *XML Schema for AVTransport PlaylistInfo*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/rpl-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rpl.xsd>.

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

[26] – *XML Schema for ScheduledRecording Metadata and Structure*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/srs-v2-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/srs.xsd>.

[27] – *XML Schema for ScheduledRecording LastChange Eventing*, UPnP Forum, September 30, 2008.
Available at: <http://www.upnp.org/schemas/av/srs-event-v1-20080930.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/srs-event.xsd>.

[28] – *XML Schema for RenderingControl TransformSettings*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/TransformSettings-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/TransformSettings.xsd>.

[29] – *XML Schema for ContentDirectory Metadata*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/upnp-v4-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/upnp.xsd>.

[30] – *The “xml:” Namespace*, November 3, 2004.
Available at: <http://www.w3.org/XML/1998/namespace>.

[31] – *XML Schema for the “xml:” Namespace*.
Available at: <http://www.w3.org/2001/xml.xsd>.

[32] – *Namespaces in XML*, Tim Bray, Dave Hollander, Andrew Layman, eds., W3C Recommendation, January 14, 1999.
Available at: <http://www.w3.org/TR/1999/REC-xml-names-19990114>.

[33] – *XML Schema Part 1: Structures, Second Edition*, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation, 28 October 2004.
Available at: <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[34] – *XML Schema Part 2: Data Types, Second Edition*, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004.
Available at: <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

[35] – *XML Schema for XML Schema*.
Available at: <http://www.w3.org/2001/XMLSchema.xsd>.

[36] – *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>.

[37] – *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>.

[38] – *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>.

[39] – *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>.

3 Terms, definitions, symbols and abbreviations

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

3.1 Provisioning terms

3.1.1

allowed

A

The definition or behavior is allowed.

3.1.2

conditionally allowed

CA

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

3.1.3

conditionally required

CR

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

3.1.4

required

R

The definition or behavior is required.

3.1.5

R/A

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

3.1.6

X

Vendor-defined, non-standard.

3.1.7

-D

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

3.1.8

CSV list (or CSV)

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

3.2 Terms specific to ScheduledRecording service

3.2.1 CDS

ContentDirectory service

3.2.2 EPG

Electronic Program Guide

3.2.3 SRS

ScheduledRecording service

3.2.4 CDS object

An object in a ContentDirectory service metadata hierarchy; that is: [item](#) or [container](#).

3.2.5 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.

3.2.6 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.

3.2.7 Channel Line-up

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

3.2.8 object

A [recordSchedule](#) or a [recordTask](#) (see definition of [recordSchedule](#) and [recordTask](#) below).

3.2.9 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 shall be present on that object and the allowed 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.

3.2.10 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).

3.2.11 [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.

3.2.12 Conflicting [recordSchedule](#)

A conflicting [recordSchedule](#) exists when one or more of its associated [recordTask](#) instances is in conflict with another [recordTask](#) instance.

3.2.13 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.

3.2.14 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) shall be taken into account when determining conflicts.

3.2.15 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.

3.2.16 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 allowed 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 [26], 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.

3.2.17 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 could 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 Annex B.

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 1 — Properties in XML

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

3.2.18 Member Property

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

3.2.19 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.

3.2.20 Multi-valued property

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

3.2.21 Single-valued property

Most independent properties are single-valued. This means that the property shall 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 could be multi-valued.

3.2.22 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 5.

For example, the phrase “*srs XML Document*” refers to an XML document based on the SRS Schema as defined in [26]. Such a document comprises a single <srs ...> root element, and it is allowed to be 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>
```

3.2.23 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 5.

For example, the phrase “*srs XML Fragment*” refers to a sequence of XML elements that are defined in the SRS Schema as defined in [26]:

```
<item id="..." ...>...</item>”
or
<recordDestination          mediaType="..."          preference="...">
...
</recordDestination>”
or
<title>...</title>
<class>...</class>
<...>...</...>
...
<...>...</...>”
```

3.2.24 actualScheduledStartTime

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

$$\text{actualScheduledEndTime} = \text{scheduledStartTime} + \text{scheduledStartTimeAdjust}$$

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

3.2.25 actualStartTime

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

$\text{actualStartTime} = \text{actualScheduledStartTime} + \text{any device-specific record startup latency.}$

3.2.26 actualScheduledEndTime

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

$$\text{actualScheduledEndTime} = \text{scheduledStartTime} + \text{scheduledDuration} + \text{scheduledDurationAdjust}$$

where scheduledStartTime 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.

3.2.27 actualEndTime

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

$\text{actualEndTime} = \text{actualScheduledEndTime} + \text{any device-specific record teardown latency.}$

3.2.28 actualScheduledDuration

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

$$\begin{aligned}
 actualScheduledDuration &= actualScheduledEndTime - actualScheduledStartTime \\
 &= \underline{scheduledDuration} + \underline{scheduledDurationAdjust} \\
 &\quad - \underline{scheduledStartTimeAdjust}
 \end{aligned}$$

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

3.2.29 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 [36], and the related standards [37] and [38]. The lexical sort algorithms are defined in [36]. It uses a secondary algorithm defined in [37] and supporting data tables defined in [38]. These three references together – [37], [36] and [38] – 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”.

3.2.30 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 clause 8 in [36].

3.2.31 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 shall 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) shall be true, at least one of (2) and (3) shall be true, and (2) should be true.

$$“A” \leq “a” < “B” \leq “b” < \dots < “Y” \leq “y” < “Z” \leq “z” \quad (1)$$

$$“A” < “a” < “B” < “b” < \dots < “Y” < “y” < “Z” < “z” \quad (2)$$

$$“A” = “a” < “B” = “b” < \dots < “Y” = “y” < “Z” = “z” \quad (3)$$

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 is summarized in the following relations, where *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*.

$\text{upper}(c) \leq \text{lower}(c)$ (4)

$\text{upper}(c) < \text{lower}(c)$ (5)

$\text{upper}(c) = \text{lower}(c)$ (6)

3.2.32 Simple Non-case-sensitive Matching

In a simple non-case-sensitive match, relation (1) above in subclause 3.2.31 shall be true, and relation (6) above in subclause 3.2.31 should be true.

3.2.33 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.

3.2.34 Boolean Sort Order

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

3.2.35 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.

3.2.36 Sequenced Lexical Sort

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

3.2.37 Sequenced Numeric Sort

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

3.2.38 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.

3.2.39 **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 subclause. The independent property subvalues are sorted according to the order specified in its subclause. Sorting of the independent property may vary with the value of the dependent property.

3.3 Symbols

3.3.1

::

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

4 Notations and Conventions

4.1 Notation

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

4.1.1 Data Types

Data type definitions come from three sources:

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

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

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

4.1.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that shall be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary characters in the string and/or its independent substrings. This document uses embedded strings in two contexts – Comma Separated Value (CSV) lists (see subclause 4.2.2) and property values in

search criteria strings. Escaping conventions use the backslash character, “\” (character code U+005C), as follows:

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

4.1.3 Extended Backus-Naur Form

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

4.1.3.1 Typographic conventions for EBNF

Non-terminal symbols are unquoted sequences of characters from the set of English upper and lower case letters, the digits “0” through “9”, and the hyphen (“-”). Character sequences between 'single quotes' are terminal strings and shall appear literally in valid strings. Character sequences between (*comment delimiters*) are English language definitions or supplementary explanations of their associated symbols. White space in the EBNF is used to separate elements of the EBNF, not to represent white space in valid strings. White space usage in valid strings is described explicitly in the EBNF. Finally, the EBNF uses the following operators in Table 2:

Table 2 — EBNF Operators

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

4.2 Derived Data Types

4.2.1 Summary

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

4.2.2 CSV Lists

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

- A list is represented as a **string** type (for state variables and action arguments) or xsd:string type (for properties).
- Commas separate values within a list.
- Integer values are represented in CSVs with the same syntax as the integer data type specified in [14] (that is: allowed leading sign, allowed leading zeroes, numeric US-ASCII)
- Boolean values are represented in state variable and action argument CSVs as either “0” for false or “1” for true. These values are a subset of the defined **boolean** data type values specified in [14]: 0, false, no, 1, true, yes.
- Boolean values are represented in property CSVs as either “0” for false or “1” for true. These values are a subset of the defined Boolean data type values specified in [34]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in 4.1.2.
- White space before, after, or interior to any numeric data type is not allowed.
- White space before, after, or interior to any other data type is part of the value.

Table 3 — CSV Examples

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

4.3 Management of XML Namespaces in Standardized DCPs

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

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

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

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

All of the namespaces used in this specification are listed in Table 4 and Table 5. For each such namespace, Table 4 gives a brief description of it, its name (a URI) and its defined “standard” prefix name. Some namespaces included in these tables are not directly used or referenced in this document. They are included for completeness to accommodate those situations where this specification is used in conjunction with other UPnP specifications to construct a complete system of devices and services. For example, since the ScheduledRecording service depends on and refers to the ContentDirectory service, the predefined “srs:” namespace prefix is included. The individual specifications in such collections all use the same standard prefix. The standard prefixes are also used in Table 5 to cross-reference additional namespace information. Table 5 includes each namespace’s valid XML document root element(s) (if any), its schema file name, versioning information (to be discussed in more detail below), and a link to the entry in Clause 2 for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 4. 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 4 — Namespace Definitions

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

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

Table 5 — Schema-related Information

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

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

4.3.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings shall use the standard namespace prefixes as declared in Table 4. In order to properly process the XML documents described herein, control points and devices shall use namespace-aware XML processors [32] for both reading and writing. As allowed by [32], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document may be different from the standard prefix. All devices shall be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. However, it is strongly recommended that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP

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

The assumed default namespace, if any, for each UPnP AV specification is given in Table 6.

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

Table 6 — Default Namespaces for the AV Specifications

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

4.3.2 Namespace Names, Namespace Versioning and Schema Versioning

The UPnP AV service specifications define several data structures (such as state variables and action arguments) whose format is an XML instance document that complies with one or more specific XML schemas, which define XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespace names that are defined by the AV Working Committee are URNs. See Table 4 for a current list of namespace names. Additionally, each namespace corresponds to an XML schema document that provides a machine-readable representation of the associated namespace to enable automated validation of the XML (state variable or action parameter) instance documents.

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

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

Whenever a new namespace version is created, a new XML schema document (.xsd) is created and published so that the new namespace definition is represented in a machine-readable form. Since a XML schema document is just a representation of a namespace definition, translation errors can occur. Therefore, it is sometime necessary to re-release a published schema in order to correct typos or other namespace representation errors. In order to easily identify the potential multiplicity of

schema releases for the same namespace, the URI of each released schema shall conform to the following format (called Form 1):

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

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

Table 5 identifies the URI formats for each of the namespaces that are currently defined by the UPnP AV Working Committee.

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

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

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

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

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

Form 3 of the undated schema URI is always linked to the most recent release of the schema that represents the highest version of the namespace that has been published. For example, on June 25, 2002 (20020625), the undated schema URI ".../av/rsc-event.xsd" was linked to the schema that is otherwise known as ".../av/rsc-event-v1-20020625.xsd". However, on May 31, 2006 (20060531), that same undated schema URI was linked to the schema that is otherwise known as ".../av/rsc-event-v2-20060531.xsd".

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

- All instance documents, whether generated by a service or a control point, shall use Form 3.

- All UPnP AV published schemas that reference other UPnP AV schemas shall also use Form 3.

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

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

ver "-" **yyyymmdd** where **ver** and **yyyymmdd** are described above.

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

4.3.3 Namespace Usage Examples

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

Example 1:

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

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

4.4 Vendor-defined Extensions

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

4.4.1 Vendor-defined Action Names

Vendor-defined action names shall begin with "X". Additionally, it should be followed by an ICANN assigned domain name owned by the vendor followed by the underscore

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

4.4.2 Vendor-defined State Variable Names

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

4.4.3 Vendor-defined XML Elements and attributes

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

4.4.4 Vendor-defined Property Names

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

5 Service Modeling Definitions

5.1 ServiceType

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

urn:schemas-upnp-org:service:*ScheduledRecording:2*

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

5.2 ScheduledRecording Service Architecture

5.2.1 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 can state: “record channel 15 at 4PM on

March 19, 2004,” while a more complex instruction can 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 shall add allowed 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 shall 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 shall return an error and the recordSchedule shall not be created.

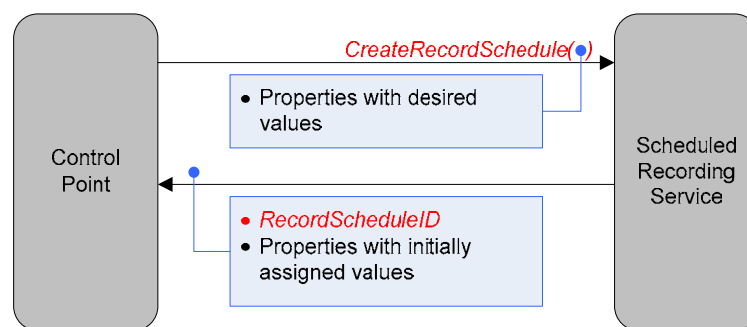


Figure 1 — Creating a new recordSchedule

Some properties are defined as allowed 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.

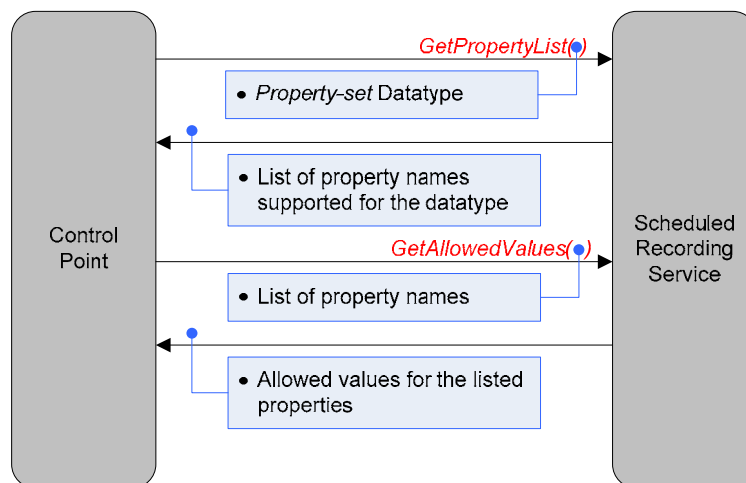


Figure 2 — Capability check.

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.

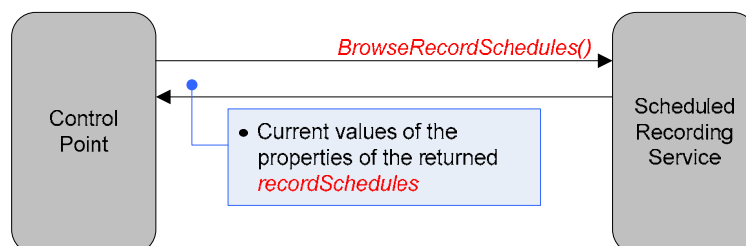


Figure 3 — Browse recordSchedule.

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

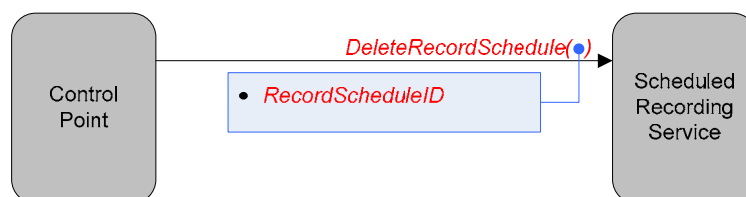


Figure 4 — Delete a recordSchedule

5.2.2 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 shall 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 shall delete all of its associated [recordTask](#) instances.

5.3 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 might be more helpful to read the action definitions before reading the state variable definitions.

5.3.1 State Variable Overview

Table 7 — State Variables

Variable Name	R/A ^a	Data Type	Allowed Value	Default Value	Eng. Units
SortCapabilities	R	string	CSV (string) See 5.3.2		
SortLevelCapability	R	ui4	See 5.3.3		
StateUpdateID	R	ui4	See 5.3.4		
LastChange	R	string	See 5.3.5		
A_ARG_TYPE_PropertyList	R	string	CSV (string) See 5.3.6		
A_ARG_TYPE_DataTypeID	R	string	See Table 8 and 5.3.7		
A_ARG_TYPE_ObjectID	R	string	See 5.3.8		
A_ARG_TYPE_ObjectIDList	CR ^a	string	CSV (string) See 5.3.9		
A_ARG_TYPE_PropertyInfo	R	string	See 5.3.10		

Variable Name	R/A ^a	Data Type	Allowed Value	Default Value	Eng. Units
<u>A_ARG_TYPE_Index</u>	<u>R</u>	<u>ui4</u>	See 5.3.11		
<u>A_ARG_TYPE_Count</u>	<u>R</u>	<u>ui4</u>	See 5.3.12		
<u>A_ARG_TYPE_SortCriteria</u>	<u>R</u>	<u>string</u>	CSV (<u>string</u>) See 5.3.13		
<u>A_ARG_TYPE_RecordSchedule</u>	<u>R</u>	<u>string</u>	See 5.3.14		
<u>A_ARG_TYPE_RecordTask</u>	<u>R</u>	<u>string</u>	See 5.3.15		
<u>A_ARG_TYPE_RecordScheduleParts</u>	<u>R</u>	<u>string</u>	See 5.3.16		
<i>Non-standard state variables implemented by an UPnP vendor go here.</i>	<u>X</u>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>
^a For a device this column indicates whether the state variable shall be implemented or not, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = non-standard, add <u>-D</u> when deprecated (e.g., <u>R-D</u> , <u>A-D</u>).					

Table 8 — Allowed values for the DataTypeID argument

Value	R/A ^a
" <u>A_ARG_TYPE_RecordSchedule</u> "	<u>R</u>
" <u>A_ARG_TYPE_RecordTask</u> "	<u>R</u>
" <u>A_ARG_TYPE_RecordScheduleParts</u> "	<u>R</u>
Vendor-defined	<u>X</u>
^a For a device this column indicates whether the state variable shall be implemented or not, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = non-standard, add <u>-</u> <u>D</u> when deprecated (e.g., <u>R-D</u> , <u>A-D</u>).	

5.3.2 SortCapabilities

This required 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.

5.3.3 SortLevelCapability

This required 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.

5.3.4 StateUpdateID

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

- a) StateUpdateID shall be incremented by 1 whenever any of 1) – 3) below occurs:
 - 1) A recordSchedule or recordTask is created or deleted.
 - 2) A recordSchedule or recordTask is modified, which means that one or more properties are added, deleted or had their value changed.
 - 3) 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.
- b) 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.
- c) The increment and the operation that caused it shall 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 1) – 3) below shall be true as seen by all external observations (including the returned values from this CreateRecordSchedule() invocation):

- 1) StateUpdateID is 10; and the new recordSchedule has not been created; and the new recordTask has not been created.

- 2) 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.
- 3) 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 might 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.

5.3.5 LastChange

Note: It is assumed that the default namespace for this subclause 5.3.5 of the specification is srs-lc.

This required 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. The schema for the *StateEvent XML Document* used in this state variable is defined in [27]. The XML header `<?xml version="1.0" ?>` is allowed. One root element, `<StateEvent>` shall contain 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 9. 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 shall follow the rules set forth in subclause 4.4. Note that future ScheduledRecording service specifications may define sub-elements for the elements. Also note that this state variable shall be properly escaped as defined in [39].

Table 9 — Allowed Elements in <StateEvent> Element

Element Name	Description
RecordScheduleCreated	A new <u>recordSchedule</u> is created.
RecordScheduleModified	One or more properties of a <u>recordSchedule</u> are modified (added, deleted or values are changed).
RecordScheduleDeleted	A <u>recordSchedule</u> is deleted.
RecordTaskCreated	A new <u>recordTask</u> is created.
RecordTaskModified	One or more properties of a <u>recordTask</u> are modified (added, deleted or values are changed).
RecordTaskDeleted	A <u>recordTask</u> is deleted.

Element Name	Description
<i>Vendor-defined</i>	See subclause 4.4.

Each element shall 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.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.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.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.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 currentRecordTaskCount 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.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 currentRecordTaskCount 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 currentRecordTaskCount 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.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.

5.3.6 A_ARG_TYPE_PropertyList

This required state variable is introduced to provide type information for various action arguments that contain a CSV list of property names. Namespace prefixes shall be included with all property names (see subclause 4.3). The exact semantics of these property names depend on the associated action.

5.3.7 A_ARG_TYPE_DataTypeID

This required state variable is introduced to provide type information for various action arguments that are used to identify a specific *property-set* data type (see subclause 3.2.16). An argument of type A_ARG_TYPE_DataTypeID can have the values listed in Table 8.

5.3.8 A_ARG_TYPE_ObjectID

This required 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.

5.3.9 A_ARG_TYPE_ObjectIDList

This conditionally required state variable shall be supported if the GetRecordScheduleConflicts() or GetRecordTaskConflicts() actions are implemented. The 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 shall be homogeneous).